Optimization Of Network Parameters And Semi-supervision In Gaussian Art Architectures

2005

Roopa Chalasani
University of Central Florida

Find similar works at: http://stars.library.ucf.edu/etd

University of Central Florida Libraries http://library.ucf.edu

Part of the Electrical and Electronics Commons

STARS Citation

http://stars.library.ucf.edu/etd/294

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
OPTIMIZATION OF NETWORK PARAMETERS AND SEMI-SUPERVISION IN GAUSSIAN ART ARCHITECTURES

by

ROOPA CHALASANI

B.S Jawaharlal Nehru Technological University, India, 2002

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering in the Department of Electrical and Computer Engineering in the College of Engineering and Computer Science at the University of Central Florida
Orlando, Florida

Spring Term
2005
ABSTRACT

In this thesis we extensively experiment with two ART (adaptive resonance theory) architectures called Gaussian ARTMAP (GAM) and Distributed Gaussian ARTMAP (dGAM). Both of these classifiers have been successfully used in the past on a variety of applications. One of our contributions in this thesis is extensively experiments with the GAM and dGAM network parameters and appropriately identifying ranges for these parameters for which these architectures attain good performance (good classification performance and small network size). Furthermore, we have implemented novel modifications of these architectures, called semi-supervised GAM and dGAM architectures. Semi-supervision is a concept that has been used effectively before with the FAM and EAM architectures and in this thesis we are answering the question of whether semi-supervision has the same beneficial effect on the GAM architectures too. Finally, we compared the performance of GAM, dGAM, EAM, FAM and their semi-supervised versions on a number of datasets (simulated and real datasets). These experiments allowed us to draw appropriate conclusions regarding the comparative performance of these architectures.
Dedicated to my beloved Family
ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Michael Georgiopoulos for rendering his assistance and support throughout the preparation of my thesis. I would also like to thank my thesis committee, Dr. Michael Haralambous and Dr. Taskin Kasparis for their valuable advice and insights on this thesis. I would like to thank my family members, my fiancé Praveen Aeluri and all my roomies for their encouragement and support.
# TABLE OF CONTENTS

LIST OF FIGURES ............................................................................................................... iv
LIST OF TABLES ................................................................................................................... vii

1. INTRODUCTION ........................................................................................................... 1
   Organization of the thesis: ...................................................................................... 5

2. LITERATURE REVIEW ............................................................................................... 6

3. Description of the network algorithms used ............................................................. 12

4. FUZZY ARTMAP ...................................................................................................... 15
   Training Phase of FAM: ..................................................................................... 20
   Performance Phase of FAM: .............................................................................. 25

5. ELLIPSOIDAL ARTMAP .......................................................................................... 28
   Training Phase of EAM ...................................................................................... 32
   Performance Phase of EAM: .............................................................................. 37

6. GAUSSIAN ARTMAP (GAM) .................................................................................. 40
   Training Phase of GAM: ...................................................................................... 46
   Performance Phase of GAM: .............................................................................. 51
   Category Choice Function (CCF): ...................................................................... 53
   Category Match Function (CMF): ...................................................................... 55

7. DISTRIBUTED GAUSSIAN ARTMAP (dGAM) ...................................................... 57
   Performance Phase of dGAM: ............................................................................. 58

8. SEMI SUPERVISED GAM ......................................................................................... 60
LIST OF FIGURES

Figure 1  Simplified summary of the important ART networks ........................................ 11

Figure 2  Category regions of FAM, EAM and GAM respectively ................................. 12

Figure 3  Simplified block diagram of FAM ..................................................................... 16

Figure 4  Block diagram of FAM architecture .................................................................. 17

Figure 5  Simplified EAM architecture ............................................................................ 28

Figure 6: Gaussian distribution that fit the independently varying data and co-varying 
data. ........................................................................................................................................... 42

Figure 7: The figure showing the 2-dimensional 2-class Gaussian data ......................... 77

Figure 8: figure showing the 2-dimensional, 4-class Gaussian data ................................. 77

Figure 9: figure showing the 2-dimensional, 6-class Gaussian data ................................. 78

Figure 10: Graph showing the performances of GAM and dGAM for 2, 4 and 6, class 
problems in each case ........................................................................................................ 83

Figure 11: Graph showing the comparison of number of categories created in each case 
by GAM and dGAM ............................................................................................................. 84

Figure 12: Graph showing the comparison of average of the Best 100 results of GAM and 
dGAM in each case ............................................................................................................. 84

Figure 13: Graph showing the comparison of the average performance of all the networks 
created by GAM and dGAM ............................................................................................... 85
Figure 14: Graph shows the average PCC, std. PCC, average number of categories created and std NC of the networks for every rho value of the 2 class 15% overlap problem for the GAM and dGAM respectively. ......................................................... 86

Figure 15: Graph showing the performance of the 2 class 15% problem for each particular gamma for the GAM and dGAM respectively. ......................................................... 87

Figure 16: Percentage correct classification on test set for ssFAM, ssEAM, ssGAM and ssdGAM against epsilon on abalone database ................................................................. 95

Figure 17: Percentage correct classification on the cross-validation set and number of categories created by the ssGAM networks against Epsilon for abalone database. 95

Figure 18 Percentage correct classification on test set for ssFAM, ssEAM, ssGAM and ssdGAM against epsilon on pageblocks database ................................................................. 96

Figure 19 Percentage correct classification on test set for ssFAM, ssEAM, ssGAM, ssdGAM against epsilon on Iris database ................................................................. 97

Figure 20 Percentage of correct classification on the cross-validation set (PCC XV) versus percentage correct classification on the train set (PCC Train) for ssGAM on 5%, 25% and 40% overlapping data respectively. ......................................................... 99

Figure 21 Percentage correct classification on cross validation set versus epsilon for ssGAM and ssdGAM ............................................................................................................. 100

Figure 22: Percentage correct classification on training set versus percentage correct classification on cross-validation set for pageblocks and abalone database respectively .............................................................................................................. 101
Figure 23: Maximum and average percentage correct classification on cross-validation set against epsilon for the pageblocks and iris databases respectively.

