Monograph:
Lim, Chee Peng and Harrison, R.F. (1995) An Incremental Adaptive Network for On-Line Supervised Learning and Probability Estimation. Research Report. ACSE Research Report 585 . Department of Automatic Control and Systems Engineering

---

An Incremental Adaptive Network for On-line, Supervised Learning and Probability Estimation

Chee Peng Lim and Robert F. Harrison

Department of Automatic Control and Systems Engineering
The University of Sheffield
Mappin Street, Sheffield S1 3JD
United Kingdom
E-mail: c.lim@sheffield.ac.uk, r.f.harrison@sheffield.ac.uk

Research Report No. 585
June 1995

Abstract

In this paper, a novel hybrid utilisation of the Fuzzy ARTMAP (FAM) neural network and the Probabilistic Neural Network (PNN) is proposed for on-line learning and probability estimation tasks. There are two distinct advantages of the hybrid network. First, FAM is used as an underlying clustering algorithm to classify the input patterns into different recognition categories during the learning phase, resulting in a significant reduction of pattern nodes required in the PNN. Second, a non-parametric posterior probability distribution estimation procedure, in accordance with the PNN paradigm (i.e. the Parzen-window estimator), is employed during the prediction phase, where a probabilistic interpretation corresponding to the Bayes decision theorem can be provided for the predictions of FAM. In addition, several modifications are addressed to integrate both the networks effectively into a unified platform for enhancing generalisation. This hybrid approach also realises an incremental learning system in which the necessity to specify a static network configuration *a priori* is eliminated, as the network is able to "grow" to accommodate new input patterns sequentially and can thus operate in non-stationary environments. The performance of the network is evaluated with benchmark classification tasks and the results are compared with other approaches. Simulation results indicate that this hybrid network is capable of asymptotically approaching the Bayes optimal classification rates.

Keywords: Fuzzy ARTMAP, Probabilistic Neural Network, Bayes strategy, pattern classification, on-line supervised learning, probability estimation

1 Introduction and Motivation

Systems architects continually strive to design machines with increasing capabilities of human-like autonomy and intelligence. Unfortunately, one of the main obstacles to successful machine learning is the memory erosion problem in real-time, potentially non-stationary environments. The fundamental issue is how a learning system is able safely to adapt to new information without corrupting or forgetting previously learned information. In essence, this is the so-called *stability-plasticity dilemma* addressed by Carpenter and Grossberg (1988, pp 77).

In this work, our goal is to design a generic autonomous learning system for general classification and probability estimation tasks. In general, discriminant analysis and Bayes decision theory have been studied for many years in the field of pattern recognition and classification.. The basic idea behind many of the approaches is to find an unknown discriminant function that best separates different classes from sample patterns, and the risk-weighted Bayes decision criterion is then employed to minimise the overall misclassification rate. Here, we adopt the neural-fuzzy approach to design an on-line learning classification system implemented in a neural network paradigm. The objective is to develop a Bayes optimal classifier that operates autonomously, on-line, in a non-stationary environment using neural networks and fuzzy set theory. The novelty of the network lies in its architecture which is able to "grow" autonomously and to accommodate input patterns sequentially, without any special hand-crafted architectural design. In addition, the network is capable of adaptively formulating complex non-linear decision boundary that asymptotically approach the Bayes optimal discriminant function, on-line, in non-stationary environments without prior knowledge of impending changes. In the following section, the issues associated with feedforward neural networks in on-line operation are discussed and an overview of two neural network architectures that are suitable for incremental learning is presented.

Feedforward networks, in particular the Multi-Layered Perceptron (MLP) and the Radial Basis Function (RBF) networks, have been extensively investigated for pattern classification and discrimination tasks. This is because they are able to form an input-output mapping from a set of exemplars and operate, in principle, as autonomous learning systems for function approximation. Theoretical results indicate that these networks possess architectures which are rich enough to approximate any arbitrary smooth enough function to an arbitrary degree of accuracy (Cybenko, 1989; Girosi and Poggio, 1990). Thus, it is likely that feedforward networks can offer a solution to the problem of developing a one-from-many classifier, and this is indeed the case as evidenced by the successful applications of such systems in various areas, e.g. Sejnowski and Rosenberg, 1987; Gorman and Sejnowski, 1988; Bounds *et al.*, 1990; Chen *et al.*, 1993a. However, there are several practical issues regarding how well a feedforward network can learn a discriminant function. In general, it is usually not known what finite size of network, i.e. number of hidden units, should be used to solve a given problem (Fujita, 1992). If the network is too small, it is not capable of forming an accurate approximation to the underlying function. On the other hand, if the network is too large, the network may specialise to the training samples and the solution obtained is likely to be a poor estimate of the actual function. Without any *a priori* information of the problem in hand, the trial-and-error method is usually used to "optimise" the performance of the network whilst some validity tests are conducted to verify its performance.

Another problem associated with the feedforward networks is that they are *static* after training. A feedforward network is generally trained off-line with some input-output data pairs. Once the training cycle is completed, no further learning is permitted when the network is in operation mode. Such an approach is viable when there is good reason to believe that the data environment is stationary and the data sample used in training is sufficiently representative. However, if the environment is non-stationary, it is necessary to repeatedly train the network whenever the environment changes, which is, of course, not a satisfactory approach. One way out of this dilemma is to determine the size of the network on-line in which the network is allowed to sequentially grow by itself in order to best approximate the underlying function in an incremental manner. In fact, this is the main principle of several augmented feedforward network architectures proposed in the literature (Chen *et al.*, 1993b; Kadirkamanathan and Niranjan, 1993; Fritzke, 1994; Shadafan and Niranjan, 1994, Sharkey and Sharkey, 1994).

The family of the Adaptive Resonance Theory (ART) networks is another example of incremental learning architecture which self-organises and self-stabilises in response to an arbitrary sequence of sample patterns in stationary and non-stationary environments (Carpenter and Grossberg, 1987a, 1987b). The key feature of the ART networks lies in its design of a novelty detector which measures the similarity between the prototype patterns stored in the network and the patterns presented to the network against a threshold. When the criterion is not satisfied, a new node is created to encode the input pattern as its prototype pattern. As a result, the number of nodes grows with time, subject to the novelty criterion, in an attempt to learn a good network configuration autonomously and on-line. As different tasks demand different capabilities from the network, this learning methodology, thus, avoids the needs to specify a pre-defined static network size or to re-train the network in non-stationary environments.

Bayesian decision theory is a classical statistical approach in pattern classification domain. The decision is expressed in probabilistic terms based on the criterion of minimising the overall misclassification rates weighted by some loss or risk factors (Fu, 1968; Duda and Hart, 1973). To implement the Bayes rule, it is necessary to assume that all the information on probabilities, namely the conditional probability densities and prior probabilities, are available. This information is then used to compute the *a posteriori* probabilities of a pattern belonging to a particular class. Unfortunately, the assumption that the forms of the underlying density functions are known is doubtful in most applications.

One solution to overcome the uncertainty is to estimate the unknown probabilities and probability density functions from the data samples. Several techniques have been developed, for instance the *parametric methods* that attempt to fit the samples to some parametrised functions and reduce the problem from one of function estimation to one of parameter estimation; and the *non-parametric methods* that directly estimate the functions from the samples by defining a continuous kernel function on every sample (Duda and Hart, 1973). In this context, feedforward networks, such as MLP and RBF, have also been extensively studied and their outputs have been shown implicitly to estimate the Bayesian *a posteriori* probabilities (White, 1989; Wan, 1990; Richard and Lippmann, 1991).

The Probabilistic Neural Network (PNN) (Specht, 1988, 1990) is a neural network architecture that applies the non-parametric method and implements the Bayes strategy in its paradigm. The PNN uses Parzen windows (Parzen, 1962) to estimate the probability

density functions from input-output data pairs. Training the network amounts to generating a node to code the input sample and associating it with its target class. Thus, the main advantage of the PNN is its speed, i.e. the ability to learn the training samples in one-pass and asymptotically achieve the Bayes optimal decision boundaries (Specht, 1990). In addition, once new information is available, the decision boundaries can be modified on-line without having to re-train the network. This autonomous learning property is comparable to that of ART.

There is a close similarity between the network connections of the PNN and Fuzzy ARTMAP (FAM), a variant of the supervised ART architectures. This similarity can be exploited to form a new system for on-line classification and prediction tasks. This paper focuses on the issues of how to integrate the PNN and FAM, together with some necessary modifications, in order to alleviate the shortcomings of each network and, at the same time, to preserve their self-organising and incremental learning properties. The advantages of the integration are two-fold: first, FAM is used as the underlying clustering algorithm to reduce the number of pattern nodes required in the PNN; second, the PNN is used as the probability estimation algorithm to provide probabilistic predictions from FAM. The objective is to design a hybrid network that is capable of classifying arbitrary input patterns into different categories on-line and to give a probabilistic interpretation of the predicted classes so that the Bayesian decision criterion can be applied in order to achieve optimal classification rates.

The structure of the rest of this paper is as follows: in section 2, the architectures of FAM and PNN are reviewed. In section 3, a hybrid system based on these two networks is presented, along with some proposed modifications and their rationale. In section 4, some simulation studies are included to demonstrate the capabilities of the hybrid network in on-line settings and to compare the performance with other approaches. This is followed by a summary and conclusions in section 5.

2 Fuzzy ARTMAP and Probabilistic Neural Networks

2.1 Fuzzy ARTMAP (FAM)

ART evolved from the analysis of how biological brains work to cope with changes in the environment in real-time and in a stable fashion. This neural network architecture was first proposed by Carpenter and Grossberg and, to date, the family of ART networks includes ART1, ART2, ART3 and Fuzzy ART for unsupervised learning (Carpenter and Grossberg, 1987a, 1987b, 1990; Carpenter *et al.*, 1991a) and ARTMAP, Fuzzy ARTMAP, ART-EMAP and Fusion ARTMAP for supervised learning (Carpenter *et al.*, 1991b, 1992, 1993b; Asfour *et al.*, 1993a, 1993b). In this paper, we concentrate on the FAM network because it has a robust architecture encompassing both fuzzy logic and the properties of ART, and is capable of handling analogue or binary input vectors, which may represent fuzzy or crisp sets of features.

(Figure 1 The FAM architecture)

FAM consists of two Fuzzy ART modules, ART_a and ART_b, which are linked by a map field, F^{ab}. Figure 1 shows a schematic diagram of the FAM network. Each fuzzy ART module has two layers of nodes: $F_{1a/1b}$ is the input layer whereas $F_{2a/2b}$ is a dynamic layer in which the number of nodes can be increased when necessary and every node encodes a prototype pattern representing a cluster of input samples. $F_{0a/0b}$ is a pre-processing layer in

which the input vectors are normalised between 0 and 1. The size of the input vectors must be kept constant in order to avoid the category proliferation problem (Moore, 1988; Carpenter *et al.*, 1991a). One of the recommended normalisation techniques is complement-coding, where the on-response and off-response amplitude information in the input vectors is preserved (Carpenter *et al.*, 1991a, 1992)

The notation used in this paper is as follows: M_a is the number of nodes in F_{1a} and N_a is the number of nodes in F_{2a}; $x_a \equiv (x_{a-1},\ldots,x_{a-M_a})$ is the F_{1a} activity vector or the Short Term Memory (STM) traces and $y_a \equiv (y_{a-1},\ldots,y_{a-N_a})$ is the F_{2a} activity vector; $w_{a\cdot j} \equiv (w_{a-j1},\ldots,w_{a-jM_a}), j = 1,\ldots,N_a$ is the jth ART_a weight vector or the Long Term Memory (LTM) traces. All these symbols apply to ART_b when the subscripts a and b are interchanged. In the map field, $x_{ab} \equiv (x_{ab-1},\ldots,x_{ab-N_b})$ is the F_{ab} activity vector and $w_{ab\cdot j} \equiv (w_{ab-j1},\ldots,w_{ab-jN_b}), j = 1,\ldots,N_a$ is the weight vector from the jth F_{2a} node to F_{ab}.

