A Comparison of Fuzzy ARTMAP and Gaussian ARTMAP Neural Networks for Incremental Learning

Eric Granger, Jean-François Connolly and Robert Sabourin

Abstract— **Automatic pattern classifiers that allow for incremental learning can adapt internal class models efficiently in response to new information, without having to retrain from the start using all the cumulative training data. In this paper, the performance of two such classifiers – the fuzzy ARTMAP and Gaussian ARTMAP neural networks – are characterize and compared for supervised incremental learning in environments where class distributions are fixed. Their potential for incremental learning of new blocks of training data, after previously been trained, is assessed in terms of generalization error and resource requirements, for several synthetic pattern recognition problems. The advantages and drawbacks of these architectures are discussed for incremental learning with different data block sizes and data set structures. Overall results indicate that Gaussian ARTMAP is the more suitable for incremental learning as it usually provides an error rate that is comparable to that of batch learning for the data sets, and for a wide range of training block sizes. The better performance is a result of the representation of categories as Gaussian distributions, and of using category-specific learning rate that decreases during the training process. With all the data sets, the error rate obtained by training through incremental learning is usually significantly higher than through batch learning for fuzzy ARTMAP. Training fuzzy ARTMAP and Gaussian ARTMAP through incremental learning often requires fewer training epochs to converge, and leads to more compact networks.**

I. Introduction

FOR a wide range of applications, machine learning represents a cost-effective and practical approach to the design of pattern classification systems. However, the performance of pattern classifiers depends heavily on the availability of representative training data, and the acquisition (collection and analysis) of such data is expensive and time consuming in many practical applications. Data presented to a pattern classifier, during either the training or operational phases, may therefore be incomplete in one of several ways. In static environments, where class distributions remain fixed, these include a limited number of training observations, missing components of the input observations, missing class labels during training, and missing classes (*i.e.,* some classes that were not present in the training data set may be encountered during operations) [8]. In addition, new information, such as input components and output classes, and drifting classes, may suddenly emerge in dynamically-changing environments, where class distributions vary in time.

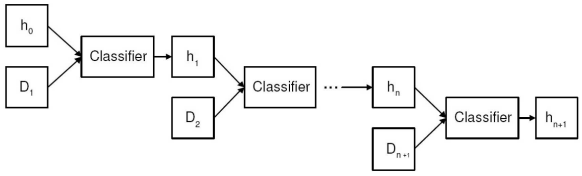


Fig. 1. A generic incremental learning scenario where blocks of data are used to update the classifier in an incremental fashion over a period of time. Let $D_1, D_2, ..., D_{n+1}$ be the blocks of training data available to the classifier at discrete instants in time $t_1, t_2, ..., t_{n+1}$. The classifier starts with initial hypothesis h_0 which constitutes the prior knowledge of the domain. Thus, h_0 gets updated to h_1 on the basis of D_1, and h_1 gets updated to h_2 on the basis of data D_2, and so forth [5].

Given a static environment in which training data is incomplete, a critical feature of future automatic classification systems designed according to the machine learning approach is the ability to update their class models incrementally during operational phases, in order to adapt to novelty encountered in the environment [5] [10]. Ideally, as new information becomes available, internal class models should be refined, and new ones should be created on the fly, without having to retrain from the start using all the cumulative training data.

For instance, in many practical applications, it is common to acquire additional training data from the environment at some point in time after the classification system has originally been trained and deployed for operations (see Figure 1). Assume that this data is characterized and labeled by a domain expert, and may contain observations belonging to classes that are not present in previous training data, and classes may have a wide range of distributions. It may be too costly or not feasible to accumulate and store all the data used thus far for supervised training, and to retrain a classifier using all the cumulative data[1]. In this case, it may only be feasible to update the system through *supervised incremental learning*.

Assuming that new training data becomes available, incremental learning provides the means to efficiently maintain an accurate and up-to-date class models. Another advantage of incremental learning is the low computational complexity required to update a classifier. Indeed, temporary storage of

The authors are with the Laboratoire d'imagerie, de vision et d'intelligence artificielle (LIVIA), École de technologie supérieure, 1100 Notre-Dame Ouest, Montreal, Qc., H3C 1K3, Canada, email: eric.granger@etsmtl.ca, jfconnolly@livia.etsmtl.ca, robert.sabourin@etsmtl.ca.

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada.

[1]To learn new data, the vast majority of classification algorithms proposed in literature must accumulate and store all training data in memory, and retrain from the start using all previously-accumulated training data.

978-1-4244-1821-3/08/$25.00©2008 IEEE

the new data is only required during training, and training is only performed with the new data. In addition, incremental learning may provide a powerful tool in human-centric applications, where a domain expert is called upon to propose new data set to gradually design and update a classification system as a complex environment unfolds. Strategies adopted for incremental learning will depend on the application – the nature of training data, the environment, performance constraints, etc. Regardless of the context, updating a pattern classification system in an incremental fashion raises several technical issues. In particular, accommodating new training data may corrupt the classifier's previously-acquired knowledge structure, and compromise its ability to achieve a high level of generalization during future operations. The *stability-plasticity dilemma* [1] refers to the problem of learning new information incrementally, yet overcoming the problem of catastrophic forgetting.

This paper focuses on techniques that are suitable supervised incremental learning in static environments, where class characteristics do not vary over time. According to Polikar [12], an incremental learning algorithm should:

1) allow to learn additional information from new data,
2) not require access to the previous training data,
3) preserve previously acquired knowledge, and
4) accommodate new classes that may be introduced with new data.

Property (3) is particularly relevant for providing high quality of results. In literature, some promising pattern classification algorithms have been reported for supervised incremental learning in static environments. For example, the ARTMAP [2] and Growing Self-Organizing [7] families of neural network classifiers, have been designed with the inherent ability to perform supervised incremental learning. Popular models from these families include the fuzzy ARTMAP, distributed ARTMAP PFAM, Gaussian ARTMAP, ellipsoid ARTMAP,Growing Cell Structure Growing Neural Gas and Growing Grid neural networks. These classifiers are inspired by the unsupervised learning paradigm used in self-organizing neural networks (SONNs), such as the Adaptive Resonance Theory (ART) [1] and Self-Organizing Feature Mapping (SOFM) [11] neural networks.