102
LIST OF TABLES

Table 1: Comparison of GAM and dGAM for 2, 4 and 6 class problems with 5, 15, 25 and 40 % overlap .......................................................... 82

Table 2: Summary of the results of the generalized performance of the networks produced on the cross validation set .......................................................... 89

Table 3: Summary of the average performance of all the networks produced on the cross-validation set for every epsilon .............................................. 90

Table 4: Summary of results for the 2 Class Artificial databases ......................... 91

Table 5: Summary of results for the 4 Class Artificial databases ........................... 91

Table 6: Summary of results for the 6 Class Artificial databases ........................... 92

Table 7: Summary of results for Real databases .................................................. 94
1. INTRODUCTION

An important performance measure of a machine-learning algorithm is its generalization capability. Generalization is characterized by the number of unseen examples correctly predicted by a learning algorithm given sample-training data from which to learn. Desirable characteristics of learning systems are as follows

Parallel Computation: Use simple local operations, which are suitable for implementation in parallel hardware.

Fast Learning: Learn the mapping quickly and reliably from as few training samples as possible.

Efficient Representation: minimize the storage requirement of the internal representation while maximizing predictive accuracy.

Resistant to noise: System’s representation should remain efficient even if the data are noisy. Training samples often contain incorrect or inconsistent input-output pairings, due to either errors in the collection data, or to the intrinsic discriminative insufficiency of the data features.

The development of incremental supervised learning systems has included a promising line of research investigating ARTMAP neural network architectures.
The most prominent ARTMAP system for classifying analog data is Fuzzy ARTMAP, which has been shown to perform well in a number of benchmarks with respect to other learning systems (Carpenter et al., 1991a, 1992a, b). Fuzzy ARTMAP is a neural network architecture for conducting supervised learning in a multidimensional setting. When Fuzzy ARTMAP is used on a learning problem, it is trained to the point that it correctly classifies all training data. This feature causes Fuzzy ARTMAP to ‘over-fit’ some data sets, especially those in which the underlying pattern has to overlap. To avoid the problem of ‘over-fitting’ we must allow for error in the training process. One solution for allowing error during training is to use a statistical approach. Such a statistical approach is used in Gaussian ARTMAP and Boosted ARTMAP. Gaussian ARTMAP satisfies the above criterion for incremental learning systems better than Fuzzy ARTMAP because it produces a more efficient representation and is more resistant to noise.

In this thesis we focus on the examination of the Gaussian ARTMAP architecture. One aspect of our investigation is pointed towards the Gaussian ARTMAP’s network parameters on which the performance of the network depends. Usually, the performance of neural network architecture is measured by its generalization ability on unseen data and by the network size created to solve a specific problem. A second aspect of our investigation is in determining of whether the Gaussian ARTMAP architecture benefits from a relatively recent
concept introduced in the neural network literature, the concept of semi-supervised learning. This concept was introduced by Anagnostopoulos, et al., (2002), as well as by others. The purpose of semi-supervised learning is to create classifiers that conserve the property of stable learning, while achieving a non-zero post training error to avoid over training that quite often leads into loss of generalization performance (i.e., performance of the classifiers on unseen data), and in creating excessively large network sizes. Semi-supervised learning has been used successfully on the Fuzzy ARTMAP and Ellipsoidal ARTMAP architectures (Madan Bharadwaj in 2003). Finally, a third aspect of our investigation is to provide a comparison of a variety of ART architectures such as FAM, EAM, and GAM as well as their corresponding counterparts called, ssFAM, ssEAM, and ssGAM. The performance comparison relies on the generalization performance and network size created by these architectures on a variety of carefully chosen simulated and real datasets.

Semi-supervised learning (SSL) refers to the semi-supervised manner according to which exemplars are formed during training to identify clusters. According to the typical fully supervised learning scheme of classifier's, training patterns that are rendered to be pertinent to an exemplar by virtue of their position in the input feature domain can be associated with or can influence the structure of this exemplar only if both of them correspond to the same class label. Furthermore, training is considered incomplete if there is at least one exemplar that mis-
predicts the class label of a training pattern. Therefore, while in fully supervised learning mode, an exemplar is not allowed to commit any misclassification error. Eventually after the completion of the learning process, a typical EBC will feature a zero post-training error.
Organization of the thesis:

The thesis starts with the literature review. The actual topic of the thesis begins from the description of the operating phases of each of the networks considered here (FAM, EAM, GAM, dGAM, and sGAM, sdGAM). The detailed description of FAM, EAM, GAM and dGAM are each included in different chapters 4, 5, 6 and 7 respectively. The training and testing phase of each of these chapters are included in their respective chapters The description of the semi-supervised learning and its implementation are discussed in chapter 8. The datasets used for experimentation artificial databases and real databases are discussed in two different sub sections of chapter 9. Furthermore, experimental results discussing the three topics of interest with GAM, i.e., effect of network parameters, semi-supervised GAM performance, and comparisons of network performances are all included in chapter, 9 as sub-chapters. Finally, the summary and conclusions from our work are outlined in chapter 10.
ART was introduced by Grossberg as a means of describing how recognition categories are self-organized in neural networks. Since then, a number of specific neural network architectures based on ART have been proposed. Some of these architectures originated from Carpenter, Grossberg and their colleagues at Boston University, but other researchers in the field have contributed to the ART literature, as well. The first ART NN architecture, named ART1, appeared in the literature in 1987 (see Carpenter and Grossberg, 1987a). This model is an unsupervised neural network model capable of self-organizing (clustering) arbitrary collections of binary input patterns. Later, in 1987, the ART2 neural network architecture was introduced (see Carpenter and Grossberg, 1987b). This architecture is capable of clustering arbitrary collections of real-valued input patterns. The ART2 neural network became obsolete in 1991, when the simpler Fuzzy ART architecture was proposed (see Carpenter, Grossberg, and Rosen, 1991). Like ART2, Fuzzy ART is capable of clustering real-valued input patterns. In addition for binary-valued inputs, the operation of Fuzzy ART reduces to that of ART1. The ART1, ART2 and Fuzzy ART architectures perform unsupervised learning. In unsupervised learning (also called self-organization) training patterns of unknown classification are used, and there is no external teaching procedure. An internal teaching function determines how network parameters are adapted.
based upon the nature of the input patterns. In this case, the teaching procedure results in the internal categorization of training patterns according to some measure of similarity amongst the patterns. That is similar training patterns are grouped together during the training of the network. These groups (or clusters) are then considered the pattern classes into which unknown input patterns are later mapped to.