In general, the FAM algorithm can be divided into four phases:

(a) Initialisation

In a Fuzzy ART module, the weight vectors subsume both the bottom-up and top-down weight vectors of ART1 (Carpenter *et al.*, 1991a, 1992). In ART_a, each F_{2a} category node weight vector fans-out to all the nodes in F_{1a}. These weight vectors are initialised to unity, i.e.

$$w_{a-j1}(0) = \ldots = w_{a-jM_a}(0) = 1 \qquad j = 1,\ldots,N_a \qquad (1)$$

To operate in the conservative mode where recoding during learning will be minimised, the choice parameter, α_a, should be initialised close to 0, i.e. $\alpha_a \to 0$. The learning rate, β_a, and the baseline vigilance parameter, ρ_a, of ART_a, are set between 0 and 1. The vigilance parameter is a threshold which regulates the coarseness or fineness of the category prototypes to be formed. The same initialisation procedure is also applicable to ART_b. In the map field, the vigilance parameter, ρ_{ab}, is also initialised between 0 and 1, whereas the weight vectors from F_{2a} to F_{ab} are set to unity, i.e.

$$w_{ab-j1}(0) = \ldots = w_{ab-jN_b}(0) = 1 \qquad j = 1,\ldots,N_a \qquad (2)$$

Note that the number of nodes in F_{ab} is the same as the number of nodes in F_{2b} and there is a one-to-one permanent link between each corresponding pair of nodes.

(b) Activities in fuzzy ART

During supervised learning, ART_a receives an input vector and ART_b receives the associated target vector. After pre-processing, let $\boldsymbol{A}$ denote the pattern vector registered in F_{1a}, which is then transmitted to F_{2a} through the weight vector. A fuzzy choice function is used to measure the response of each F_{2a} prototype node as follows:

$$T_j(\boldsymbol{A}) = \frac{\left|\boldsymbol{A} \wedge w_{a-j}\right|}{\alpha_a + \left|w_{a-j}\right|} \qquad j = 1,\ldots,N_a \qquad (3)$$

The fuzzy MIN operator ($\wedge$) and the size $|\,.\,|$ are defined as: $(\boldsymbol{x} \wedge \boldsymbol{y})_i \equiv \min(x_i, y_i)$ and $|\boldsymbol{x}| \equiv \sum_i |x_i|$ (Zadeh, 1965).

The maximally responsive node is selected as the winner, denoted as node J, while all other nodes are shut down in accordance with the winner-take-all competition. The winning node then sends its weight vector to F_{1a} and a vigilance test is performed to test the

similarity between the F_{1a} activity vector and the input vector against the vigilance parameter:

$$\frac{|x_a|}{|A|} = \frac{|A \wedge w_{a-J}|}{|A|} \geq \rho_a \qquad (4)$$

where w_{a-J} is the weight vector of the Jth winning node in F_{2a}. If this novelty test is satisfied, resonance is said to occur and learning takes place. However, if the test fails, the winning node is inhibited and A is re-transmitted to F_{2a} to search for a new winner which is able to fulfil the vigilance test. If such a node does not exist, a new node is recruited to code the input vector. The same search cycle for the target vector goes on simultaneously in ART_b where a prototype node in F_{2b} that best matches the target vector will be found. In general, an independent Fuzzy ART module is employed as ART_b to self-organise the target vectors. However, in one-from-N classifications, ART_b can be replaced by a single layer containing N nodes. Then, the N-bit teaching stimulus can be coded to have unit value corresponding to the target category and zero for all others.

(c) Activities in the map field

Assuming that both ART_a and ART_b are active, a prediction from the F_{2a} winner is sent to F_{2b} via the map field. A map field vigilance test is then performed to determine the prediction, i.e.

$$\frac{|x_{ab}|}{|y_b|} = \frac{|y_b \wedge w_{ab-J}|}{|y_b|} \geq \rho_{ab} \qquad (5)$$

Note that if K is the winning node in F_{2b}, then the activity vector $y_{b-k} = 1$ *if* $k = K$, $y_{b-k} = 0$ *if* $k \neq K$; $k = 1,\ldots,N_b$. Thus, if the prediction is correct, $|x_{ab}| = 1$, or else $|x_{ab}| = 0$. A correct prediction will lead to learning in ART_a, ART_b and the map field. On the other hand, an incorrect prediction will initiate an activity called *match-tracking* in which a search cycle in ART_a is triggered. Match-tracking raises ρ_a to a value slightly greater than $|x_a|/|A|$ in order to cause the ART_a vigilance test to fail. A new node will then be chosen as the winner in F_{2a} and a fresh prediction is sent to F_{ab}. In other words, match-tracking provides a means to select a node which satisfies both the ART_a and the map field vigilance tests. If no such node exists, F_{2a} is shut down and the input vector will be ignored.

(d) Learning

Once search ends, the winning F_{2a} weight vector is updated according to:

$$w_{a-J}^{(new)} = \beta_a (A \wedge w_{a-J}^{(old)}) + (1 - \beta_a) w_{a-J}^{(old)} \qquad (6)$$

A node without any participation in learning is known as an uncommitted node. It will become a committed node when information is encoded in the LTMs. *Fast learning* corresponds to setting $\beta_a = 1$ in equation (6) at all time while *fast-commit, slow-recode learning* corresponds to setting $\beta_a = 1$ for an uncommitted node and $\beta_a < 1$ for a committed node. Note that equation (6) is also applicable to ART_b with obvious modifications. During fast learning, the map field weight vector is assigned thus $w_{ab-J} \rightarrow x_{ab}$. This means that the Jth winning node in F_{2a} is linked to the Kth winning node in F_{2b} via the map field and their association is permanent.

2.2 Modified Fuzzy ARTMAP

In our previous work, we indicated that FAM is unable to establish one-to-many mappings, i.e. forming an association from an F_{2a} prototype node to more than one F_{2b} target output via the map field (Lim and Harrison, 1994). In statistical pattern classification tasks, overlapping regions frequently occur in the input space in which a particular cluster may belong to more than one target output subject to different probabilities of class membership. This scenario is not significant with real-valued input vectors. This is because the probability of having two identical, real-valued patterns, whether from the same category or not, in a finite training set is vanishingly small. In other words, it is rarely the case to have two identical patterns belonging to different outputs in the data set, thus one-to-many mappings as described above would not occur. However, in cases of discrete-valued input vectors, especially binary-valued patterns of low dimension, the same input vector may be associated with distinct outputs with different probabilities.

We therefore propose a constraint on ρ_a, during match-tracking, as follows:

$$0 \leq \rho_a \leq \min\left(1, \frac{|A \wedge w_{a-J}|}{|A|} + \delta\right) \tag{7}$$

where δ is a small positive value that will lead to a search in ART_a. The effect of this constraint is to recruit a new node in F_{2a} to code the input vector instead of ignoring it as would occur in the original algorithm, thus establishing two similar prototype nodes to map to different outputs. In addition, a frequency measure scheme which records the number of correct predictions made by the nodes in F_{2a} is also introduced and this information is used to facilitate the selection of the winning node (Lim and Harrison, 1994).

2.3 The Probabilistic Neural Networks (PNN)

The PNN is a neural network architecture that can be used directly to implement the Bayes strategy in its learning paradigm for pattern classification problems. The PNN learns orders of magnitude more quickly than the MLP with back-propagation learning (Specht, 1990). The algorithm is able to form complex non-linear decision surfaces that will converge to the decision boundaries found by the Bayes optimal classifier (often the definition of optimality), when given sufficient training data. More importantly, it is possible to alter the decision boundaries on-line, thus, making the PNN suitable for incremental learning and real-time applications.

The key feature of the PNN is its ability to estimate the required probability density functions (pdfs) by using "Parzen windows" (Parzen, 1962) based on the data samples, i.e., a non-parametric density estimation procedure. Let $x_1,\ldots,x_n$ denote a sequence of samples of a one-dimensional, statistically independent random variable with unknown pdf, $f(x)$. Parzen (1962) showed that $f(x)$ can be estimated from the samples as:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\lambda_n}K\left(\frac{x-x_i}{\lambda_n}\right) \tag{8}$$

where $K(.)$ is a kernel estimator centred on each sample and λ_n is a smoothing parameter that depends on the sample size, n. Several constraints have to be satisfied for the estimation to asymptotically approach the true underlying function (Parzen, 1962; Specht, 1990). Cacoullos (1966) further extended Parzen's results to cover multi-dimensional

cases. Assuming that Gaussian kernel functions are chosen, the multivariate estimate for a sequence of M-dimensional vector random variables, $x_1,\ldots,x_n$, can be expressed as:

$$f(x)=\frac{1}{(2\pi)^{M/2}\sigma^{M}n}\sum_{i=1}^{n}\exp\left(-\frac{(x-x_i)^t(x-x_i)}{2\sigma^2}\right) \qquad (9)$$

where σ is the smoothing parameter of the kernel function.

(Figure 2 The PNN architecture)

The PNN realises the pdf estimator in a feedforward neural network architecture. Figure 2 depicts a schematic diagram of the PNN network for binary classification. It consists of four layers of nodes, namely input layer, pattern layer, summation layer and output layer. The input layer receives an M-dimensional vector, x, and distributes it to all the nodes in the pattern layer. The nodes in the pattern layer are divided into different categories depending on their output classes. These nodes perform a non-linear (exponential) transformation on dot-product as:

$$z_i = x \cdot w_i \qquad (10)$$

$$g(z_i)=\exp(\frac{z_i-1}{\sigma^2}) \qquad (11)$$

where $g(z_i)$ and w_i are the activation function and weight vector of pattern node i. If x and w_i are normalised to unit length, $g(z_i)$ is equivalent to $\exp\left(-\frac{(x-w_i)^t(x-w_i)}{2\sigma^2}\right)$ which is the same form as the exponential part in equation (9).

The summation nodes sum the activation levels of the pattern nodes corresponding to each class. To implement the Bayes decision criterion, the posterior probability, $p(C_k|x)$, can be computed according to:

$$p(C_k|x)=p(x|C_k)p(C_k)l(C_k|C_j) \qquad (12)$$

where the class pdf, $p(x|C_k)$, is weighted by the prior probability, $p(C_k)$, and the risk or loss factor, $l(C_k|C_j)$ (the risk of choosing C_k when C_j is the actual class, $j\neq k$), and assuming that the probability that x occurs is unity. While the prior probabilities can be estimated from the relative frequency of data samples used, the loss factors have to be determined beforehand to reflect the significance of incorrect decisions. The output node then selects the maximum *a posteriori* (MAP) probability to give a prediction of the target class. On the other hand, in a risk-weighted classification, one can assign different loss factors to each of the incorrect decisions to reflect the significance of misclassifications. In this case, the minimum-risk class computed by the Bayes rule is selected as the predicted target category (Melsa and Cohn, 1978).

Training the PNN is accomplished by generating a pattern node, connecting it to the summation node of the target category and assigning the input vector as the weight vector. Note that in addition to Gaussian kernel estimators, many alternative forms of kernel functions have been described by Specht (1990, 1992), such as the Euclidean distance, the "city-block" distance and the dot-product function which accommodates input vectors of varying lengths. Although the training procedure is very rapid and non-iterative, the number of pattern nodes required is the same as the number of training samples. While this is not an issue for small training sets, it will result in an explosive number of pattern nodes if

large or unbounded data sets are employed. Besides, the storage requirement and computational burden increase corresponding to the size of the data sets.