In contrast, some well-known pattern classifiers, such as the Support Vector Machine (SVM) [15], and the Multi-Layer Perceptron (MLP) [13] and Radial Basis Function (RBF) [6] neural networks have been adapted to perform supervised incremental learning under various assumptions of data sources, and of properties for the learning algorithm. These include the Incremental SVMs, Resource Allocation Network (RAN), and its variants (ERAN, RAN-LTM, RANEKF, MRAN) the Growing and Pruning (GAP) RBF; Parameter Incremental Learning, Incremental Back-Propagation and MLP, and Constructive networks neural networks. Finally, some high-level architectures, based on well-known pattern classifiers, *e.g.*, Ensemble of Classifiers, have also been proposed.

In this paper, the performance of fuzzy ARTMAP [4] and Gaussian ARTMAP [16] neural networks are characterize and compared for supervised incremental learning of new data in static environments. To the authors' knowledge, their incremental learning capabilities are not assessed in literature. Their potential for incremental learning of new blocks of training data is assessed empirically in terms of generalisation error during operations and resource requirements. An experimental protocol has been defined such that the impact on performance of learning a new block of training data incrementally, after each network has previously been trained, is assessed for different types of synthetic pattern recognition problems. The first type of problem consists of data with overlapping class distributions, whereas the second type involves data with complex decision boundaries but no overlap. The advantages and drawbacks of these architectures are discussed for incremental learning data using different data block sizes, and using different data set structures (overlap, dispersion, etc.).

In the next section, the fuzzy ARTMAP and Gaussian ARTMAP networks are briefly reviewed, with an emphasis on properties which make each one suitable to perform supervised incremental learning in static environments, where class distributions are fixed. Then, the experimental protocol, performance measures and synthetic data sets, used for proof-of-concept computer simulations are described in Section 3. Finally, experimental results are presented and discussed in Section 4.

II. ARTMAP Neural Networks

ARTMAP refers to a family of neural network architectures based on Adaptive Resonance Theory (ART) [1] that is capable of fast, stable, on-line, unsupervised or supervised, incremental learning, classification, and prediction [2]. It combines an ART unsupervised neural network [1] with a map field.

A key feature of ARTMAP networks is their unique solution to the stability-plasticity dilemma. The plasticity of ART-based networks has been well documented. During training, ART networks adjusts previously-learned categories in response to familiar inputs, and creates new categories dynamically in response to inputs different enough from those previously seen. A vigilance test allows to regulate the maximum tolerable difference between any two input patterns in a same category. Low vigilance leads to broad generalization and abstract memories, while high vigilance leads to narrow generalization and detailed memories. The map field links category nodes to output class nodes. Finally, a match tracking process resets a category node linked to a class node that produces an incorrect class prediction.

As mentioned, ARTMAP neural networks have been designed with the ability to perform supervised incremental learning as defined in [12]. In supervised learning mode, the sequential learning process grows the number of F_2 category nodes progressively, according to a problem's complexity. The vigilance and match tracking process provide the mechanisms to control the local impact of new data on the existing knowledge structure.

The original ARTMAP [2] architecture can process binary-valued input patterns by employing ART1 [1] as the ART network. The popular fuzzy ARTMAP [4] then integrated the fuzzy ART [3] to process both analog and binary-valued input patterns. Since then, several ARTMAP networks have been proposed in order to improve the performance of these architectures. Members of the ARTMAP family can be broadly divided according to their internal matching process, which depends on either deterministic or probabilistic category activation. The deterministic type consists of networks such as fuzzy ARTMAP, ART-EMAP, ARTMAP-IC, default ARTMAP, simplified ARTMAP, distributed ARTMAP, etc., and represent each class using one or more category hyper-rectangles. In contrast, the probabilistic type consists of networks such as PROBART, PFAM, MLANS, Gaussian ARTMAP, ellipsoid ARTMAP, boosted ARTMAP, μARTMAP, etc., and represent each class using one or more probability density functions.

This paper will focus on two popular networks – the fuzzy ARTMAP and Gaussian ARTMAP. The rest of this section highlights the main differences between these two networks. (Refer to Table I for a summary of algorithmic differences.)

A. Fuzzy ARTMAP:

The fuzzy ART neural network consists of two fully connected layers of nodes: an M node input layer, F_1, and an N node competitive layer, F_2. A set of real-valued weights $\mathbf{W} = \{w_{ij} \in [0,1] : i = 1,2,...,M;\ j = 1,2,...,N\}$ is associated with the F_1-to-F_2 layer connections. Each F_2 node j represents a recognition category that learns a prototype vector $\mathbf{w}_j = (w_{1j}, w_{2j}, ..., w_{Mj})$. The F_2 layer of fuzzy ART is connected, through learned associative links, to an L node map field F^{ab}, where L is the number of classes in the output space. A set of binary weights $\mathbf{W}^{ab} = \{w^{ab}_{jk} \in \{0,1\} : j = 1,2,...,N;\ k = 1,2,...,L\}$ is associated with the F_2-to-F^{ab} connections. The vector $\mathbf{w}^{ab}_j = (w^{ab}_{j1}, w^{ab}_{j2}, ..., w^{ab}_{jL})$ links F_2 node j to one of the L output classes.