Supervised learning, on the other hand, requires a set of training patterns of known classification and an external teaching procedure. The teaching procedure is used to adapt network weights according to the network’s response to the training patterns. Normally, this adjustment is proportional to the amount of error present while attempting to classify the current input pattern. The use of supervised learning can be logically separated into two phases—a *training phase* and a *performance phase*. In the training phase, a training set is formed from representative samples taken from the environment in which the neural network is expected to operate. This training set should include sample patterns from all the pattern classes being categorized. Next, the training patterns are applied to the network inputs, and the external teacher modifies the system through the use of a training algorithm. Once acceptable results have been obtained from the training phase, the network may be used in the performance phase. In the performance phase, an unknown pattern is drawn from the environment in which the network operates and applied to the network inputs. At this point, the neural
network is expected to perform the recognition task for which it has been trained. If the neural network is able to correctly classify with a high probability input patterns that do not belong to the training set, then it is said that the neural network is able to generalize. Good generalization is one of the most sought after performance measures of a trained neural network.

A number of supervised ART architectures have been introduced by the Boston University group of researchers for performing supervised learning. These include ARTMAP (Carpenter, Grossberg and Reynolds, 1991) for which the input patterns to the network should be of binary nature, Fuzzy ARTMAP (Carpenter, et al., 1992), ARTE-MAP (Carpenter and Ross, 1995), Gaussian ARTMAP (Williamson, 1996), and ARTMAP-IC (Carpenter and Markuzon, 1998), where the input patterns are real-valued. The primary purpose of the last three contributions to the ART family is to improve the generalization performance of Fuzzy ARTMAP. In conjunction with the vigorous activity of researchers at Boston University in developing ART architectures, other researchers in the field, independently developed, analyzed, and applied ART architectures or ART-like architectures to a variety of problems. A short, incomplete list of such efforts include adaptive fuzzy leader clustering (AFLC) (Kim, Mitra, 1994), LAPART (Healy, et al., 1993), integrated adaptive fuzzy clustering (IAFC)(Kim and Mitra, 1994), fuzzy min-max clustering (Simpson, 1993), fuzzy min-max classification (Simpson, 1992), Adaptive Hamming net (Hung, et al., 1997), Boosted ARTMAP

The literature mostly relevant to the goals of this thesis is the work of Verzi (Verzi, et al., 2001) where he introduced the version of the semi-supervised ART architectures that we plan to focus on in our work, and the work of Anagnostopoulos (see Anagnostopoulos, et al., 2002) where he utilized a similar semi-supervised context for the special case of Ellipsoidal ART architectures. Other researchers in the past have tried to solve the category proliferation problem that ART networks suffer from. We single out the work of Williamson (Williamson, 1996), where he introduced a new ART architecture called Gaussian ARTMAP and demonstrated with experimentation that is more robust to noisy data than Fuzzy ARTMAP. It is also worth mentioning the work by Gomez-Sanchez, et al., (see Gomez-Sanchez, et al., 2002) where he introduced a modification of Fuzzy ARTMAP (called \( \mu \) ARTMAP) allowing training error, thus addressing the problem of overfitting and category proliferation. Gomez-Sanchez also presented experimental results that demonstrated the advantages of \( \mu \) ARTMAP compared to Fuzzy ARTMAP and BARTMAP (another variation of Fuzzy ARTMAP by Verzi (see Verzi et al., 1998) whose invention had similar goals). In a later conference paper Verzi (see Verzi, et al., 2001) introduced the concept of semi-supervision analyzed in this thesis and showed with preliminary
experimental results that his proposed semi-supervision approach has merit against a variety of ART algorithms (including Gaussian ARTMAP and $\mu$ ARTMAP). The purpose of this thesis is not to provide a detailed comparison of how the aforementioned approaches deal with the ART category proliferation problem. Instead the goal is to demonstrate, with the extensive and exhaustive experimentation, that the semi-supervision, initially proposed by Verzi (Verzi, et al., 2001) and later by Anagnostopoulos (Anagnostopoulos, et al., 2002), is indeed a meritorious method of handling the overtraining and category proliferation problem in ART, as well as in other exemplar-based classifiers.
Figure 1 *Simplified summary of the important ART networks*
3. Description of the network algorithms used

The three networks employed for experimentation are Fuzzy ARTMAP (FAM), Ellipsoidal ARTMAP (EAM) and the Gaussian ARTMAP (GAM). All the three networks have a three-layered structure. The three network algorithms operate in two phases namely the training phase and the performance phase. They create nodes in their category representation layer. In the training phase all the network algorithms create templates and those templates are carried on to the performance phase to find out the generalization performance of the network. The networks differ in the geometric form that each algorithm uses to represent the category node. FAM employs hyper-rectangles (rectangles in a 2 dimensional input space), EAM employs hyper-ellipses (ellipses in a 2 dimensional input space) and GAM uses Gaussian surfaces. The geometric shape of a category node is the shape the node will assume to mark its territory in the input domain.

Figure 2 Category regions of FAM, EAM and GAM respectively
All the three network algorithms follow the same training and performance procedures except for their geometrical representations and their category choice and match functions.