3 A Hybrid Network—Probabilistic Fuzzy ARTMAP (PFAM)

By inspecting Figures 1 and 2, it is clear that there is a close similarity in the network structure between FAM and PNN. The F_{1a} and F_{2a} layers of FAM correspond to the input and pattern layers of PNN, respectively, whereas the map field layer, F_{ab}, corresponds to the summation layer. In essence, in one-from-N classifications, each node in F_{2a} is permanently associated with only one node in F_{ab}, via the map field weights. This is then linked to the target output in F_{2b}. Thus, the F_{ab} nodes can be used to sum outputs from all the F_{2a} nodes connected to a particular target category, taking the role of the summation nodes in the PNN. It is this observation of the structural correspondence between the two networks which provides the basis for the proposed architecture.

A drawback of the PNN is the need to recruit a new pattern node for every input vector. As suggested in the literature (Burrascano, 1991; Specht, 1992; Musavi *et al.*, 1992a, 1993), this problem can be alleviated by using a clustering technique to reduce the number of pattern nodes required. The idea is to find a set of reference vectors, or prototypes, to represent sets of samples clustered in the input space. Instead of using all the input samples, the prototypes are used to situate the kernel functions. Consequently, the storage and computational burdens can be lightened while the ability to estimate the pdfs is maintained.

In general, ART networks implement a learning paradigm similar to the sequential leader clustering algorithm (Hartigan, 1975). This algorithm selects the first input pattern as the prototype of the first cluster. The next pattern is compared to the first prototype against a threshold of a similarity measure. If this criterion is not satisfied, a new cluster is established, where the input pattern is assigned as the new cluster prototype. This algorithm enables the number of clusters to grow with time to accommodate new input patterns sequentially. In view of the suitability of the learning algorithm and the similarity of the network structure, FAM provides a platform whereupon the PNN and FAM networks can be integrated. As a result, we propose a *Probabilistic Fuzzy ARTMAP (PFAM)* network for on-line applications, which operates in two phases. In the *learning phase*, FAM is utilised as the underlying, supervised, clustering algorithm to categorise similar input samples into clusters and the F_{2a} nodes are used to code the prototype samples. Subsequently, in the *prediction phase*, the PNN is utilised to perform probability estimation of the input vectors in relation to different output classes. Bayes' decision theorem may then be applied to select the maximum *a posteriori* output, or to implement some risk-weighted classification rules. Nevertheless, some modifications are essential so that the hybrid network can achieve better performance. The following section presents the necessary alterations and explains the rationale behind the procedures. It should be noted that Carpenter *et al.* (1993a) have examined the capabilities of FAM in probability estimation with various learning modes for the two noisy, nested spirals problem. However, our work here investigates the integration of FAM and PNN for on-line learning and prediction tasks where the Parzen-window technique is applied for probability estimation.

3.1 Map Field Activity

In FAM, only one of the elements of the map field weight vectors, w_{ab-j}, $j = 1,...,N_a$, has unit value which in turn indicates the link between category prototypes in F_{2a} to their outputs in F_{2b}. However, in order to accommodate clustering, the connections between the pattern nodes and the summation nodes have to be increased correspondingly (Specht, 1992). Thus, if the jth F_{2a} node successfully predicts and classifies an input pattern, the link should be incremented by one. Then, the kernel activation can be weighted by w_{ab-j} to represent the strength of each prototype. Besides, summing w_{ab-j} provides useful information on the prior class probabilities. In order to implement the above modifications, the STM vector, x_{ab}, in equation (5) is replaced by:

$$x_{ab} = y_b \wedge \frac{w_{ab-J}}{|w_{ab-J}|} \tag{13}$$

during map field prediction and the vigilance test, and assume that both ART_a and ART_b are active. Once the prediction is confirmed, the map field weight vector is updated according to:

$$w_{ab-J} = w_{ab-J} + x_{ab} \tag{14}$$

In addition, the frequency of data samples from each category can be calculated according to:

$$S_k = \sum_{j=1}^{N_a} w_{ab-jk} \tag{15}$$

$$S_{Total} = \sum_{k=1}^{N_b} S_k \tag{16}$$

Therefore, the prior probabilities of each of the categories can be estimated from the ratio of equations (15) and (16), i.e.:

$$p(C_k) = \frac{S_k}{S_{Total}} \tag{17}$$

3.2 Centre Estimation

The weight vectors in FAM do not reflect the centres, or means, of the input clusters represented by the F_{2a} nodes. As the algorithm utilises the fuzzy MIN operator in its learning rule in equation (6), the weight vector is a monotonically decreasing quantity and settles at the lower bound of the input vector or the weight vector ($A \wedge w_{a-J}$). In complement coding, the category boundaries have a hyper-rectangular shape where the lower vertices are represented by the weight vectors while the upper vertices are represented by the complement-coded part. The size of these hyper-rectangles grows with time but is governed by ρ_a (Carpenter *et al.*, 1991a, 1992). Figure 3 shows some of the possible category formations by the weight vectors.

(Figure 3 Category formation by complement-coded input vectors)

In PFAM, the weight vectors are suitable for fuzzy operations in the learning phase. However, during the prediction phase, since the weight vectors do not encode the proper locations of the cluster centroids in the input space, the accuracy of the pdfs approximated by the kernel functions may be affected. To obviate this shortcoming, the use of the Kohonen self-organising feature map learning algorithm (Kohonen, 1989) is proposed as

this method has been demonstrated to possess the ability to form "code-book" reference prototypes in vector quantisation.

A new set of weight vectors is introduced from F_{2a} to F_{1a} in PFAM and is called the kernel centre weight vectors,

$$w^K_{a-j} \equiv (w^K_{a-j1},\ldots,w^K_{a-jM_a}) \qquad j=1,\ldots N_a \tag{18}$$

During the learning phase, the original weight vector (hereafter referred to as category weight vector) of the Jth F_{2a} winning node is adaptively modified using equation (6), whereas its kernel centre weight vector is updated according to the Kohonen rule:

$$w^{K(new)}_{a-J} = w^{K(old)}_{a-J} + \eta(t)\left(A - w^{K(old)}_{a-J}\right) \tag{19}$$

where $\eta(t)$ is a monotonically decreasing scalar learning rate. These kernel centre vectors will be used in the prediction phase for the kernel functions to construct the pdfs.

3.3 Width Estimation

The widths or variances of the kernel functions play an important role in the Parzen-window probability estimation. They define the influence of receptive fields interpolated by the kernel functions from the centres and, hence, regulate the smoothness of the pdfs. In the PNN, a general smoothing parameter, σ, is used for all the kernels, which means that all the Gaussian kernels have the same covariance matrix, Σ_i; the covariance matrix is diagonal; and the eigenvalues are equal such that $\Sigma_i = \sigma \mathrm{I}$. In classification tasks, since it is essential for the kernel functions to maintain the local properties of the input space, the above assumptions may not yield optimal performance (Musavi *et al.*, 1992a, 1992b).

Musavi *et al.* (1992a, 1992b, 1993) propose that the Gram-Schmidt orthogonalisation technique can be used to estimate the widths of Gaussian kernel functions. The idea is to find the widths such that the amount of overlapping of kernel functions from distinct classes is minimised so that local neighbourhoods are preserved. In general, a Gaussian kernel function can be described as:

$$\phi(\|x-c_i\|) = \frac{1}{(2\pi)^{M/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(x-c_i)^t \Sigma_i^{-1}(x-c_i)\right) \tag{20}$$

where M is the dimension of the input space, x is the input vector, Σ_i and c_i are the covariance matrix and cluster centre of the ith kernel respectively. The parameter that needs to be estimated is the covariance matrix. In a multi-dimensional space, the loci of points of constant density, i.e. the constant potential surfaces, as defined by equating equation (20) to a constant, are hyper-ellipsoids. The eigenvectors of Σ_i define their principle axes whereas the square-root of the corresponding eigenvalues define the lengths of the principle axes. While kernels from the same category are allowed to overlap, it is crucial to separate kernels from distinct categories in order to avoid any significant overlapping, thus generalising the capability of the network to discriminate distinct classes.

One solution is to constraint the extension of kernels from the centres to the M nearest samples of other classes. The Gram-Schmidt orthogonalisation method can be used to construct Σ_i by finding the M nearest distinct neighbour. First, the covariance matrix for the ith kernel is decomposed as:

$$\Sigma_i = Q_i \Lambda_i Q_i^t \tag{21}$$

where the diagonal entries of Λ_i contain the eigenvalues of Σ_i, and the columns of Q_i contain its eigenvectors. Then, the eigenvectors and eigenvalues are determined from the following procedure. The first normalised axis can be formed by setting (Musavi *et al.*, 1992a, 1993)

$$y_1 = x_1 - c_i, \qquad e_1 = \frac{y_1}{\|y_1\|} \tag{22}$$

where x_1 is the nearest sample from other classes and c_i is the centre of the *i*th kernel. Note that $\|y_1\|$ and e_1 constitute the first eigenvalue and normalised eigenvector respectively. The second axis is formed by searching the next nearest sample of other classes that satisfies two conditions: it has the shortest normal to e_1 and its projection to e_1 has to be less than $\|y_1\|$. In general, for *M*-dimensional input space, the *M* axes can be obtained by using equations (22) and (23):

$$y_j = (x_j - c_i) - \sum_{k=1}^{j-1} e_k^t (x_j - c_i) e_k \qquad j = 2,\ldots,M \tag{23}$$

For all the (*j*-1) eigenvectors, the following projection criterion has to be met:

$$\left| e_k^t (x_j - c_i) \right| < \|y_k\| \qquad k = 1,\ldots,j-1 \tag{24}$$

Once the eigenvalues and eigenvectors are found, Σ_i can be constructed subject to certain constraints. The details of this procedure is explained in Musavi *et al.* (1992a, 1993).

The above approach employs a vector of smoothing parameters to adjust the window width of each kernel function. However, the storage requirement increases significantly compared with using a general smoothing parameter in the PNN, i.e. each prototype vector requires M^2 memory locations to store the covariance matrix. Besides, the algorithm is significantly more complex and demands more computational time. We have also found that the approach can sometimes result in an ill-conditioned covariance matrix especially when there are many centres from distinct classes clustered together in a small region of the input space. The constraint on the input vectors to be in the range of 0 to 1 further aggravates the situation. This phenomenon is described in Musavi *et al* (1992a) as "conservativeness in meshed clusters" where small widths are formed to prevent large overlapping. One way out of this problem, as proposed in Musavi *et al.* (1992a), is to set the eigenvalue as:

$$\lambda_i = \frac{\left\| x_{fi} - c_i \right\|^2}{r} \tag{25}$$

where x_{fi} is the farthest neighbour of other classes from the centre c_i and r is an "overlapping parameter". Equation (25) yields an enclosing hypersphere around each prototype centre, instead of a hyper-ellipsoid using the Gram-Schmidt orthogonalisation procedure. It provides a solution to preserving local neighbourhoods and, at the same time, reduces the computational and storage requirements. Note that to achieve 50% of the maximum value of a Gaussian function, one can assign $r = 2\ln\zeta$, where $1 < \zeta \le 2$ (Musavi *et al.*, 1992a). Another method of choosing r is to define the probability density enclosed by the hypersphere so that overlapping of neighbouring kernels can be controlled. As proposed in Musavi *et al.* (1992a, 1993), a transformation process has to be performed to evaluate the Gaussian function in order to obtain the value of r. According to equation (25), a different *scalar* smoothing parameter (instead of a vector) is used for each kernel, i.e.:

$$\Sigma_i = \sigma_i^2 I \tag{26}$$

where $\sigma_i^2 = \lambda_i$. Equation (20) is thus reduced to

$$\phi(\|x - c_i\|) = \frac{1}{(2\pi)^{M/2}\sigma^M} \exp\left(-\frac{(x-c_i)^t(x-c_i)}{2\sigma^2}\right) \tag{27}$$

where calculation of the determinant and inversion of the covariance matrix can be avoided.