In supervised training mode, ARTMAP classifiers learn an arbitrary mapping between training set patterns $\mathbf{a} = (a_1, a_2, ..., a_m)$ and their corresponding binary supervision patterns $\mathbf{t} = (t_1, t_2, ..., t_L)$. These patterns are coded to have unit value $t_K = 1$ if K is the target class label for $\mathbf{a}$, and zero elsewhere. The following algorithm describes fuzzy ARTMAP learning:

1. Initialisation: Initially, all the F_2 nodes are uncommitted, all weight values w_{ij} are initialized to 1, and all weight values w^{ab}_{jk} are set to 0. An F_2 node becomes committed when it is selected to code an input vector $\mathbf{a}$, and is then linked to an F^{ab} node. Values of the learning rate β, choice α, match tracking ε, and baseline vigilance $\bar{\rho}$ parameters are set.

2 Input pattern coding: When a training pair $(\mathbf{a}, \mathbf{t})$ is presented to the network, $\mathbf{a}$ undergoes a transformation called complement coding, which doubles its number of components. The complement-coded input pattern has $M = 2m$ dimensions and is defined by $\mathbf{A} = (\mathbf{a}, \mathbf{a}^c) = (a_1, a_2, ..., a_m; a^c_1, a^c_2, ..., a^c_m)$, where $a^c_i = (1 - a_i)$, and $a_i \in [0,1]$. The vigilance parameter ρ is reset to its baseline value $\bar{\rho}$.

3. Prototype selection: Pattern $\mathbf{A}$ activates layer F_1 and is propagated through weighted connections $\mathbf{W}$ to layer F_2. Activation of each node j in the F_2 layer is determined by the *Weber law choice function* T_j. The F_2 layer produces a binary, winner-take-all pattern of activity $\mathbf{y} = (y_1, y_2, ..., y_N)$ such that only the node $j = J$ with the greatest activation value $J = \arg\max\{T_j : j = 1,2,...,N\}$ remains active; thus $y_J = 1$ and $y_j = 0, j \neq J$. Node J propagates its top-down expectation back onto F_1 and the *vigilance test* is performed. This test compares the degree of match between $\mathbf{w}_J$ and $\mathbf{A}$ against the dimensionless *vigilance parameter*. If the test is passed, then node J remains active and resonance is said to occur. Otherwise, the network inhibits the active F_2 node and searches for another node J that passes the vigilance test. If such a node does not exist, an uncommitted F_2 node becomes active and undergoes learning (Step 5).

4. Class prediction: Pattern $\mathbf{t}$ is fed directly to the map field F^{ab}, while the F_2 category $\mathbf{y}$ learns to activate the map field via associative weights $\mathbf{W}^{ab}$. The F^{ab} layer produces a binary pattern of activity $\mathbf{y^{ab}} = (y^{ab}_1, y^{ab}_2, ..., y^{ab}_L) = \mathbf{t} \wedge \mathbf{w}^{ab}_J$ in which the most active F^{ab} node $K = \arg\max\{y^{ab}_k : k = 1,2,...,L\}$ yields the class prediction ($K = k(J)$). If node K constitutes an incorrect class prediction, then a *match tracking* signal raises the vigilance parameter ρ just enough to induce another search among F_2 nodes in Step 3. This search continues until either an uncommitted F_2 node becomes active (and learning directly ensues in Step 5), or a node J that has previously learned the correct class prediction K becomes active.

5. Learning: Learning input $\mathbf{a}$ involves updating prototype vector $\mathbf{w}_J$, and, if J corresponds to a newly-committed node, creating an associative link to F^{ab}. The prototype of F_2 node J, $\mathbf{w}_J$, is updated according to the learning rate parameter β. With complement coding and fast learning ($\beta = 1$), fuzzy ARTMAP represents category j as an m-dimensional hyper-rectangle R_j that is just large enough to enclose the cluster of training set patterns $\mathbf{a}$ to which it has been assigned. A new association between F_2 node J and F^{ab} node K ($k(J) = K$) is learned by setting $w^{ab}_{Jk} = 1$ for $k = K$, where K is the target class label for $\mathbf{a}$, and 0 otherwise. The next training subset pair $(\mathbf{a}, \mathbf{t})$ is presented to the network in Step 2

Supervised training ends in accordance with some learning strategy, following one or more training epochs[2]. Once the weights $\mathbf{W}$ and $\mathbf{W}^{ab}$ have been found through this process, ARTMAP can predict a class label for an input pattern by performing Steps 2, 3 and 4 without any vigilance or match tests. During testing, a pattern $\mathbf{a}$ that activates node J is predicted to belong to class $K = k(J)$.

B. Gaussian ARTMAP:

The Gaussian ARTMAP has a similar overall neural network architecture and training process as fuzzy ARTMAP, yet differs significantly in a few respects. First, it represents category j as a Gaussian density function, defined by two

[2]An epoch is defined as one complete presentation of all the patterns of a finite training data set.

TABLE I

EQUATIONS USED BY THE ARTMAP NETWORKS. WITH FUZZY ARTMAP, $|\cdot|$ IS THE L^1 NORM OPERATOR ($|\mathbf{w}_j| \equiv \sum_{i=1}^{M} |w_{ji}|$), $\wedge$ IS THE FUZZY AND OPERATOR ($(\mathbf{A} \wedge \mathbf{w}_j)_i \equiv \min(A_i, w_{ji})$), AND α, β, ε AND $\bar{\rho}$ ARE THE CHOICE, LEARNING RATE, MATCH TRACKING AND BASELINE VIGILANCE PARAMETERS, RESPECTIVELY. WITH GAUSSIAN ARTMAP, γ IS THE INITIAL STANDARD DEVIATION ASSIGNED TO NEWLY-COMMITTED F_2 NODES.