_Simplified Training Phase of all the network algorithms FAM, EAM & GAM_

1. The patterns in the training set are present at the input layer of the network. The order in which the patterns are presented is set to change (100 different orders of pattern presentation are presented in this work).
2. The patterns presented are training according to the steps of the algorithm it uses (FAM, EAM, GAM).
3. The patterns stop training when the convergence criterion of the algorithm is met. In this work we set the convergence criterion of all the network algorithms to be the same. The patterns stop training when no new nodes are created in that epoch or maximum number of list presentations is reached (in this work the maximum number of list presentations are limited to 100).
4. After the training is completed all the category nodes sustain the information needed for the operation of performance phase.
Simplified Performance Phase of all the network algorithms FAM, EAM & GAM

1. The patterns in the testing set are present at the input layer of the trained network.

2. The output, which is the class label of the pattern, is obtained from the output node of the trained network.

3. The obtained output class label is compared with the actual class label of the pattern and the performance of the network is calculated depending on the number of correctly classified patterns and number of incorrectly classified patterns.
4. FUZZY ARTMAP

The simplified FAM architecture consists of four layers of nodes as shown in Figure 2.1. The first layer, $F_0^a$, consists of $M$ nodes (where $M$ is the dimensionality of the input patterns) and it is also called as an input pre-processing layer. When an input vector $x$ of dimensionality $M$ is fed to the first layer it pre-processes the input layer by complement coding it and gives as an output the vector $I$ of dimensionality $2M$, such that

$$I = (x, x^c) = (x_1, \ldots, x_M, x_1^c, \ldots, x_M^c)$$

The second layer in the FAM architecture is the input layer $F_1^a$. This $F_1^a$ layer consists of $2M$ nodes. The output of the first layer is fed as an input to the second layer. The third layer in the FAM architecture is the category representation layer $F_2^a$. This layer has $N_a$ number of nodes. The value of $N_a$ is initially set to 1 and its value changes as the architecture proceeds. Each node in this layer is called a category. A category in the category representation layer is in reality a compressed representation of a group of patterns. A group of similar patterns are represented by one category. The fourth layer in the FAM architecture is the output layer $F_2^b$. It has exactly $N_b$ nodes where $N_b$ represents the number of output classes.
The actual architecture of FAM has two ART modules designated as $ART_a$ and $ART_b$, as well as an inter-ART module as shown in Figure. Inputs are presented at the $ART_a$ module, while their corresponding outputs are presented at the $ART_b$ module. The inter-ART module includes a MAP field whose purpose is to determine whether the correct mapping has been established from inputs to outputs.
The FAM architecture incorporates the knowledge it learns in the form of interconnection weights. It uses the following sets of weights:

1. Bottom-up weights, which originate from the input layer $F_1^a$ and terminate in the category representation layer $F_2^a$.

2. Top-down weights that originate from the layer $F_2^a$ and terminate in the layer $F_1^a$. 

Figure 4 Block diagram of FAM architecture
3. The inter-ART weights which are the interconnection weights originating from every node $j$ in the category representation region to every node in the output layer $F_2^b$.

FAM operates in two phases, training phase and performance (testing) phase. In the training phase FAM is presented with a set of input/output pairs and the network is trained. This is a learning phase in which FAM learns the output. In the testing phase the FAM is presented with the input and the output obtained is compared with the actual output and the performance of FAM is calculated. There are two parameters that are set by the user. These are the choice parameter $\beta$, and the baseline vigilance parameter $\rho$. The choice parameter $\beta$ takes values in the interval $(0, \infty)$ and it affects the bottom-up inputs applied at the nodes of layer $F_2^a$ when a pattern is presented in the input layer. The larger the choice parameter the greater is the tendency for the algorithm to create new nodes in the category representation layer. The baseline vigilance parameter $\rho$ takes values in the interval $[0, 1]$. Small values of $\rho$ (closer to 0) result in the network clustering dissimilar patterns together, in other words creating coarse clusters. The larger values of $\rho$ (closer to 1) result in the network clustering only very similar patterns together and hence creating many fine clusters.
The other network parameters like the vigilance parameter $\rho_a$ and the number of nodes $N_a$ in the category representation layer are the parameters that are tuned by EAM itself. The vigilance parameter $\rho_a$ takes value from $[\overline{\rho}_a, 1]$. It is by default set to $\overline{\rho}_a$ when training begins. In some special circumstances during training $\rho_a$ is incremented to values higher than $\overline{\rho}_a$. The value of the vigilance parameter $\rho_a$ is always reverted back to the baseline vigilance value $\overline{\rho}_a$ after the end of the presentation of an input/output pair. Suppose $\rho_a$ is incremented to $\overline{\rho}_a + y$ during the presentation of input/output pair $(I^r, O^r)$, then $\rho_a$ is reverted back to its original value of $\overline{\rho}_a$ when the next input/output pair $(I^{r+1}, O^{r+1})$ is presented. The last network parameter to be discussed is $N_a$, the number of nodes in the category representation layer plus one. The ‘one’ increment represents the one uncommitted node in the $F_z^a$ layer. The uncommitted node represents a node that has not encoded any of the input patterns presented so far. The $N_a$ parameter is not set by the user and is a function of the training process as a whole.
Training Phase of FAM:

The step-by-step implementation of off-line training in FAM is presented below:

1. Set the network parameter $\beta_a$ from the interval $(0, \infty)$ and $\bar{\rho}_a$ from the interval $[0, 1]$ and $\epsilon$ from the interval $[0, 1]$. The initial weight values for the top-down weights ($w_{ji}^{ab} ; j = 1, ..., N_a, i = 1, ..., 2M$) are chosen to be equal to 1. The initial weight values for the inter-ART weights ($W_{jk}^{ab} ; j = 1, ..., N_a, k = 1, ..., N_b$) are set to 0. The number of nodes in the $F_1^a$ layer is set to $2M$. The number of nodes in the $F_2^a$ layer is set to 1. Before training the only node in the $F_2^a$ layer is the uncommitted node. The number of nodes in the $F_2^b$ layer ($N_b$) is set to the number of output classes. If there are 4 output classes the number of nodes in the $F_2^b$ output layer is 4. The index $r$ of the input/output pairs is set to 1.

2. Present the $r^{th}$ input pattern ($I^r, O^r$) is presented to FAM. That is, the input pattern $I^r$ is presented at the input layer $F_1^a$ and the output $O^r$ is presented at the output layer $F_2^b$. The vigilance parameter $\rho_a$ is set to the baseline vigilance value $\bar{\rho}_a$.