Since PFAM operates in on-line settings, the kernel functions have to be re-evaluated whenever a new sample is learned by the network. This is to ensure that the pdfs constructed can best discriminate the input samples based on the current knowledge in the network. Consequently, to reduce the computational burden and to increase the speed of response in real-time applications, equation (27) is adopted. In addition, the local properties of individual clusters is still maintained to increase the generalisation of the network. Thus, a trade-off between generalisation and computational as well as storage overhead is achieved.

3.4 Fuzzification and Defuzzification

To combat the category proliferation problem, complement-coding is recommended and widely used in FAM. In fact, the problem is avoided as long as the size of the input vector is kept constant (Carpenter *et al.*, 1991a, 1992), i.e. for $\gamma > 0$,

$$|A| \equiv \sum_{i=1}^{M} |A_i| = \gamma \tag{28}$$

However, normalised vectors can sometimes obscure some crucial information, for example, co-linear vectors will become indistinguishable after normalisation (Wasserman, 1993). In practical applications, it is not unusual to encounter situations involving a few tens or even hundreds of features coded as an input vector. With complement-coding, the input space is doubled in size. In the context of density estimation, substantially more data samples are required to approximate high-dimensional pdfs accurately, and in fact increase in dimensionality often causes the estimation procedure to become intractable (Duda and Hart, 1973; Silverman, 1986). For example, in an experiment to approximate a standard multivariate Gaussian function using Gaussian kernel functions at the origin of a multi-dimensional space, in order for the relative mean-square-error of both the actual and estimated functions to fall below 0.1, one would require 4 data samples in 1 dimension, 19 data samples in 2 dimensions and 842,000 data samples in 10 dimensions (Silverman, 1986). It is, therefore, obvious that dimensionality can lead to severe difficulties in probability estimation in high-dimensional space.

Since FAM is developed based on the concept of fuzzy set theory, it thus motivates us to use fuzzy representation of features. In general, fuzzy set theory is developed as a means to represent and utilise uncertain information that is usually expressed in linguistic terms. To date, researchers have investigated the possibilities of incorporating neural networks and fuzzy logic into hybrid systems (Nie and Linkens, 1992; Lin, 1994; Wang, 1994). The aim is to automate and realise the integration of low-level learning ability of neural networks and the high-level, human-like expression and reasoning ability of fuzzy logic systems. FAM is an implementation of such system. Therefore, we propose to incorporate the fuzzification and defuzzification procedures into PFAM. The advantages of this approach are two-fold: the fuzzification process enables fuzzy representation of crisp features for classification during the learning phase; whereas the defuzzification process

allows the input space to be maintained for probability estimation during the prediction phase.

3.4.1 Fuzzification

A crisp set is a representation of variables whose membership function only takes two values {0,1}. A fuzzy set, however, is a generalisation of a crisp set where the membership function can take on any value from 0 to 1. Let Y be a collection of features, for example $Y = R^n$, and be called the universe of discourse. A fuzzy set F in Y is characterised by a membership function $m_F : Y \rightarrow [0,1]$, where $m_F(y)$ represents the grade or degree of membership of $y \in Y$ in the fuzzy set F.

(Figure 4 Fuzzy membership functions)

Figure 4 depicts three membership functions of fuzzy set, namely "small", "medium" and "large". The process of fuzzification performs a mapping from a crisp point into the membership functions, each containing different degree of membership in the interval [0,1]. For an m-dimensional input vector, $A = (a_1,\ldots,a_m)$, fuzzification transforms the element in each dimension into a point in an i-dimensional unit hypercube, $I^i = [0,1]^i$. In the above example, where $i = 3$, the input vector is decomposed into a $3m$-dimensional membership vector, i.e.

$$A = [m_{small}(a_1), m_{medium}(a_1), m_{large}(a_1),\ldots, m_{small}(a_m), m_{medium}(a_m), m_{large}(a_m)] \quad (29)$$

It should be noted that in this case, the number of nodes in F_{1a} increases proportional to the number of fuzzy partitions, i.e. $M_a = 3m$. However, the size of the input vector should be kept constant in agreement with equation (28). Thus, the total value of all the membership functions is constrained by $\sum m_i(y) = \gamma$, $\forall i, \forall y$, where i denotes the fuzzy set, "small", "medium" or "large", and y denotes the element of input vector, $a_j, j = 1,\ldots,m$. During learning, the membership vector is presented to the network for it to self-organise into different recognition categories, and is encoded in the F_{2a} nodes according to the algorithm of the learning phase. The category weight vectors, w_a, determine the category formation whereas the kernel centre weight vectors, w_a^K, store the centroids of the membership functions.

(Figure 5 Fuzzy centroid defuzzification)

3.4.2 Defuzzification

As mentioned in section 3.2, w_a^K encodes the centroid of each prototypical category of F_{2a}. to situate the kernel function. However, with fuzzification, the input space increases in size linearly with the fuzzy partition used to represent the input vector in F_{1a}. This presents a problem for the accurate approximation of the pdfs, as described at the beginning of this section. Thus, to avoid the increase in dimensionality, defuzzification of w_a^K is carried out to restore the membership vector in F_{1a} into a new set of kernel centre vectors confined to lie within the universe of discourse, in F_{0a}. These are called the defuzzified kernel centres, $d_{a-j}^K = (d_{a-j1}^K,\ldots,d_{a-jM_a}^K), j = 1,\ldots,N_a$. During the prediction phase, the defuzzified kernel centre and the input vector are transmitted from F_{0a} to F_{2a} directly, to perform the kernel estimation, bypassing the F_{1a} layer.

There are two commonly used defuzzification schemes, namely maximum-membership defuzzification and fuzzy centroid defuzzification (Kosko, 1992). The former method selects the maximal membership of the fuzzy sets used, i.e. $\max(m_i(y)), \forall i$. This is not suitable for the kernel estimator as it does not reflect the cluster centroid. As a result, the latter method is adopted. The *fuzzy centroid defuzzification* scheme computes the centre of mass of the fuzzy set i, i.e.:

$$c_i = \frac{\int y m_i(y) dy}{\int m_i(y) dy} \tag{30}$$

where the limits of integration correspond to the range of the fuzzy set i in the universe of discourse. To obtain the scalar centroid from three combined fuzzy sets of "small", "medium" and "large", the following equation can be used (Kosko, 1992):

$$v_k = \frac{\sum w_i c_i I_i}{\sum w_i I_i} \qquad \forall i \tag{31}$$

where $I_i = \int m_i(y) dy$ and c_i are from equation (30) and w_i represents a user-selected weighting parameter. Here, $w_i = 1, \forall i$. Figure 6 depicts a defuzzified centroid output from three fuzzy membership functions. The defuzzified centroids constitute a prototype vector to situate the kernel function for pdf estimation, i.e. for the jth F_{2a} node, the defuzzified kernel centre vector is

$$d^K_{a-jk} = v_k \quad k = 1,\ldots, M_a$$

It should be noted that the fuzzification and defuzzification procedures are an alternative to employing a purely fuzzy representation of input features for classification, in order to exploit the potential of FAM thoroughly, as well as to preserve dimensionality for probability estimation. If these procedures are excluded, then $\mathbf{w}^K_a$, instead of $\mathbf{d}^K_a$, can be used as centroids for the kernel estimator to approximate the pdfs.

(Figure 6 The PFAM architecture)

3.5 Summary of the algorithm

Figure 6 illustrates the network structures of PFAM to realise the hybrid utilisation of FAM and PNN in two different phases. The entire algorithm of the proposed PFAM for on-line learning and probability estimation is as follows:

Learning Phase:

(i) Fuzzify the input vector from F_{0a} to F_{1a}

(ii) Feed forward the fuzzified vector from F_{1a} to F_{2a} via $\mathbf{w}_a$, compute the match function as in equation (3), then select the winning node (denoted as node J)

(iii) Feed back the prototype of the winning node from F_{2a} to F_{1a} and perform the vigilance test as in equation (4)

(iv) If the vigilance test fails, trigger the search cycle and go to step (ii).
Else go to step (v)

(The above cycle goes on in ART_b simultaneously)

(v) Send a prediction from F_{2a} to F_{ab} and perform the map field vigilance test as in equation (5)

(vi) If the map field vigilance test fails, trigger match-tracking and go to step (ii) Else go to step (vii)

(vii) Learning: update the LTMs w_{a-j}, w_{ab-j} and w^K_{a-j}, according to equations (6), (14) and (19); update the LTM of the winning node in ART_b according to (6)

Prediction Phase:

(i) Register the input vector at F_{0a}, defuzzify $w^K_{a-j}, j = 1,\ldots,N_a$ to obtain the defuzzified kernel centre vectors, $d^K_{a-j}, j = 1,\ldots,N_a$, and send to F_{0a}

(ii) Feed forward the input and defuzzified centres from F_{0a} to the kernel functions in F_{2a}, bypassing the F_{1a} layer

(iii) Compute the width estimation and the kernel functions as in equations (25) and (27)

(iv) Sum the kernel estimates from F_{2a}, weighted by w_{ab}, to the corresponding categories in the map field to obtain the pdfs

(v) Compute the posterior probabilities for each category $p(C_k|x)$ according to equation (12).

(vi) Select the maximum *a posteriori* (MAP) or the minimum-risk estimate according to the Bayes rule and predict the target output in ART_b

Note that equation (7) should be taken into consideration in PFAM to implement one-to-many mappings, especially when dealing with binary input vectors. The fuzzification and defuzzification steps are optional operations depending on the nature of the problem in hand.

4 Simulation Studies

4.1 Statistical Pattern Classification

To evaluate the capabilities and effectiveness of PFAM, the experiment on separation of two Gaussian sources has been selected. Kohonen *et al.* (1988) formulated this experiment as a benchmark study for statistical pattern recognition with neural networks. The difficulties of the experiment are three-fold, namely large overlapping of the class distributions, high degree of non-linearity in the decision boundaries and high dimensionality of the input space. In their demonstration, two classes of Gaussian variables are generated, where classes C_1 and C_2 are represented by multivariate normal distributions with mean vectors, $\mu_1 = (0,\ldots,0)$ and $\mu_2 = (\zeta,0,\ldots,0)$. The variances of C_1 and C_2 are set equal to 1 and 4 respectively for all dimensions. Then, two sets of simulations, with $\zeta = 2.32$ ("easy task") and $\zeta = 0$ ("hard task"), were performed and the input dimension was increased from 2 to 8. Several networks, such as the MLP with back-propagation learning (BP), the Boltzmann machine (BM) and learning vector quantisation (LVQ), have been used and their results compared with the theoretical (Bayes) limits (Kohonen *et al.*, 1988).