Algorithmic step	**ARTMAP neural network**	
	fuzzy ARTMAP	Gaussian ARTMAP
1. Initialization:	$\alpha > 0$, $\beta \in [0,1]$, $0 < \varepsilon \ll 1$, $\bar{\rho} \in [0,1]$	$\gamma > 0$, $0 < \varepsilon \ll 1$, $\bar{\rho} \in [0,1]$
2. Input pattern coding:	$\mathbf{A} = (\mathbf{a}, \mathbf{a}^c)$ $(M = 2m)$	$\mathbf{A} = \mathbf{a}$ $(M = m)$
3. Prototype selection:		
– choice function	$T_j(\mathbf{A}) = \lvert\mathbf{A} \wedge \mathbf{w}_j\rvert / (\alpha + \lvert\mathbf{w}_j\rvert)$	$g_j(\mathbf{A}) = \begin{cases} \frac{n_j}{\prod_{i=1}^{M} \sigma_{ji}} G_j(\mathbf{A}) & \text{if } G_j(\mathbf{A}) > \rho \\ 0 & \text{otherwise} \end{cases}$
– vigilance test	$\lvert\mathbf{A} \wedge \mathbf{w}_J\rvert \geq \rho m$	$G_j(\mathbf{A}) = \exp\left\{ -\frac{1}{2} \sum_{i=1}^{M} \frac{(A_i - \mu_{ji})^2}{\sigma_{ji}^2} \right\} > \rho$
– F_2 activation	$y_j = 1$ only if $j = J$ (winning node)	$y_j = g_j / (0.01 + \sum_{l=1}^{N_c} g_l)$
4. Class prediction:		
– prediction function	$S_k^{ab}(\mathbf{y}) = \sum_{j=1}^{N} y_j w_{jk}^{ab}$	$S_k^{ab}(\mathbf{y}) = \sum_{j=1}^{N_c} y_j w_{jk}^{ab}$
– match tracking	$\rho' = (\lvert\mathbf{A} \wedge \mathbf{w}_J\rvert / m) + \varepsilon$	$\rho' = \exp\left\{ -\frac{1}{2} \sum_{j \in E_K} y_j^* \sum_{i=1}^{M} \frac{(A_i - \mu_{ji})^2}{\sigma_{ji}^2} \right\} + \varepsilon$
5. Learning:		
– prototype update	$\mathbf{w}'_J = \beta(\mathbf{A} \wedge \mathbf{w}_J) + (1 - \beta)\mathbf{w}_J$	$n'_j = n_j + y_j^*$ $\mu'_{ji} = (1 - \frac{y_j^*}{n_j})\mu_{ji} + \frac{y_j^*}{n_j} A_i$ $\sigma'_{ji} = \sqrt{(1 - \frac{y_j^*}{n_j})\sigma_{ji}^2 + \frac{y_j^*}{n_j}(A_i - \mu_{ji})^2}$

vectors: its mean $\mu_j = (\mu_{j1}, \mu_{j2}, ..., \mu_{jM})$ and its standard deviation $\sigma_j = (\sigma_{j1}, \sigma_{j2}, ..., \sigma_{jM})$ (together they replace prototype vector $\mathbf{w}_j$). A scalar, n_j, accumulates the amount of relative activation obtained by F_2 node j on training set patterns. The relative activation is employed to reduce the learning rate of categories during the training process.

During training, the number of committed F_2 nodes, N_c, is initially set to 0. Newly-committed F_2 nodes increment N_c, and undergo the initialization step: setting $\mu_J = \mathbf{A}$, $\sigma_{Ji} = \gamma$, $w_{JK}^{ab} = 1$ and $n_J = 1$. Committed F_2 nodes that pass the vigilance test for pattern $\mathbf{a}$ are allowed to activate, and distribute a pattern of activity $\mathbf{y} = (y_1, y_2, ..., y_{N_c})$. Match tracking and learning are performed according to the relative activation over the "ensemble" E_K of F_2 nodes linked to the predicted F^{ab} node K. The relative activation over E_K is defined by the distributed pattern $\mathbf{y}^* = (y_1^*, y_2^*, ..., y_{N_c}^*)$, where $y_j^* = y_j / \sum_{l \in E_K} y_l$ only if $j \in E_K$, and $y_j^* = 0$ otherwise.

III. EXPERIMENTAL METHODOLOGY

In order to observe the impact on performance of training fuzzy ARTMAP and Gaussian ARTMAP with supervised incremental learning for different data structures, several data sets were selected for computer simulations. The synthetic data sets are representative of pattern recognition problems that involve either (1) simple decision boundaries with overlapping class distributions, or (2) complex decision boundaries, where class distributions do not overlap on decision boundaries. The synthetic data sets correspond to 2 or 3 class problems, with a 2 dimensional input feature space. Each data subset is composed of an equal number of 5,000 patterns per class, for a total of 20,000 (2 classes) or 30,000 (3 classes) randomly-generated patterns.

Prior to a simulation trial, each data set is normalized according to the min-max technique, and partitioned into two equal parts – the learning and test subsets. The learning subset is again divided into training and validation subsets. They respectively contain 2/3 and 1/3 of patterns from the learning subset. In order to perform block-wise hold-out validation, the training and validation subsets are divided into either b blocks. Each block D_i (for $i = 1, 2, ..., b$) contains an equal number of patterns per class. To observe the impact on performance of learning new blocks of training data incrementally, using different data block sizes, two different scenarios are observed. The first scenario consists in training with $b = 10$ larger blocks containing 1000 patterns, while the second one consists in training with $b = 100$ small blocks containing 100 patterns.

During each simulation trial, the fuzzy ARTMAP and Gaussian ARTMAP are trained using a batch learning and incremental learning process. For *batch learning*, the number of blocks D_i used for training is progressively increased from 1 to 10 (when $|D_i| = 1000$) or from 1 to 100 (when $|D_i| = 100$). For the first trial, performance is assessed after initializing an ARTMAP network and training it on B_1, Then it is assessed after initializing another ARTMAP network and training from the start on $B_1 \cup B_2$, and so on, until all b blocks are learned. For *incremental learning*, the ARTMAP networks are trained until convergence, on successive blocks of data D_i when $|D_i| = 100$ and when $|D_i| = 1000$). At first, performance is assessed after initializing an ARTMAP network and training on B_1. Then it is assessed after training the *same* ARTMAP network incrementally on B_2, and so on, until all b blocks are learned.