3. Calculate the bottom-up inputs to all the nodes in the layer $F_2^a$ of FAM due to the presentation of input pattern $I^r$ at the input layer. All nodes including the uncommitted nodes are to be included when the bottom-up values are
computed. The bottom-up input for a node $j$ in $F_2^a$ is calculated according to the following equation.

$$T(w^a_j \mid I') = \frac{|I' \land w^a_j|}{\beta_a + |w^a_j|}$$

4. Choose the node in $F_2^a$ which receives the maximum bottom-up input. In other words, choose the node with the highest $T(w^a_j \mid I')$ value. Let us assume that the index of the chosen node is $j_{\text{max}}$. Once the node is chosen check to see if the node passes the vigilance test. Now we have three cases to consider.

   a. If node $j_{\text{max}}$ is the uncommitted node it satisfies the vigilance criterion and we can go on to Step 5.

   b. If node $j_{\text{max}}$ is a committed node and it satisfies the vigilance criterion we can go on to Step 5. A node $j_{\text{max}}$ satisfies the vigilance criterion if

   $$\frac{|I' \land w^a_{j_{\text{max}}}|}{|I'|} \geq \rho_u$$

   c. If node $j_{\text{max}}$ does not satisfy the vigilance criterion disqualify the node from the competition by setting $T(w^a_{j_{\text{max}}} \mid I') = -1$, and go back to Step 4.

5. At this point after the vigilance test, we can again distinguish three cases:

   a. If node $j_{\text{max}}$ is an uncommitted node and the output $O'$ of the input/output pair $(I', O')$ is such that its component $k_{\text{max}}$ is one and
the rest are zeros, then set \( W_{j_{\text{max}}}^{ab} = 1 \). For example in the four class classification problem (‘A’, ‘B’, ‘C’ & ‘D’) discussed earlier, if \( O^r \) represents ‘D’, that is \( O^r = [0 \ 0 \ 0 \ 1] \), then \( k_{\text{max}} \) would be 4 and \( W_{j_{\text{max}}}^{ab} = 1 \). The rest of the components of \( W_{jk}^{ab} \) would be set to zero. Furthermore, the top-down weight vector \( w_{j_{\text{max}}}^a \) becomes equal to \( w_{j_{\text{max}}}^a \land I^r \). After effecting these weight changes go to Step 6.

b. If node \( j_{\text{max}} \) is a committed node and the output \( O^r \) of the input/output pair \((I^r, O^r)\) is such that its component \( k_{\text{max}} \) is one and the rest are zeros, and at the same time \( W_{j_{\text{max}}}^{ab} \) is such that its component \( k_{\text{max}} \) is one and the rest are zeros, then the desired output \( O^r \) matches the actual output, represented by \( W_{j_{\text{max}}}^{ab} \) (e.g., for the four class letter recognition problem case both \( O^r \) and \( W_{j_{\text{max}}}^{ab} \) are equal to \([0 \ 0 \ 0 \ 1]\)). See Figure 2.3 for illustration. Now the top-down weight vector \( w_{j_{\text{max}}}^a \) becomes equal to \( w_{j_{\text{max}}}^a \land I^r \). After effecting this weight change go to Step 6.

c. If node \( j_{\text{max}} \) is a committed node and the Output \( O^r \) of the input/output pair \((I^r, O^r)\) is such that its component \( k_{\text{max}} \) is one and the rest are zeros, while at the same time \( W_{j_{\text{max}}}^{ab} \) is such that a different than the \( k_{\text{max}} \) component is one and the rest are zeros, then the desired output \( O^r \) does not match the actual output, represented by \( W_{j_{\text{max}}}^{ab} \). In this case node \( j_{\text{max}} \) is reset by setting...
$T(w_{rj_{\text{max}}}^{\alpha} \mid I^r) = -1$, the vigilance level is increased to the value of \\
\[\frac{|I^r \wedge w_{rj_{\text{max}}}^{\alpha}|}{|I^r|},\] and we return back to Step 4 to find another node \\
$j_{\text{max}}$ that maximizes the bottom-up input and satisfies the vigilance, \\
while predicting at the same time the correct output.

6. Unless all input/output pairs have been presented and we have reached \\
the an epoch (i.e., presentation of all the input/output pairs), $r$ is \\
incremented to $r + 1$ and we go back to Step 2 to present the $r + 1^{th}$ \\
input/output pair. If all input/output pairs have been presented then two \\
cases can be distinguished.

a. In the previous list presentation at least one component of the top-

down weights or the inter-ART weights have been changed. In this 

case we go back to Step 2 and present the first input/output pair in 

the set of input/output pairs, by setting $r$ to 1.

b. In the previous list presentation no weight changes occurred in the 

top-down weights and the inter-ART weights. Hence training is 

considered to be complete and the network is considered to have 

learnt the training patterns perfectly.

After the training process is completed, the weights $w_{ji}^{\alpha}; j = 1,....N_a, i = 1,....,2M$, 

and $W_{jk}^{ab}; j = 1,....,N_a, k = 1,....,N_b$, are stored to be used in the performance 

phase. Furthermore, for the performance phase of FAM, the network parameters
$\bar{\rho}_a$ and $\beta_a$ are set to be equal to the values that they had during the training phase of FAM.
Performance Phase of FAM:

The step-by-step implementation of FAM’s performance phase is described below:

1. Initialize the weights to the values that they had at the end of the training phase of FAM. The network parameters $\rho_a$, $\beta_a$, $D$, $\mu$ and $\omega$ are chosen equal to the values that they had during the training phase. The vigilance parameter value $\rho_a$ is set equal to the baseline vigilance parameter $\bar{\rho}_a$.

2. Present the $r$-th test pattern (i.e., test pattern $\tilde{I}^r$) to FAM. That is the test pattern is applied to the $F_1^a$ input layer.