In our demonstration, we have selected the "hard task" with 8-dimensional input samples as this represents a benchmark experiment in high dimension for statistical classification problems. The same experiment has been reported by Musavi *et al.* (1993), using a minimum error neural network (MNN), a PNN-like network structure with implementation of the Gram-Schmidt orthogonalisation, where the performance is compared with that of the PNN. Throughout the simulations, both ART_a and ART_b operate

in conservative and fast-learning mode, i.e. $\alpha_a = \alpha_b = 0.001$, and $\beta_a = \beta_b = 1.0$. The respective vigilance parameters are $\rho_a = 0.3$, $\rho_b = 1.0$, $\rho_{ab} = 1.0$. The overlapping parameter used in the prediction phase is set to 0.8. All the simulations employ a single-epoch, on-line learning approach with the operational cycle proceeds as follows: an input sample is first presented to ART_a with its target output to ART_b. Then, a prediction is sent to ART_b according to the algorithm of the prediction phase. The prediction is confirmed with the answer in ART_b, thus producing a classification result. Learning ensues to associate the input sample with its target output according to the algorithm of the learning phase.

Classification performance is assessed with a 1000-sample window to calculate the on-line results, e.g. the accuracy at sample 2000 is the percentage of correct predictions from trials 1001-2000. In all the simulations, a data set of 10000 input samples, i.e. 5000 samples for each class, are generated and the results are averages of 5 independent runs. It should be noted that in a stochastic process, a single time history characterising a random phenomenon is referred to as a single realisation. Each realisation represents only one of the many possible outcomes that may have occurred. Thus, to make a general statement about performance, averages must be taken across the ensemble, which should ideally be an infinite set of realisations. In addition, even though the problem under scrutiny may itself be stationary, the on-line predicted outcomes form a non-stationary process owing to the build-up of templates, especially in the early stages of classification (the so-called finite-operating-time problem). Nevertheless, we have empirically demonstrated that the windowed time average result is able consistently to approach the ensemble result and lie mostly within its 95% confidence interval (Lim and Harrison, 1994, 1995). During the simulations, the input vectors are fuzzified to three membership functions, as described in section 3.4.1, before they are presented to the PFAM network. As for FAM, the input vectors are first scaled between 0 and 1, and then complement-coded in the F_{0a} layer.

4.1.1 Results

(Table 1 Comparison of performance with various networks)

Table 1 illustrates a performance comparison of various networks, such as BP, BM2 (Boltzmann machine with binary-coded input vectors), LVQ (Kohonen *et al.*, 1988), MNN and PNN (Musavi *et al.*, 1993), as well as the results of FAM and PFAM from our simulations. All the networks, except FAM and PFAM, employed the off-line training method with different network configurations. The results of MNN and PNN in Musavi *et al.* (1993) used different training set sizes, but we tabulate the 300-sample case as it shows the best performance. They also mentioned that the performance of PNN could be improved by fine-tuning its smoothing parameter and an error-rate of 22% was achieved by Specht (Musavi *et al.*, 1993).

From Table 1, BM2 shows the best result, where the input range was divided into 20 subranges in each dimension and encoded with a binary coding scheme. As described by Kohonen *et al.* (1988), this is a "brute-force" approach and requires a massive network configuration and processing time. The performance of LVQ and MNN is comparatively equal, but is slightly inferior to that of PFAM. Likewise, PNN and FAM have similar error rates, whereas BP gives a better result compared to that of the two networks.

(Figure 7 On-line results of the Gaussian sources separation simulations)

Figure 7 depicts a comparison of on-line performance between FAM and PFAM. The results are an average of 5 runs where the standard deviations are represented by error bars. Although the standard deviations are estimated from a small sample size, it serves as an indication of the extent in which the results tend to vary. In general, PFAM shows an improvement in accuracy as the number of input samples increases and it asymptotically approximates the Bayes limits. Besides, the dispersion of the standard deviation becomes smaller with increasing number of samples. It can be seen that PFAM outperforms FAM, with a difference of 15% at the end of the experiments. However, PFAM also creates substantially more committed F_{2a} nodes than FAM, with a total of 293 and 88 nodes respectively. This may be due to the increase in dimension after the fuzzification process. Since the number of committed nodes represents "knowledge" stored in the network, in order to "fairly" compare the performance of both networks, additional experiments were conducted with a higher value of the vigilance parameter so that more prototypes can be established by FAM. By increasing ρ_a of FAM from 0.3 to 0.69, the two networks have a similar number of F_{2a} nodes throughout the experiments as depicted in Figure 7. Nevertheless, despite having an equivalent number of prototypes, there was no significant improvement in the performance of FAM.

4.2 Waveform Recognition

The waveform recognition task is introduced by Breiman *et al.* (1984) and is extensively evaluated with the decision tree classifiers advocated in their book, Classification and Regression Trees (CART). This domain is selected because it represents a high-dimensional task in a noisy environment, and it has also been recognised as a "difficult concept to learn" (Murphy and Aha, 1995). It is a three-class problem based on three triangular waveforms, $w_1(t)$, $w_2(t)$ and $w_3(t)$. These waveforms are represented by 21-dimensional measurement vectors, $\boldsymbol{x} = (x_1,...,x_{21})$, with random noise added. Each class is a combination of two base waveforms as depicted in Figure 8. To obtain a measurement vector, a uniformly distributed random variable, μ, and 21 normally distributed random variables with mean 0 and variance 1, $\varepsilon_1,...,\varepsilon_{21}$, are generated. Class 1 waveforms are represented by (Breiman *et al.*, 1984, pp. 49-55):

Class 1 $\quad x_i = \mu w_1(i) + (1-\mu) w_2(i) + \varepsilon_i, \quad i = 1,...,21$

Similarly, waveforms of classes 2 and 3 can be generated with the above procedure, i.e.:

Class 2 $\quad x_i = \mu w_1(i) + (1-\mu) w_3(i) + \varepsilon_i, \quad i = 1,...,21$

Class 3 $\quad x_i = \mu w_2(i) + (1-\mu) w_3(i) + \varepsilon_i, \quad i = 1,...,21$

Another version of the recognition task is to add the waveform data with an additional 19 noise variables. Hence, the input dimension is increased to 40, with the last 19 elements being normally distributed random numbers with mean 0 and variance 1. An analytic expression can be derived for the Bayes rule to yield an error rate of 0.14 (Breiman *et al.*, 1984). The two waveform generation programmes are obtained from the UCI repository of machine learning databases and domain theories (Murphy and Aha, 1995).

The performance of CART and the Nearest Neighbour (NN) classifiers are assessed with off-line experiments, namely using a training set of 300 samples and a test set of 5000 samples. In our simulations, we employ the on-line learning approach (as described in section 3.1) with 5000 samples. The samples are generated with equal prior class

probabilities. Likewise, a window of 1000-samples is used to calculate the on-line classification accuracy. Both FAM and PFAM operate in conservative, fast-learning mode with the baseline vigilance parameter, ρ_a, set to 0.25 and the overlapping parameter of PFAM set to 5.0. The input vectors are fuzzified for PFAM and are scaled between 0 and 1 and complement-coded for FAM.

4.2.1 Results

(Table 2 Comparison of performance with NN and CART)

The misclassification rates of CART and NN classifiers (Breiman *et al.*, 1984, pp. 168-171), as well as the on-line results (average of 5 runs) of FAM and PFAM at the end of the 5000-samples presentation, are presented in Table 2. Although the performance of the NN classifier is good with waveform data, the addition of noise significantly deteriorates its classification ability. The performance of the other three systems is little affected by the presence of noise. By using a higher value of ρ_a, FAM was able to achieve the same results as that of CART. Overall, PFAM yielded the best approximation to the Bayes limits in both waveform and waveform-plus-noise simulations. Furthermore, it should be noted that the number of committed nodes (prototypes) created by both FAM and PFAM was less than that used in training both NN and CART, i.e. 300 samples. In the waveform simulations, there were an average of 74 and 31 committed F_{2a} nodes in PFAM and FAM; while in the waveform-plus-noise simulations, the average number of F_{2a} nodes were 102 and 29 respectively.

(Figures 9 and 10 Results of the waveform and waveform-plus-noise simulations)

Figures 9 and 10 compare the performance of FAM and PFAM. The average accuracies and number of F_{2a} nodes resulted from 5 runs were plotted against increasing number of input samples, along with some indications of the standard deviation shown as error bars. The performance of FAM and PFAM is quite consistent with or without added noise. In general, PFAM demonstrated significant improvements in classification accuracy compared to FAM and achieved a closer proximity to the Bayes limits for the two data sets. Again, FAM created less number of F_{2a} nodes than PFAM for equal value of the vigilance parameter. Raising the vigilance parameter of FAM from 0.1 to 0.66 (waveform) and from 0.1 to 0.56 (waveform-plus-noise) resulted in an increase in the number of committed nodes to a number close to that for PFAM, as shown in Figures 9 and 10. Nevertheless, the improvements in accuracy were only 4% and 5% respectively at the end of the simulations. In both cases, FAM performance fell below that of PFAM by 10% and 12% respectively.

5 Summary and Conclusions

A promising approach is proposed which exploits both the advantages of neural networks and those of fuzzy logic systems is to integrate them onto a unified platform, such that the low-level learning ability of neural networks and the high-level, human-like expression and reasoning ability of fuzzy logic systems can be thoroughly incorporated. In fact, this is implemented in the FAM architecture in which a synthesis of neural information processing and fuzzy set theory is realised. However, in the statistical classification domain, the prediction of FAM could not provide an estimate of probabilities of the target categories. Therefore, the PNN paradigm is utilised to compensate for the deterministic characteristics of the predictions from FAM.

In view of the suitability of the learning methodologies and the similarity of network connections, we have integrated both FAM and the PNN to form a new hybrid network for on-line applications. The PFAM network incorporates the benefits of FAM and the PNN and, at the same time, alleviates the drawbacks of both the networks. It offers an autonomous learning system for general classification and probability estimation tasks. The autonomy leads the network continuously to achieve better performance with time, analogous to human learning behaviour in natural environments. The advantages of PFAM lie in its autonomous learning and probability estimation abilities. On one hand, the network is able to adapt to changes in the environment by self-growing and self-organising the incoming patterns sequentially. It, therefore, eliminates (i) the need to specify a static network size *a priori* since it is capable of establishing a good network configuration incrementally, and (ii) the re-training problem when the environment is non-stationary. On the other hand, PFAM is also capable of adaptively formulating complex non-linear decision boundaries that asymptotically approach the Bayes optimal on-line, in non-stationary environments. In addition, the probabilistic predictions from PFAM enable the application of the Bayes decision criterion to select the maximum *a posteriori* decision or to implement some risk-weighted classifications in order to minimise the overall misclassification overhead.

To optimise the performance of PFAM, several modifications have been proposed to integrate both FAM and the PNN efficiently and to enhance the generalisation capability of the hybrid system. These include procedures for centre and width estimation. The Kohonen learning algorithm is employed to fine-tune the prototypes for them to home-in to the centroids of the category clusters formed by the FAM algorithm. The width estimation procedure controls the overlapping of kernel functions using the nearest sample of other classes in order to preserve local neighbourhoods of distinct clusters. In addition, fuzzification of input patterns is proposed to allow fuzzy representation of features, thus exploiting the full potential of FAM. Then, the defuzzification process allows PFAM to maintain dimensionality in probability estimation using the Parzen-window approach.

To evaluate the capabilities of PFAM, several benchmark classification tasks in high-dimensional, noisy environments have been conducted.. The first experiment assesses the performance of PFAM using two densely overlapping Gaussian sources. Comparing performances of various networks, PFAM produces a better result in handling this task, as tabulated in Table 1. However, it should be noted that PFAM learns on-line with only one pass through the data and requiring little in the way of architectural design. In addition, according to the Parzen-window theorem, the estimated probability functions would more precisely converge to the actual underlying function when a larger sample size is provided. PFAM empirically exhibits this property where its performance is able to approach the Bayes optimal classification rates as the prototype samples increase.