For each trial, training ends after the epoch for which

the generalization error is minimized on the corresponding validation subset. Learning is performed using a hold-out validation technique, with network training halted for validation after each epoch. The performance of fuzzy ARTMAP and Gaussian ARTMAP was measured when using standard parameter settings that yield minimum network resources (internal categories, epochs, etc.). Fuzzy ARTMAP hyper-parameters are set to $\beta = 1$, $\alpha = 0.001$, $\bar{\rho} = 0$ and $\varepsilon = 0.001$, whereas Gaussian ARTMAP hyper-parameters are set to $\gamma = 0.1$ and $\bar{\rho} = 0$.

Since ARTMAP performance is sensitive to the presentation order of the training data, the pattern presentation orders were always randomized from one epoch to the next. In addition, each simulation trial was repeated 10 times with 10 different randomly generated data sets. The average performance of fuzzy and Gaussian ARTMAP was assessed in terms of resources requirements and generalisation error. The amount of resources is measured by compression and convergence time. *Compression* refers to the average number of training patterns per category prototype created in the F_2 layer. *Convergence time* is the number of epochs required to complete learning for a training strategy. It does not include presentations of the validation subset used to perform hold-out validation. *Generalisation error* is estimated as the ratio of incorrectly classified test subset patterns over all test set patterns. The combination of compression and convergence time provides useful insight into the amount of processing required by fuzzy ARTMAP during training to produce its best asymptotic generalisation error. Average results, with corresponding standard error, are always obtained, as a result of the 10 independent simulation trials.

Of the six synthetic data sets selected for simulations, three have simple decision boundaries with overlapping class distributions, $D_{2N}(\varepsilon_{tot})$, $D_{3N}(\varepsilon_{tot})$, and $D_{XOR}(\varepsilon_{tot})$, and three have complex decision boundaries without overlap, D_{CIS}, $D_{XOR\text{-}u}$ and D_{P2}. It is assumed that data from the classes in D_{2N}, D_{3N} and D_{XOR} is randomly generated by sources with the same Gaussian noise, and that the total theoretical probability of error associates with these problems is denoted by ε_{tot}. Data from the classes in D_{CIS}, $D_{XOR\text{-}u}$ and D_{P2} are generated according to a uniform distribution, and the total theoretical probability of error for these data sets is 0, since class distributions do not overlap on decision boundaries.

The $D_{2N}(\varepsilon_{tot})$ data consists of two classes, each one defined by a normal distribution in a two dimensional input feature space [9]. Both sources are described by variables that are independent and have equal variance σ^2, therefore distributions are hyperspherical. In fact, $D_{2N}(\varepsilon_{tot})$ refers to 3 data sets, where the degree of overlap, and thus the total probability of error between classes differs for each set. The degree of overlap is varied from a total probability of error ε_{tot} = 1%, 13%, and 25%, by adjusting the variance of both classes to σ^2 = 1.0, 4.3, and 11.9, respectively. With the $D_{3N}(\varepsilon_{tot})$ problem, data from three classes are randomly-generated by 3 normal distributions with mean values arranged according to an equilateral triangle, with a distance of 5.12 from one center to another. With the $D_{XOR\text{-}u}$ problem, data is generated by 2 classes according to bi-modal distributions. The four normal distributions are centered in the 4 squares of a classical XOR problem. The degree of overlap is varied from a total probability of error, ε_{tot} = 1%, 13%, and 25%, by adjusting the variance of both classes to σ^2 = 1.0, 2.3, and 3.3, respectively.

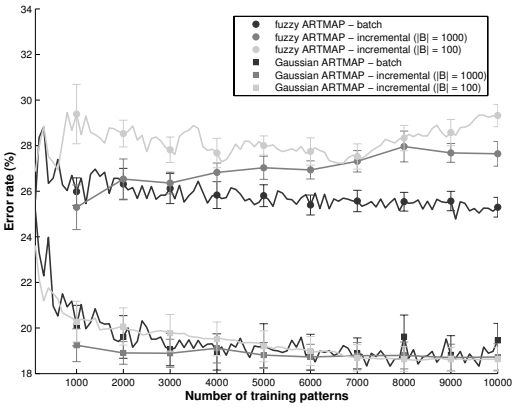
The Circle-in-Square problem D_{CIS} requires a classifier to identify the points of a square that lie inside a circle, and those that lie outside a circle [2]. The circle's area equals half of the square. It consists of one non-linear decision boundary where classes do not overlap. With the $D_{XOR\text{-}u}$ data, the 'on' and 'off' classes of the classical XOR problem are divided by a horizontal decision bound at $y = 0.5$, and a vertical decision bound at $x = 0.5$. Finally, with the D_{P2} problem, each decision region of its 2 classes is delimited by one or more of its four polynomial and trigonometric functions, and belongs to one of the two classes [14]. It consists of four non-linear boundaries, and class definitions do not overlap.

IV. Simulation Results

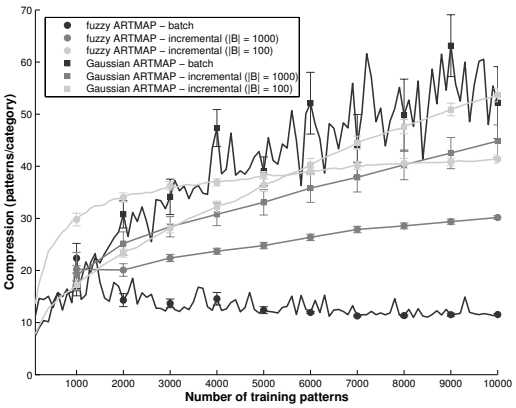
Figure 2 presents the average performance achieved as a function of the training subset size, when fuzzy ARTMAP and Gaussian ARTMAP are trained using batch and incremental learning on the $D_{XOR}(13\%)$ data set. For incremental learning, block sizes of 100 and 1000 are employed. Very similar tendencies are found in simulation results with other data sets with class distributions overlap and decision boundaries are linear.