3. Calculate the bottom-up inputs from $F_1^a$ to all nodes in $F_2^a$ layer due to the presentation of the test pattern $\tilde{I}^r$. During the computation all nodes including the uncommitted nodes are to be taken into account. The bottom-up inputs are computed based on the formula

$$T(w_j^a | \tilde{I}^r) = \frac{|\tilde{I}^r \land w_j^a|}{\beta_a + |w_j^a|}$$

4. Choose the $F_2^a$ node that receives the maximum bottom-up input from $F_1^a$. Let us assume that the index of the chosen $F_2^a$ node is $j_{max}$. Check to
see if this node satisfies the vigilance criterion. To do so, we distinguish three cases:

a. If node $j_{\text{max}}$ is an uncommitted node it satisfies the vigilance test automatically. Go to Step 5.

b. If node $j_{\text{max}}$ is a committed node and it satisfies the vigilance criterion go to Step 5. A node $j_{\text{max}}$ satisfies the vigilance criterion if

$$\frac{|\tilde{T}^r \wedge w_{j_{\text{max}}}^a|}{|\tilde{T}^r|} \geq \rho_a$$

c. If node $j_{\text{max}}$ does not satisfy the vigilance criterion disqualify the node from the competition by setting $T(w_{j_{\text{max}}}^a | \tilde{T}^r) = -1$, and then go back to the beginning of Step 4.

5. After the vigilance test, we distinguish three cases:

a. If node $j_{\text{max}}$ is uncommitted then the output of the presented test pattern is designated as “unknown”. Go to Step 6.

b. If node $j_{\text{max}}$ is a committed node, and $w_{j_{\text{max}}}^{ab} = 1$, while the rest of the $w_{j_{\text{max}}, k}^{ab}$'s are equal to 0, then designate the output of the network $O^r$ as the vector $w_{j_{\text{max}}}^{ab}$. For example in the four class case, if $k_{\text{max}} = 4$, then $w_{j_{\text{max}}}^{ab}$ is equal to $[0 \ 0 \ 0 \ 1]$. Set the output $O^r$ to be equal to $w_{j_{\text{max}}}^{ab}$ which is $[0 \ 0 \ 0 \ 1]$. Go to Step 6.

6. If all the test patterns in the test set have not been applied to the network then go back to Step 2 and present the next input/output test pair in the sequence. If we have presented all the input/output test pairs then the
results can be analyzed to find the misclassification error and other such statistics.
The architecture of Ellipsoidal ARTMAP (EAM) is similar to that of FAM except for the case such that the first layer of FAM which complement codes the input pattern is not present in EAM. Complement encoding of input patterns does not take place in EAM. The EAM architecture has only three layers of nodes.

![Simplified EAM architecture](image)

**Figure 5** *Simplified EAM architecture*

The first layer in the EAM architecture is the input layer $F_1^a$. This $F_1^a$ layer consists of M nodes. The second layer in the EAM architecture is the category representation layer $F_2^a$. This layer has $N_a$ number of nodes. The value of $N_a$ is
initially set to 1 and its value changes as the architecture proceeds. Each node in this layer is called a category. A category in the category representation layer is in reality a compressed representation of a group of patterns. A group of similar patterns are represented by one category. The third layer in the EAM architecture is the output layer $F_2^b$. It has exactly $N_b$ nodes where $N_b$ represents the number of output classes.

The EAM architecture incorporates the knowledge it learns in the form of interconnection weights. It uses the following sets of weights

1. Bottom-up weights, which originate from the input layer $F_1^a$ and terminate in the category representation layer $F_2^a$.

2. Top-down weights that originate from the layer $F_2^a$ and terminate in the layer $F_1^a$.

3. The inter-ART weights which are the interconnection weights originating from every node $j$ in the category representation region to every node in the output layer $F_2^b$.

EAM operates in two phases, training phase and performance (testing) phase. In the training phase EAM is presented with a set of input/output pairs and the network is trained. This is a learning phase in which EAM learns the output. In the testing phase the EAM is presented with the input and the output obtained is compared with the actual output and the performance of EAM is calculated. As in
all architectures there are two parameters that are set by the user. These are the choice parameter $\beta_a$, and the baseline vigilance parameter $\rho_a$. The choice parameter $\beta_a$ takes values in the interval $(0, \infty)$ and it affects the bottom-up inputs applied at the nodes of layer $F_2^a$ when a pattern is presented in the input layer. The larger the choice parameter the greater is the tendency for the algorithm to create new nodes in the category representation layer. The baseline vigilance parameter $\rho_a$ takes values in the interval $[0, 1]$. Small values of $\rho_a$ (closer to 0) result in the network clustering dissimilar patterns together, in other words creating coarse clusters. The larger values of $\rho_a$ (closer to 1) result in the network clustering only very similar patterns together and hence creating many fine clusters. Apart from these parameters EAM has its own EAM-specific parameters. The most important one is the common minor-to-major axis length ratio $\mu$. The minor-to-major axis length ratio $\mu$ takes values between 0 and 1. $\mu \in (0,1]$. It determines the ratio of the length of major axis and the minor axis in the ellipsoidal categories to be created. The ratio $\mu$ is preserved throughout the training and performance phase of EAM. The other parameters are $D > 0$, which is typically set equal to the EAM input space diameter $D_o$ ($D_o = \sqrt{M}/\mu$) and the parameter $\omega \geq 1/2$ for an uncommitted node’s CCF value, whose counterpart in FAM is $w_u$.