In the second experiment, the waveform simulations posed an evaluation in a high dimensional, noisy environment. The results obtained are comparable to other approaches like the nearest neighbour and tree classifiers, although requiring less computation and lower network complexity. As expected, the increase in dimension in the waveform-plus-noise simulations by adding noise to the waveform data results in a larger network configuration. Nevertheless, the network still maintains performance levels even under noisy conditions.

From the simulation studies, convincing results have been obtained. These results indicate empirically that the hybrid structure of PFAM is suitable for on-line classification and probability estimation tasks. Owing to PFAM's autonomously learning capability, the network is able to learn the incoming patterns incrementally and continuously, rather than by being trained on a fixed set of samples, thus it might be expected to approach theoretical classification limits as time goes on.

References

Asfour, R.Y., Carpenter, G.A., Grossberg, S., Lesher, G. W. (1993a). Fusion ARTMAP: A Neural Network Architecture for Multi-channel Data Fusion and Classification. *Proc. of World Congress on Neural Networks*, Vol. II, pp 210-215.

Asfour, R.Y., Carpenter, G.A., Grossberg, S., Lesher, G. W. (1993b). Fusion ARTMAP: An Adaptive Fuzzy Network for Multi-channel Classification. *Proc. of the third Int. Conf. on Industrial Fuzzy Control and Intelligent Systems*, pp 155-160.

Bounds, D.G., Lloyd, P.J., Mathew, B.G. (1990), A Comparison of Neural Network and Other Pattern Recognition Approaches to the Diagnosis of Low Back Disorders. *Neural Networks*, **3**(5), pp 583-592.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. (1984). *Classification and Regression Trees*. Belmont: Wadsworth Inc.

Burrascano, P. (1991). Learning Vector Quantization for the Probabilistic Neural Network. *IEEE Trans. on Neural Networks*, **2**(4), pp 458-461.

Cacoullos, T. (1966). Estimation of a Multivariate Density. *Annals of the Institute of Statistical Mathematics* (Tokyo), **18**(2), pp 179-189.

Carpenter, G.A., Grossberg, S. (1987a). A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine. *Computer Vision, Graphics and Image Processing*, **37**, pp 54-115.

Carpenter, G.A., Grossberg, S. (1987b). ART 2: Stable Self-Organizing of Pattern Recognition Codes for Analog Input Patterns. *Applied Optics*, **26**, pp 4919-4930.

Carpenter, G.A., Grossberg, S. (1988). The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network. *IEEE Computer*, **21**, pp 77-88.

Carpenter, G.A., Grossberg, S. (1990). ART 3: Hierarchical Search using Chemical Transmitters in Self-Organizing Pattern Recognition Architectures. *Neural Networks*, **3**, pp 129-152.

Carpenter, G.A., Grossberg, S., Rosen, D.B. (1991a). Fuzzy ART : Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System. *Neural Networks*, **4**, pp 759-771.

Carpenter, G.A., Grossberg, S., Reynolds, J.H. (1991b). ARTMAP: Supervised Real-time Learning and Classification of Nonstationary Data by a Self-Organizing Neural Network. *Neural Networks*, **4**, pp 565-588.

Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., Rosen, D.B. (1992) Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps. *IEEE Trans. on Neural Networks*, **3**(5), pp 698-712.

Carpenter, G.A., Grossberg, S., Reynolds, J.H. (1993a). Fuzzy ARTMAP, Slow Learning and Probability Estimation. *Proc. of World Congress on Neural Networks*, **II**, pp 26-30.

Carpenter, G.A., Ross, W.D. (1993b). ART-EMAP: A Neural Network Architecture for Learning and Prediction by Evidence Accumulation. *Proc. of World Congress on Neural Networks*, **III**, pp 649-656.

Chen, S., Mulgrew, B., Grant, P.M. (1993a) A Clustering Technique for Digital Communications Channel Equalization using Radial Basis Function Networks. *IEEE Trans. on Neural Networks*, **4**(4), pp 570-590.

Chen, Y. Q., Thomas, D.W., Nixon, M.S. (1993b). Generating-Shrinking Algorithm for Learning Arbitrary Classification. *Neural Networks*, **7**(9), pp 1477-1489.

Cybenko, G. (1989). Approximation by Superposition of a Sigmoidal Function. *Mathematics of Control, Signals and Systems*, **2**, pp 303-314.

Duda, R.O., Hart, P.E. (1973). *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons.

Fritzke, B. (1994). Growing Cell Structures—A Self-organising Network for Unsupervised and Supervised Learning. *Neural Networks*, **7**(9), pp 1441-1460.

Fu, K.S. (1968). *Sequential Methods in Pattern Recognition and Machine Learning*. New York: Academic Press

Fujita, O. (1992). Optimization of the Hidden Unit Function in Feedforward Neural Networks. *Neural Networks*, **5**, pp 755-764.

Girosi, F., Poggio, T. (1990). Networks and the Best Approximation Property. *Biological Cybernetics*, **63**, pp 169-176.

Gorman, R.P., Sejnowski, T.J. (1988). Learning Classification of Sonar Targets using a massively parallel network. *IEEE Trans. on Acoustics, Speech and Signal Processing*, **36**(7), pp 1135-1140.

Hartigan, J.A. (1975). *Clustering Algorithm.* New York: John Wiley & Sons.

Kadirkamanathan, V., Niranjan, M. (1993). A Function Approach to Sequential Learning with Neural Networks. *Neural Computation*, **5**(6), pp 954-975.

Kohonen, T., Barna, G., Chrisley, R. (1988). Statistical Pattern Recognition with Neural Networks: Benchmarking Studies. *Proc. IEEE Int. Conf. on Neural Networks*, **I**, pp. I61-I68.

Kohonen, T. (1989). *Self-organization and Associative Memory (3rd. Edition).* Berlin: Springer-Verlag.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems.* Englewood Cliffs: Prentice-Hall Inc.

Lim, C.P., Harrison, R.F. (1994). Modified Fuzzy ARTMAP Approaches Bayes Optimal Classification Rates: An Empirical Demonstration. To appear in *Neural Network.*

Lim, C.P., Harrison, R.F. (1995). Minimal Error Rate Classification in a Non-stationary Environment via a Modified Fuzzy ARTMAP network. *Proc. of Int. Conf. on Artificial Neural Networks and Genetic Algorithms*, pp. 503-506.

Lin, C.T. (1994). *Neural Fuzzy Control Systems with Structure and Parameter Learning.* Singapore: World Scientific.

Melsa, J.L., Cohn, D.L. (1978). *Decision and Estimation Theory.* Tokyo: McGraw-Hill Kogakusha Ltd.

Moore, B. (1988). ART1 and Pattern Clustering. *Proc. of 1988 Connectionist Models Summer School*, pp 174-185.

Murphy, P.M., Aha, D.W. (1995). UCI Repository of Machine Learning Databases, [Machine-readable Data Repository]. Irvine, CA: University of California, Department of Information and Computer Science.

Musavi, M.T., Ahmed, W., Chan, K.H., Faris, K.B., Hummels. D.M. (1992a). On the Training of Radial Basis Function Classifiers. *Neural Networks*, **5**, pp 595-603.

Musavi, M.T., Kalantri, K., Ahmed, W., (1992b). Improving the Performance of Probabilistic Neural Networks. *Proc. IEEE Int. Conf. Neural Networks*, **I**, pp 595-600.

Musavi, M.T., Kalantri, K., Ahmed, W., Chan, K.H. (1993). A Minimum Error Neural Network (MNN). *Neural Networks*, **6**, pp 397-407.

Nie, J.H., Linkens, D.A. (1992). Neural Network-based Approximate Reasoning: Principles and Implementation. *Int. Journal of Control*, **56**(2), pp 399-413.

Parzen, E. (1962). On Estimation of a Probability Density Function and Mode. *Annals of Mathematical Statistics*. **33**, pp 1065-1076.

Richard, M.D., Lippmann, R.P. (1991). Neural Network Classifiers Estimate Bayesian *a posteriori* Probabilities. *Neural Computation*, **3**, pp 461-483.

Sejnowski, T.J., Rosenberg, C.R. (1987). Parallel Networks that Learn to Pronounce English Text. *Complex Systems*, **1**, pp 145-168.

Shadafan, R.S., Niranjan, M. (1994). A Dynamic Neural Network Architecture by Sequential Partitioning of the Input Space. *Neural Computation*, **6**(6), pp 1202-1223.

Sharkey, N. E. and Sharkey, A.J.C. (1994). Understanding Catastrophic Interference in Neural Nets. *Research Report-CS-94-4*, Department of Computer Science, University of Sheffield, UK.

Silverman, B.W. (1986). *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall.

Specht, D.F. (1988). Probabilistic Neural Networks for Classification, Mapping or Associative Memory. *Proc. IEEE Int. Conf. Neural Networks*, **I**, pp 525-532.

Specht, D.F. (1990). Probabilistic Neural Networks. *Neural Networks*, **3**, pp 109-118.

Specht, D.F. (1992). Enhancements to Probabilistic Neural Networks. *Proc. IEEE Int. Conf. Neural Networks*, **I**, pp 761-768.

Wan, E.A. (1990). Neural Network Classification: A Bayesian Interpretation. *IEEE Trans. on Neural Networks*, **1**(4), pp 303-305.

Wang, L.X. (1994). *Adaptive Fuzzy Systems and Control*. Englewood Cliffs: Prentice Hall.

Wasserman, P.D. (1993). *Advanced Methods in Neural Computing*. New York: Van Nostrand Reinhold.

White, H. (1989). Learning in Artificial Neural Networks: A Statistical Perspective. *Neural Computation*, **1**, pp 425-464.

Zadeh, L. (1965). Fuzzy Sets. *Information and Control*, **8**, pp 338-353.

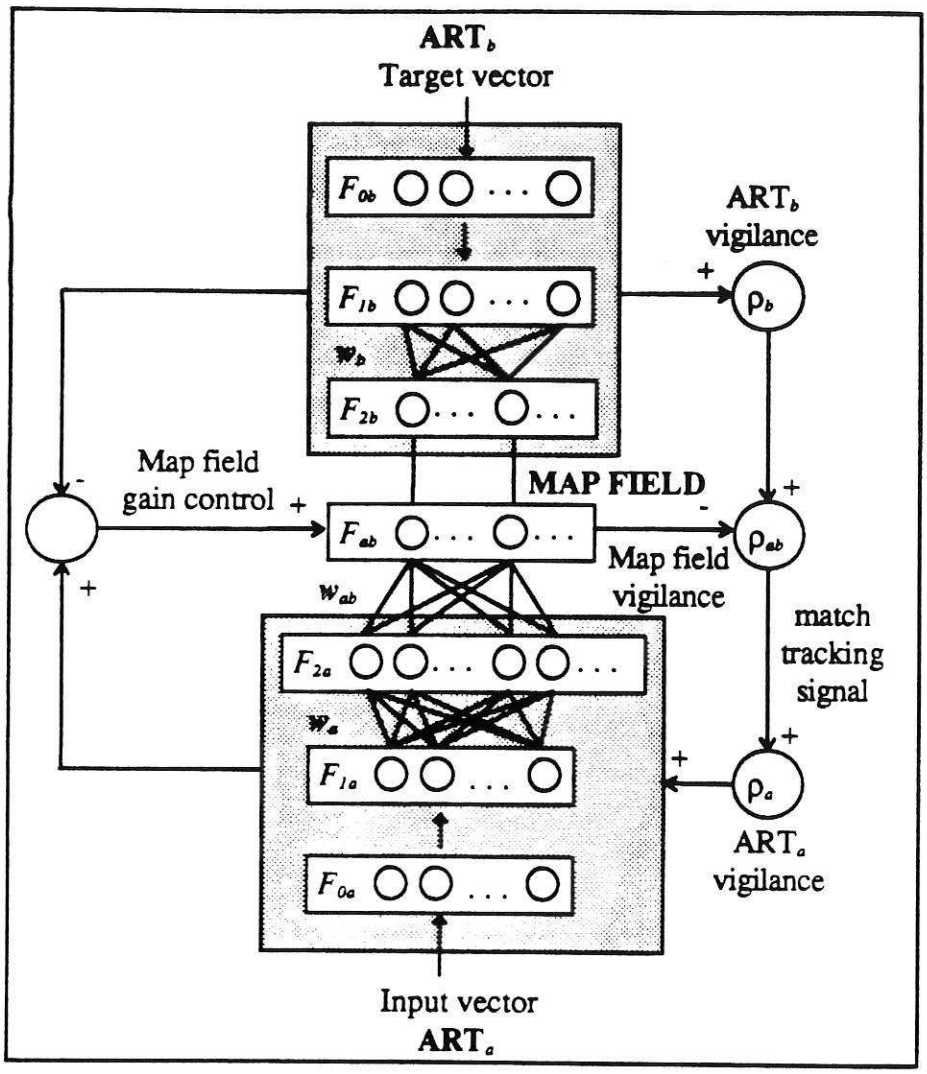

Figure 1 A schematic diagram of the Fuzzy ARTMAP (FAM) network.