As shown in Figure 2(a), the error rate obtained by training fuzzy ARTMAP through incremental learning is generally significantly higher than that obtained through batch learning. Using the smaller block size ($|D_i| = 100$) yields a higher error rate that with the larger block size ($|D_i| = 1000$). In addition, error tends to grow with the number of blocks having been learned. For example, after the fuzzy ARTMAP network undergoes incremental learning of 100 blocks with $|D_i| = 100$, the average error is about 29.3%, yet after learning 10 blocks with $|D_i| = 1000$, the error is about 27,6%. As expected, since the $D_{XOR}(\xi_{tot})$ data is sampled from normal distributions, Gaussian ARTMAP achieves a lower generalisation error than fuzzy ARTMAP. However, the average generalisation error obtained by performing incremental learning with Gaussian ARTMAP is comparable to that obtained through batch learning. This holds true for the smaller and larger block sizes. For example, whether the Gaussian ARTMAP network is trained with incremental or batch learning, the average error rate after all the training data has been learned is about 18.6%.

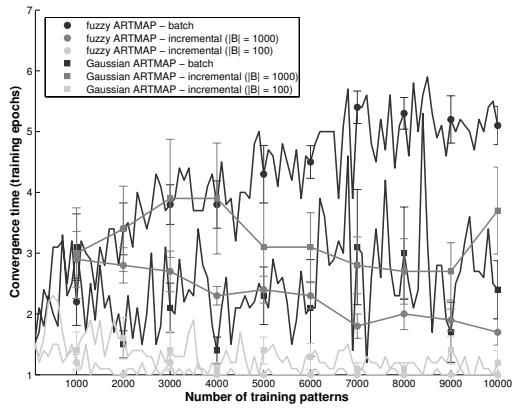
Although the error is greater, Figure 2(b) indicates that the compression obtained when fuzzy ARTMAP is trained through incremental learning is significantly higher than if trained through batch learning, and it tends to grow as the block size is decreased. Incremental learning also tends to reduce the number of training epochs required for fuzzy ARTMAP to converge (see Figure 2(c)). As the block size



(a) Generalisation error.

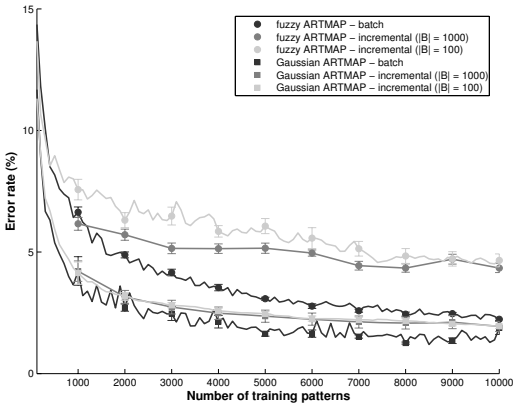


(b) Compression.

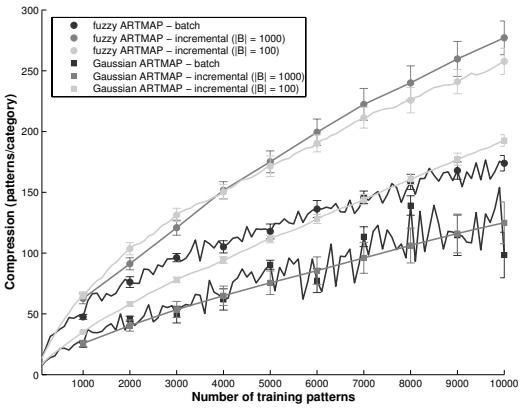


(c) Convergence time.

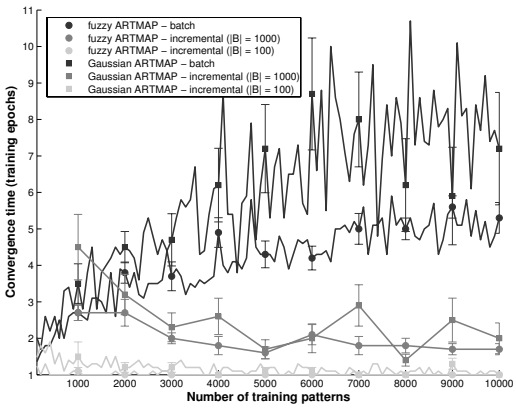
Fig. 2. Average performance of fuzzy ARTMAP and Gaussian ARTMAP versus training subset size for $D_{\mathrm{XOR}}(13\%)$. Both networks are trained using batch and incremental (with $|D_i| = 100$ and $|D_i| = 1000$) learning. Error bars are standard error of the sample mean.



(a) Generalisation error.



(b) Compression.



(c) Convergence time.

Fig. 3. Average performance of fuzzy ARTMAP and Gaussian ARTMAP versus training subset size for D_{CIS}. Both networks are trained using batch and incremental (with $|D_i| = 100$ and $|D_i| = 1000$) learning. Error bars are standard error of the sample mean.

decreases, the convergence time tends towards 1. For example, after fuzzy ARTMAP undergoes incremental learning of 100 blocks with $|D_i| = 100$, the average compression and convergence time are about 40 patterns/category and 1.0 epoch, respectively. After learning 10 blocks with $|D_i| = 1000$, the compression and convergence time are about 30 patterns/category and 1.8 epochs. This compares favorably to fuzzy ARTMAP trained through batch learning, where the compression and convergence time are about 10 patterns/category and 5.0 epochs. In this case, the performance of fuzzy ARTMAP as the training set size grows is indicative of overtraining [9]. With Gaussian ARTMAP, results indicate that the compression and convergence time obtained through incremental learning with are comparable to that of batch learning. Incremental learning provides a slightly lower convergence time with $|D_i| = 100$.