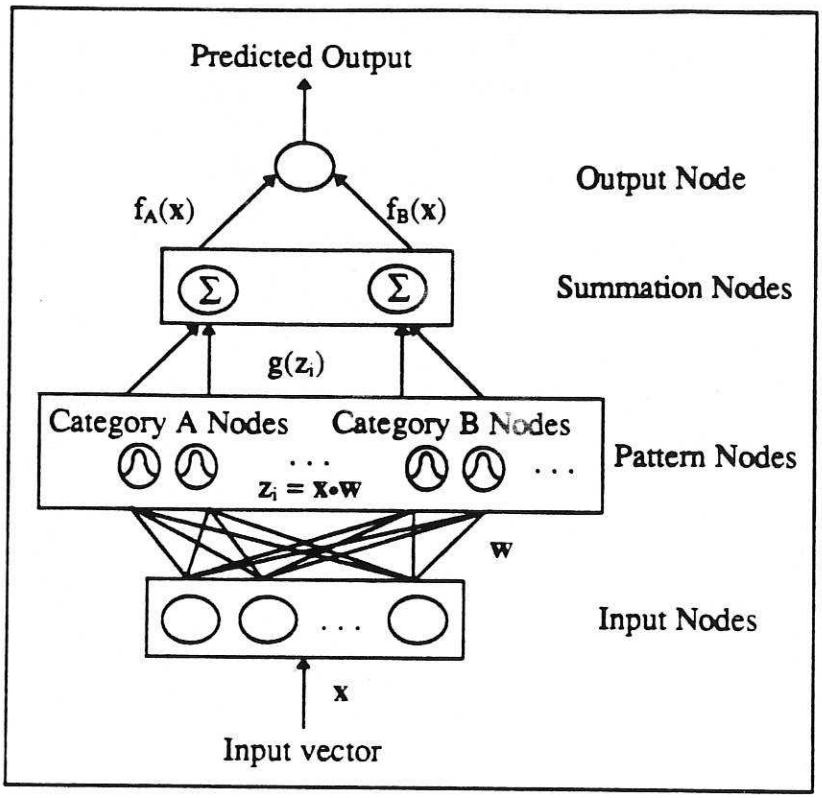

Figure 2 A schematic diagram of the Probabilistic Neural Network (PNN).

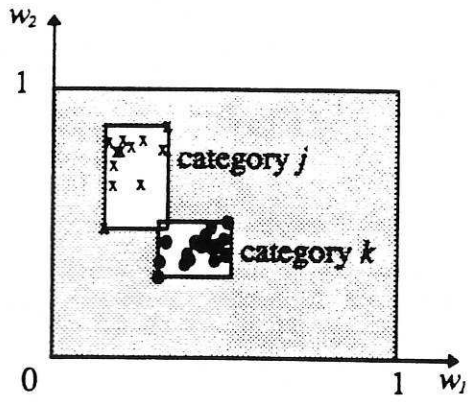

Figure 3 With fast learning and complement coding, the rectangles enclose all the samples that have activated the corresponding categories without reset, and the weight vectors are encoded by the vertices of the rectangles.

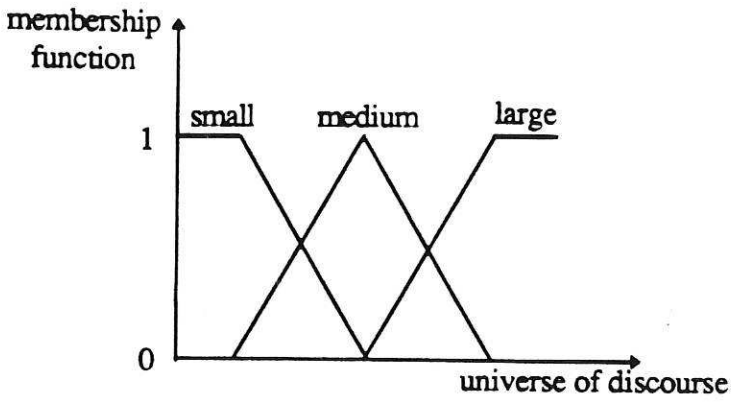

Figure 4 Membership functions for three fuzzy sets, namely "small", "medium" and "large".

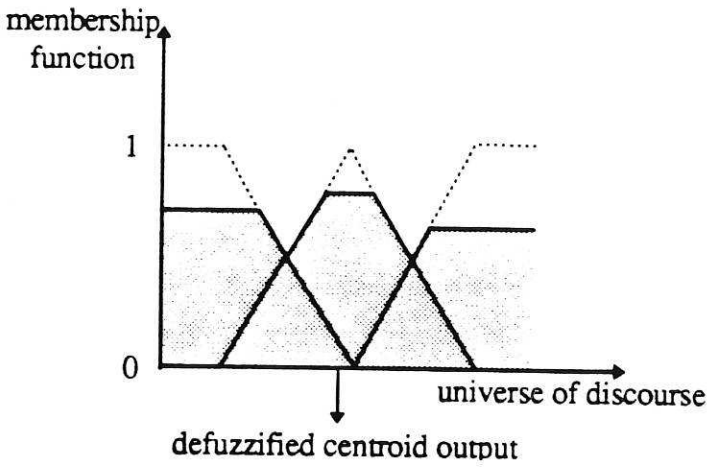

Figure 5 Defuzzification combines the membership functions to yield a centroid output in the universe of discourse.

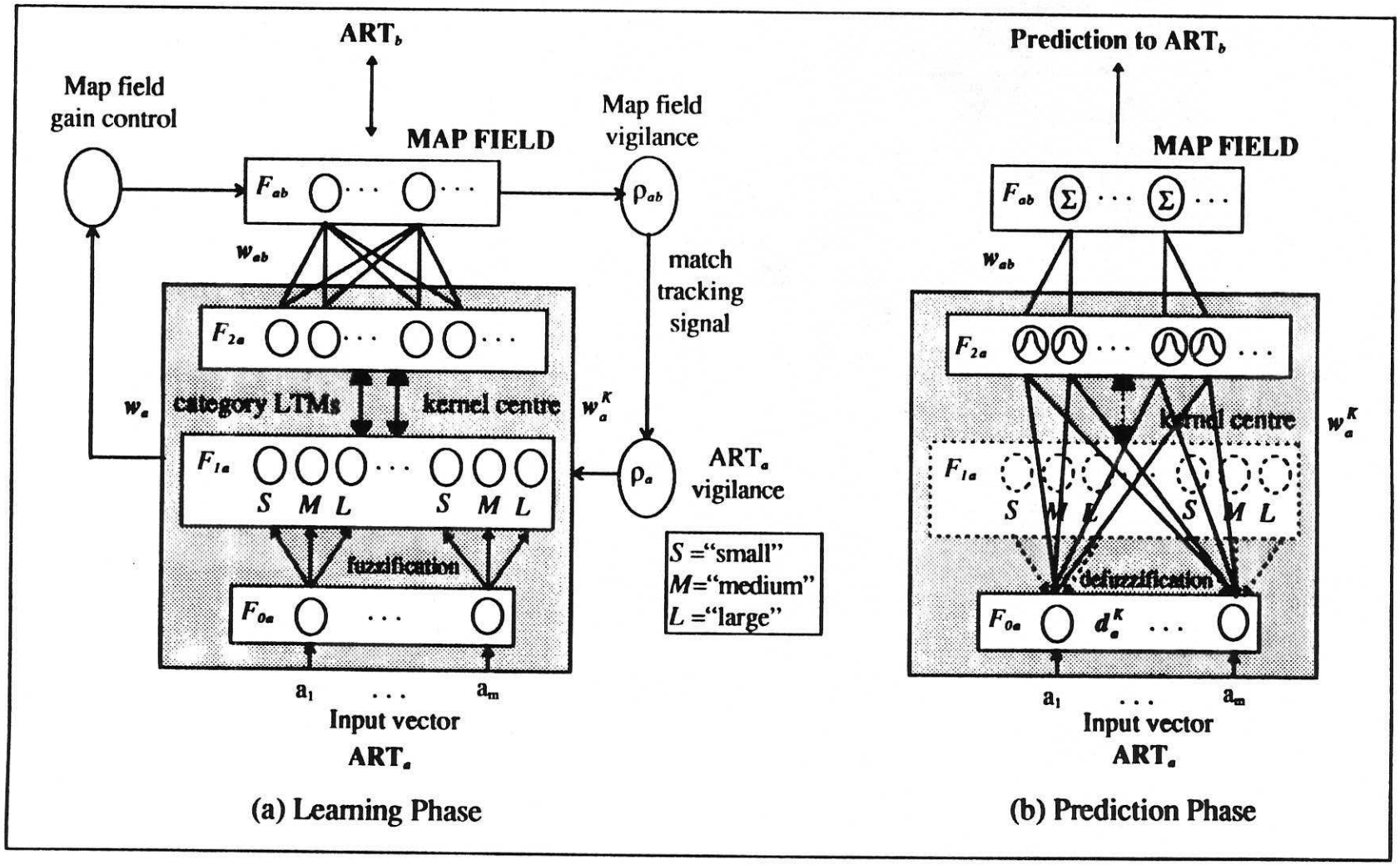

Figure 6 A schematic diagram of the Probabilistic Fuzzy ARTMAP (PFAM) network. During the learning phase, an m-dimensional input vector is fuzzified into a $3m$-dimensional membership vector for classification. During the prediction phase, the kernel centres are defuzzified to re-install into m-dimensional prototype vectors for probability density function estimation.

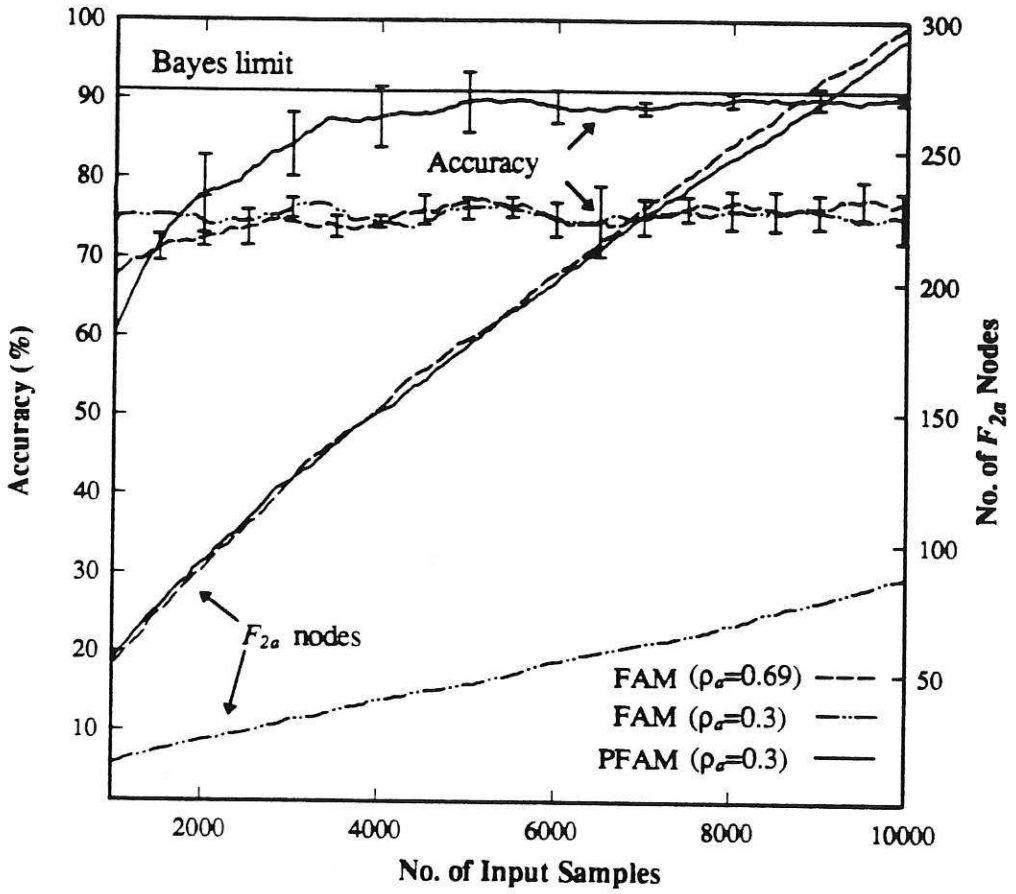

Figure 7 On-line classification results (with error bars) and growth pattern of F_{2a} nodes (without error bars) of the Gaussian source separation experiment. Increasing ρ_a from 0.3 to 0.69 causes an equivalent number of committed nodes in FAM and PFAM, but the improvement of FAM result is insignificant.

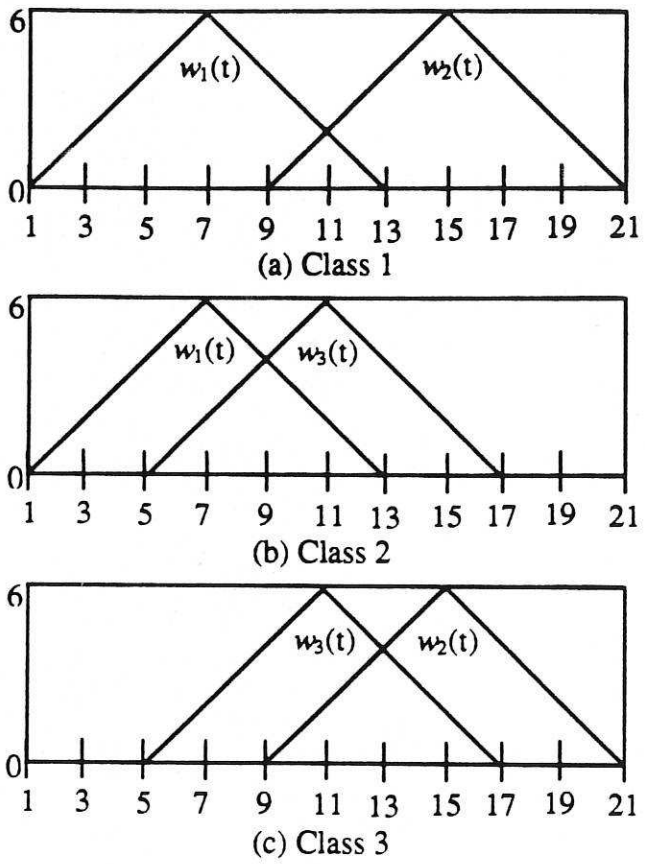

Figure 8 Three classes of waveform, each class is a combination of two types.

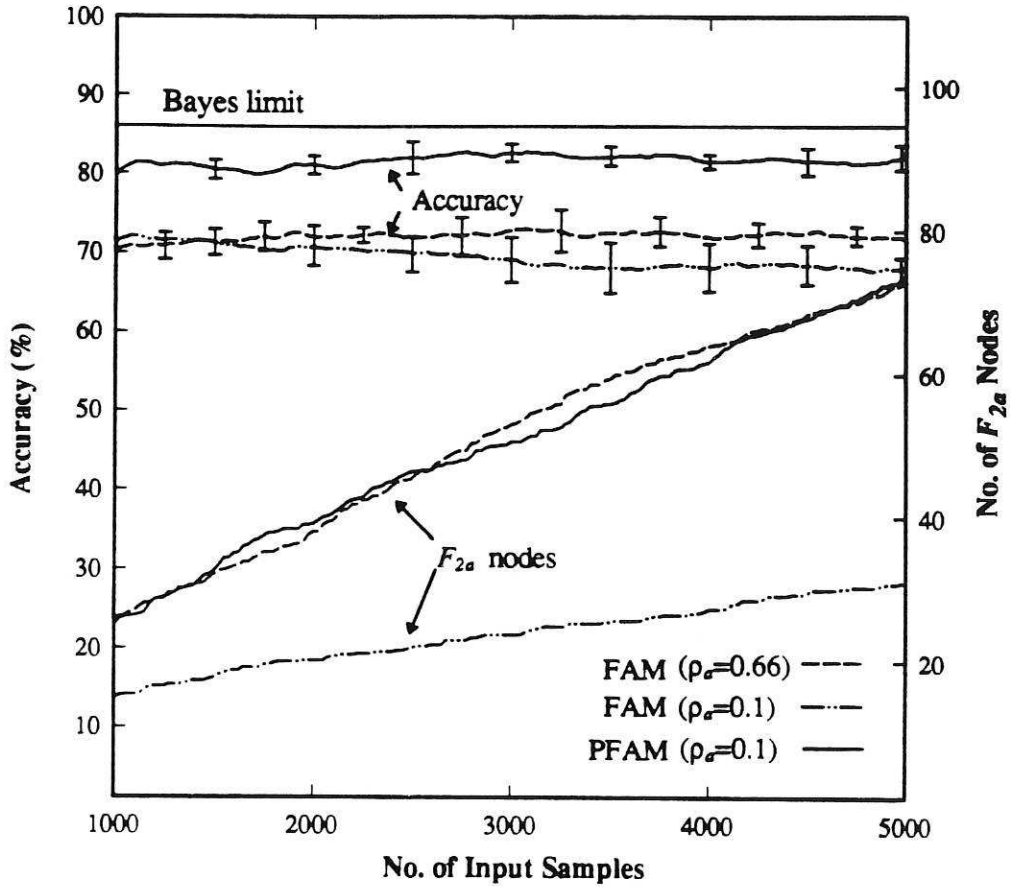

Figure 9 On-line classification results (with error bars) and growth pattern of F_{2a} nodes (without error bars) of the waveform experiment. FAM shows an improvement of 4% in its performance when ρ_a is raised from 0.1 to 0.66, but this is still 10% less accurate than PFAM.

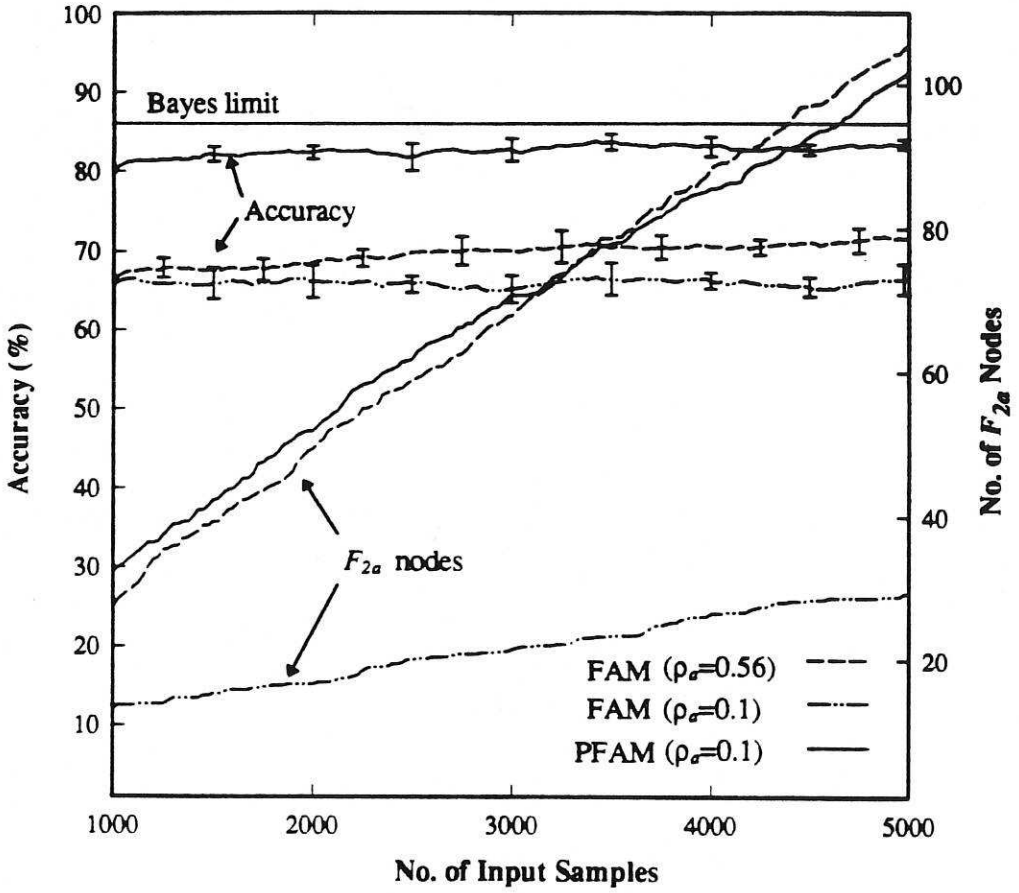

Figure 10 On-line classification results (with error bars) and growth pattern of F_{2a} nodes (without error bars) of the waveform-plus-noise experiment. FAM shows an improvement of 5% in its performance when ρ_a is raised from 0.1 to 0.56, but this is still 12% less accurate than PFAM.

Table 1 Comparison of performance of various networks in terms of misclassification rates expressed in percentages

Bayes Error Rate	BP	BM2 (Binary)	LVQ	MNN	PNN	FAM	PFAM
9.0	18.9	9.4	13.4	13	25.69 / 22 (different σ)	24.9 ($\rho_a = 0.3$) 22.8 ($\rho_a = 0.69$)	10

Table 2 Comparison of performance of various classifiers in terms of misclassification rates expressed in percentages

Data Set	Bayes Error Rate	NN	CART	FAM	PFAM
Waveform	14	22	28	32 ($\rho_a = 0.1$) 28 ($\rho_a = 0.66$)	18
Waveform-plus-noise	14	62	28	33 ($\rho_a = 0.1$) 28 ($\rho_a = 0.56$)	16