Figure 3 presents the average performance achieved as a function of the training subset size, when fuzzy ARTMAP and Gaussian ARTMAP are trained using batch and incremental learning on the D_{CIS} data set. Here again, block sizes of 100 and 1000 are employed for incremental learning. Very similar tendencies are found in simulation results for other data set where complex boundaries and class distributions that do not overlap.

As shown in Figure 3(a), when the training set size increases, the average generalisation error of fuzzy ARTMAP and Gaussian ARTMAP trained with either batch or incremental learning decreases asymptotically towards its minimum. However, the generalisation error obtained by training fuzzy ARTMAP through incremental learning is generally significantly higher than that obtained through batch learning. As with the data that has overlapping class distributions, the error tends to grow as the block size decreases. However, after the fuzzy ARTMAP network performs incremental learning of 100 blocks with $|D_i| = 100$, the average error is comparable to after learning 10 blocks with $|D_i| = 1000$ (about 4.5%). The generalisation error achieved by performing incremental learning with Gaussian ARTMAP is comparable to that obtained through batch learning, regardless of block size. For example, whether the Gaussian ARTMAP network is trained with incremental or batch learning, the average error rate after all the training data has been learned is about 1.9%.

As shown in Figure 3(b) and (c), training fuzzy ARTMAP through incremental learning yields a significantly higher compression than with batch learning. Furthermore, the convergence time associated with incremental learning is considerably lower than with batch learning. Results indicate that as the block size is decreased, the convergence time with incremental learning tends towards 1. For example, after fuzzy ARTMAP undergoes incremental learning of 100 blocks with $|D_i| = 100$, the average compression and convergence time are about 260 patterns/category and 1.0 epoch, respectively. After learning 10 blocks with $|D_i| = 1000$, the compression and convergence time are about 280 patterns/category and 1.8 epochs. With Gaussian ARTMAP, results indicate that the compression and convergence time obtained through incremental learning also tends to grow and the convergence time also tends to decrease with incremental learning. This is particularly true as the block size decreases. For example, after Gaussian ARTMAP undergoes incremental learning of 100 blocks with $|D_i| = 100$, the average compression and convergence time are about 175 patterns/category and 1.0 epoch, respectively.

Overall results indicate that when fuzzy ARTMAP undergoes incremental learning, the error rate tends to degrade as the block size decreases. Incremental learning consists in training on the data of each block in isolation, over one or more training epochs. The view of an environment is more uncertain or incomplete as the block size is reduced.

Since the first blocks form the basis for future updates, results underline the importance of initiating incremental learning with blocks that contain enough representative data from the environment. With overlapping data, the first block could be organized to grow classes from the inside towards the overlapping regions, through some active learning strategy. With complex boundaries, the first block could be organized to define the non-linear bourns between classes.

For all data sets, training fuzzy ARTMAP and Gaussian ARTMAP through incremental learning generally requires fewer training epochs to converge, and leads to more compact networks. Since the Gaussian ARTMAP error rate is often comparable for batch and incremental learning, results suggest that a cost-effective strategy for supervised training of a finite training set D consists in dividing the data into n blocks, and training Gaussian ARTMAP with successive blocks D_i for $i = 1, ..., n$, through incremental learning.

Table II shows the average generalisation error obtained with the fuzzy ARTMAP and Gaussian neural networks trained using batch and incremental learning on $D_{2\mathrm{N}}(\xi_{tot})$, $D_{\mathrm{XOR}}(\xi_{tot})$, D_{CIS} and D_{P2}. Training was performed on 5,000 patterns per class. For incremental learning, the networks were trained with 100 blocks of sizes $|D_i| = 100$ and 10 blocks of size $|D_i| = 1000$. Based on these results, Gaussian ARTMAP is the more suitable for incremental learning of new blocks in environments where class distributions are fixed. Overall, it usually performs comparably to the batch learning case for all the data sets, and for small and large block sizes. The better performance is due in part to the representation of categories as Gaussian distributions, and to the use of L^2 norms for pattern matching. Another asset is the category-specific learning rate that decreases during the training process, according to the training patterns assigned to a category. On the other hand, fuzzy ARTMAP pattern matching and is based on L^1 norms and category hyperrectangles. It also relies on a global learning rate that remains fixed over the training process.

V. Conclusions

In practical applications, classifiers found inside automatic pattern recognition systems may generalize poorly as they are designed prior to operations using limited training data. In response to new information, techniques for incremental

TABLE II

AVERAGE GENERALISATION ERROR OF FUZZY ARTMAP AND GAUSSIAN ARTMAP CLASSIFIERS TRAINED USING BATCH AND INCREMENTAL LEARNING ON ALL DATA FROM SYNTHETIC SETS. VALUES IN PARENTHESIS ARE STANDARD ERROR OF THE SAMPLE MEAN.

Data set	**Average generalisation error (%)**			
	fuzzy ARTMAP		**Gaussian ARTMAP**	
	batch	**incremental** $\lvert D_i\rvert = 1000 \rightarrow \lvert D_i\rvert = 100$	**batch**	**incremental** $\lvert D_i\rvert = 1000 \rightarrow \lvert D_i\rvert = 100$
$D_{2N}(1\%)$	2.1 (0.1)	3.2 (0.3) → 2.8 (0.2)	1.4 (0.1)	1.4 (0.1) → 1.4 (0.1)
$D_{2N}(13\%)$	21.6 (0.3)	23.8 (0.5) → 24.9 (0.6)	15.6 (0.5)	15.6 (0.6) → 15.4 (0.5)
$D_{2N}(25\%)$	35.8 (0.2)	37.8 (0.2) → 38.4 (0.2)	26.8 (0.2)	27.4 (0.2) → 27.3 (0.4)
$D_{3N}(1\%)$	2.4 (0.3)	3.0 (0.3) → 3.2 (0.4)	1.8 (0.6)	1.7 (0.3) → 1.5 (0.2)
$D_{XOR}(1\%)$	1.2 (0.1)	1.7 (0.2) → 1.7 (0.2)	2.6 (0.6)	3.7 (0.8) → 1.9 (0.6)
$D_{XOR}(13\%)$	25.3 (0.4)	27.6 (0.5) → 29.3 (0.5)	19.5 (0.7)	18.7 (0.6) → 18.6 (0.5)
$D_{XOR}(25\%)$	43.0 (0.3)	44.0 (0.3) → 44.0 (0.3)	35.7 (0.2)	37.4 (0.5) → 37.0 (0.3)
D_{CIS}	2.23 (0.05)	4.3 (0.2) → 4.7 (0.3)	1.9 (0.3)	1.9 (0.2) → 1.9 (0.1)
$D_{XOR\text{-}u}$	0.25 (0.07)	0.6 (0.1) → 0.9 (0.6)	1.4 (0.4)	1.9 (0.5) → 2.4 (0.6)
D_{P2}	5.11 (0.09)	7.9 (0.2) → 9.4 (0.4)	3.5 (0.3)	4.4 (0.3) → 5.9 (0.5)

learning would allow such classifiers to efficiently adapt class models during operational phases, without having to retrain from the start using all the training data, and without corrupting the previously-learned knowledge structure.

In this paper, the performance of fuzzy ARTMAP and Gaussian ARTMAP neural classifiers are characterize and compared for supervised incremental learning of new blocks of training data in static environments. Their potential for incremental learning is assessed thought a comprehensive set of computer simulations on synthetic data with overlapping distributions, and with complex decision boundaries but no overlap. The performance of the two networks is compared in terms of generalisation error and resource requirements, using different data block sizes.

With all the data sets used for simulations, the average error rate obtained by training fuzzy ARTMAP through incremental learning is usually significantly higher than that obtained through batch learning. This error tends to grow as the block size decreases. In contrast, the error obtained through incremental learning with Gaussian ARTMAP is comparable to that obtained through batch learning, regardless of the training block size. Results also indicate that training fuzzy ARTMAP and Gaussian ARTMAP through incremental learning often requires fewer training epochs to converge, and leads to more compact networks. As the block size decreases, the compression often increases, and the convergence time tends towards 1. Overall results indicate that Gaussian ARTMAP is the more suitable for incremental learning as it provides an error rate that is usually comparable to that of batch learning for all data sets, and over a wide range of training block sizes. The better performance is a result of the representation of categories as Gaussian distributions, and to category-specific learning rate that decreases during the training process.

The first blocks of training data form the basis for future updates, and incremental learning should be initiated with enough representative data from the environment. This is particularly relevant when fuzzy ARTMAP undergoes incremental learning, since the error rate tends to degrade as the block size decreases. Future work should involve assessing the impact on performance of the first block, and estimating lower bounds on the block size for different ARTMAP networks and data set structures.

REFERENCES

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

[2] Carpenter, G. A., Grossberg, S., and Reynolds, J. H., "ARTMAP: Supervised Real-Time Learning and Classification of Nonstationary Data by a SONN," *Neural Networks*, 4, 565-588, 1991.

[3] Carpenter, G. A., Grossberg, S., and Rosen, D. B., "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System", *Neural Networks*, **4:6**, 759-771, 1991.

[4] Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., Reynolds, and Rosen, D.B., "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps," *IEEE Trans. on Neural Networks*, **3:5**, 698-713, 1992.

[5] Caragea, D., Silvescu, A., and Honavar, V., "Towards a Theoretical Framework for Analysis and Synthesis of Agents That Learn from Distributed Dynamic Data Sources," In *Emerging Neural Architectures Based on Neuroscience*. Berlin: Springer-Verlag, 2001.

[6] Chen, S., Cowan, C. F. N., Grant, P. M., "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," *IEEE Trans. on Neural Networks*, **2:2**, 302-309, 1991.

[7] Fritzke, B., "Growing Self-Organizing Networks - Why?," *Proc. European Symposium on Artificial Intelligence*, 6172, 1996.

[8] Granger, E., Rubin, M. A., Grossberg, S., & Lavoie, P. "Classification of Incomplete Data Using the Fuzzy ARTMAP Neural Network," *Proc. Int'l Joint Conference on Neural Networks*, Vol. IV, 35-40, 2000.

[9] Granger, E., Henniges, P., Sabourin, R., and Oliveira, L. S., "Supervised Learning of Fuzzy ARTMAP Neural Networks Through Particle Swarm Optimization," *J. of Pattern Recognition Research*, **2:1**, 27-60, 2007.

[10] Kasabov, N., "Evolving Fuzzy Neural Networks for Supervised/Unsupervised Online Knowledge-Based Learning," *IEEE Trans. on Systems, Man, and Cybernetics*, **31:6**, 902-918, 2001.

[11] Kohonen, T., *Self-Organization and Associative Memory*, Springer-Verlag, 3rd ed., 1989.

[12] Polikar, R., Udpa, L., Udpa, S., and Honavar, V., "Learn++: An Incremental Learning Algorithm for MLP Networks," *IEEE Trans. Systems, Man, and Cybernetics*, **31:4**, 497-508, 2001.

[13] Rumelhart, D. E., Hinton, G. Williams, R. J., "Learning Internal Representations by Error Propagation," In Rumelhart D. E., and McClelland, J, L., eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1/318-362, MIT Press, 1986.

[14] Valentini, G., "An Experimental Bias-Variance Analysis of SVM Ensembles Based on Resampling Techniques," *IEEE Trans. Systems, Man, and Cybernetics - Part B: Cybernetics*, **35:6**, 1252-1271, 2005.

[15] Vapnik, V., *The Nature Stat. Learning Theory*, Springer-Verlag, 1995.

[16] Williamson, J. R., "A Constructive, Incremental-Learning Neural Network for Mixture Modeling and Classification," *Neural Computation*, **9:7**, 1517-1543, 1997