The other network parameters like the vigilance parameter $\rho_a$ and the number of nodes $N_a$ in the category representation layer are the parameters that are tuned
by EAM itself. The vigilance parameter $\rho_a$ takes value from $[\bar{\rho}_a, 1]$. It is by default set to $\bar{\rho}_a$ when training begins. In some special circumstances during training $\rho_a$ is incremented to values higher than $\bar{\rho}_a$. The value of the vigilance parameter $\rho_a$ is always reverted back to the baseline vigilance value $\bar{\rho}_a$ after the end of the presentation of an input/output pair. Suppose $\rho_a$ is incremented to $\bar{\rho}_a + y$ during the presentation of input/output pair $(I^r, O^r)$, then $\rho_a$ is reverted back to its original value of $\bar{\rho}_a$ when the next input/output pair $(I^{r+1}, O^{r+1})$ is presented. The last network parameter to be discussed is $N_a$, the number of nodes in the category representation layer plus one. The ‘one’ increment represents the one uncommitted node in the $F^a_2$ layer. The uncommitted node represents a node that has not encoded any of the input patterns presented so far. The $N_a$ parameter is not set by the user and is a function of the training process as a whole.
Training Phase of EAM

The step-by-step implementation of off-line training in EAM is presented below:

1. Set the network parameter $\beta_a$ from the interval $(0, \infty)$, $\bar{\rho}_a$ from the interval $[0, 1]$, $\mu$ from the interval $(0, 1]$, $D>0$ and $\omega \geq 1/2$. The initial weight values for the inter-ART weights $W_{jk}^{ab}$, $j = 1, \ldots, N_a$, $k = 1, \ldots, N_b$) is set to 0. The number of nodes in the $F_1^a$ layer is set to $M$. The number of nodes in the $F_2^a$ layer is set to 1. Before training the only node in the $F_2^a$ layer is the uncommitted node. The uncommitted node does not have any representation in EAM and hence the weight vector $w_{1i}$ is not initialized. It is initialized only when it encodes the first input pattern. The number of nodes in the $F_2^b$ layer is set to the number of output classes. If there are 4 output classes the number of nodes in the $F_2^b$ output layer is 4. The index $r$ of the input/output pairs is set to 1. The vigilance parameter value $\rho_a$ is set equal to the baseline vigilance parameter $\bar{\rho}_a$.

2. Present the $r^{th}$ input pattern $(I^r, O^r)$ is presented to EAM. That is, the input pattern $I^r$ is presented at the input layer $F_1^a$ and the output $O^r$ is presented at the output layer $F_2^b$. The vigilance parameter $\rho_a$ is set to the baseline vigilance value $\bar{\rho}_a$. 


3. Calculate the bottom-up inputs to all the nodes $F^u_2$ of EAM due to the presentation of input pattern $I'$ at the input layer. All nodes including the uncommitted nodes are to be included when the bottom-up values are computed. The bottom-up inputs for a node $j$ in $F^u_2$ are calculated according to the following equation.

$$T(w^a_j \mid I) = T(w^a_j \mid x) = \frac{D - R^a_j - \max \{R^a_j, \| x - m^a_j \|_{c_j} \}}{D - 2R^a_j + \beta_a}$$

4. Choose the node in $F^u_2$ which receives the maximum bottom-up input. In other words, choose the node with the highest $T(w^a_j \mid I')$ value. Let us assume that the index of the chosen node is $j_{\text{max}}$. Once the node is chosen check to see if the node passes the vigilance test. Now we have three cases to consider.

a. If node $j_{\text{max}}$ is the uncommitted node it satisfies the vigilance criterion and we can go on to Step 5.

b. If node $j_{\text{max}}$ is committed node and it satisfies the vigilance criterion we can go on to Step 5. A node $j_{\text{max}}$ satisfies the vigilance criterion if

$$\rho(w^a_j \mid I) = \rho(w^a_j \mid x) = 1 - \frac{R^a_j + \max \{R^a_j, \| x - m^a_j \|_{c_j} \}}{D} \geq \rho_a$$

c. If node $j_{\text{max}}$ does not satisfy the vigilance criterion disqualify the node from the competition by setting $T(w^a_{j_{\text{max}}} \mid x') = -1$, and go back to Step 4.

5. At this point after the vigilance test, we can again distinguish three cases:
a. If node \( j_{\text{max}} \) is an uncommitted node and the Output \( O' \) of the input/output pair \((I', O')\) is such that its component \( k_{\text{max}} \) is one and the rest are zeros, then set \( W_{j_{\text{max}}, k_{\text{max}}}^{ab} = 1 \). For example in the four class classification problem (‘A’, ‘B’, ‘C’ & ‘D’) discussed earlier, if \( O' \) represents ‘D’, that is \( O' = [0 \ 0 \ 0 \ 1] \), then \( k_{\text{max}} \) would be 4 and \( W_{j_{\text{max}}, 4}^{ab} = 1 \). The rest of the components of \( W_{jk}^{ab} \) would be set to zero. Furthermore, the top-down weight vector, the template, \( w_{j_{\text{max}}}^{a} \) is updated. That is the \( m_{j_{\text{max}}}^{a} \), \( d_{j_{\text{max}}}^{a} \) and \( R_{j_{\text{max}}}^{a} \) values of the category are changed as follows:

\[
\begin{align*}
m_{j_{\text{max}}}^{a} &= x'^{r} \\
d_{j_{\text{max}}}^{a} &= 0 \\
R_{j_{\text{max}}}^{a} &= 0
\end{align*}
\]

After effecting the weight change as described go to Step 6.

b. If node \( j_{\text{max}} \) is a committed node and the Output \( O' \) of the input/output pair \((I', O')\) is such that its component \( k_{\text{max}} \) is one and the rest are zeros, that is for the four class case it is \([0 \ 0 \ 0 \ 1]\), and \( w_{k_{\text{max}}, j_{\text{max}}}^{ab} \) also has its \( k_{\text{max}} \) component to be one and the rest of the \( k \) components as zeros, that is for the four class case \( W_{j_{\text{max}}}^{ab} = [0 \ 0 \ 0 \ 1] \), then the desired output \( O' \) matches the actual output, represented by \( W_{j_{\text{max}}}^{ab} \). Furthermore, the top-down weight vector \( w_{j_{\text{max}}}^{a} \) is updated. That is the \( m_{j_{\text{max}}}^{a} \) and \( R_{j_{\text{max}}}^{a} \) values of the category are changed as follows:

\[
\begin{align*}
m_{j_{\text{max}}}^{a} &= x'^{r} \\
d_{j_{\text{max}}}^{a} &= 0 \\
R_{j_{\text{max}}}^{a} &= 0
\end{align*}
\]
\[
m^{a}_{j_{\text{max}}^{\text{new}}} = m^{a}_{j_{\text{max}}^{\text{old}}} + \frac{1}{2} \left( 1 - \frac{\min \left\{ R^{a}_{j_{\text{max}}^{\text{old}}}, \| x^r - m^{a}_{j_{\text{max}}^{\text{old}}} \|_{c^{a}_{j_{\text{max}}^{\text{old}}}} \right\} \right) (x^r - m^{a}_{j_{\text{max}}^{\text{old}}})
\]

\[
R^{a}_{j_{\text{max}}^{\text{new}}} = R^{a}_{j_{\text{max}}^{\text{old}}} + \frac{1}{2} \left( \max \left\{ R^{a}_{j_{\text{max}}^{\text{old}}}, \| x^r - m^{a}_{j_{\text{max}}^{\text{old}}} \|_{c^{a}_{j_{\text{max}}^{\text{old}}}} \right\} - R^{a}_{j_{\text{max}}^{\text{old}}} \right)
\]

If the category being updated is a point category, that is, it is a category that has encoded only one pattern so far, then \(d_{j_{\text{max}}^{a}}\) is updated as follows,

\[
d^{a}_{j_{\text{max}}^{\text{new}}} = \frac{x^r - m^{a}_{j_{\text{max}}^{\text{old}}}}{\| x^r - m^{a}_{j_{\text{max}}^{\text{old}}} \|_2}
\]

If the category has encoded more than one patterns, then \(d_{j_{\text{max}}^{a}}\) is updated as follows

\[
d^{a}_{j_{\text{max}}^{\text{new}}} = d^{a}_{j_{\text{max}}^{\text{old}}}
\]

After effecting the changes in the templates go to Step 6.

c. If node \(j_{\text{max}}\) is a committed node and the Output \(O^r\) of the input/output pair \((I^r, O^r)\) is such that its component \(k_{\text{max}}\) is one and the rest are zeros, that is for the four class case it is \([0 0 0 1]\), and \(w^{ab}_{k_{\text{max}},j_{\text{max}}}\) has its \(k_{\text{max}} \neq 1\), that is for the four class case \(W^{ab}_{j_{\text{max}}} \neq [0 0 0 1]\), then the desired output \(O^r\) does not matches the actual output, represented by \(W^{ab}_{j_{\text{max}}}\). Hence node \(j_{\text{max}}\) is reset by setting \(T(w^{a}_{j_{\text{max}}} | x^r) = -1\), the vigilance level is increased to the value of

\[
1 - \frac{R^{a}_{j} + \max \left\{ R^{a}_{j}, \| x - m^{a}_{j} \|_{c^{a}_{j}} \right\}}{D}
\]

and we return back to Step 4 to find...
another node \( j_{\text{max}} \) that maximizes the bottom-up input and satisfies the vigilance, while predicting at the same time the correct output.

6. Unless all input/output pairs have been presented and we have reached the end of a pattern presentation, \( r \) is incremented to \( r+1 \) and we go back to Step 2 to present the \( r+1 \)th input/output pair. If all input/output pairs have been presented then two cases can be distinguished.

   a. In the previous list presentation at least one component of the top-down weights or the inter-ART weights have been changed. In this case we go back to Step 2 and present the first in the set of input/output pairs, by setting \( r \) to 1.

   b. In the previous list presentation no weight changes occurred in the top-down weights and the inter-ART weights. Hence training is considered to be complete and the network is considered to have learnt the training patterns perfectly.

7. After the training process is completed, the templates \( w_{j}^{a} \), \( j = 1, \ldots, N_{a} \), and \( w_{jk}^{ab} \), \( j = 1, \ldots, N_{a} \), \( k = 1, \ldots, N_{b} \), are stored to be used in the performance phase. Furthermore, for the performance phase of EAM, the network parameters \( \bar{\alpha}_{a} \), \( \beta_{a} \), \( D \), \( \mu \) and \( \omega \) are set to be equal to the values that they had during the training phase of EAM.
Performance Phase of EAM:

The step-by-step implementation of EAM performance phase is described below:

1. Initialize the templates $w_j^a$; $j = 1,...,N_a$ and the inter-ART weights $W_{jk}^{ab}$; $j = 1,...,N_a$, $k = 1,...,N_b$, to the values that they had at the end of the training phase of FAM. The network parameters $\beta_a$, $\rho_a$, $D$, $\mu$ and $\omega$ are chosen equal to the values that they had during the training phase. The vigilance parameter value $\rho_a$ is set equal to the baseline vigilance parameter $\overline{\rho}_a$.

2. Present the $r$-th test pattern (i.e., test pattern $\tilde{T}^r$) to FAM. That is the test pattern is applied to the $F_1^a$ input layer.

3. Calculate the bottom-up inputs from $F_1^a$ to all nodes in $F_2^a$ layer due to the presentation of the test pattern $\tilde{T}^r$. During the computation all nodes including the uncommitted nodes are to be taken into account. The bottom-up inputs are computed based on the formula

$$T(w_j^a \mid \tilde{T}) = T(w_j^a \mid \tilde{x}) = \frac{D - R_j^a - \max \{R_j^a, \|\tilde{x} - m_j^a\|_c\}}{D - 2R_j^a + \beta_a}$$

4. Choose the $F_2^a$ node that receives the maximum bottom-up input from $F_1^a$. Let us assume that the index of the chosen $F_2^a$ node is $j_{\text{max}}$. Check
to see if this node satisfies the vigilance criterion. To do so, we distinguish three cases:

a. If node $j_{max}$ is an uncommitted node it satisfies the vigilance test automatically. Go to Step 5.

b. If node $j_{max}$ is a committed node and it satisfies the vigilance criterion go to Step 5. A node $j_{max}$ satisfies the vigilance criterion if

$$
\rho(w^a_{j_{max}} \mid \tilde{x}) = 1 - \frac{R^a_{j_{max}} + \max \left\{ R^a_{j_{max}}, \| \tilde{x} - m^a_{j_{max}} \|_C \right\}}{D} \geq \overline{\rho}_a
$$

c. If node $j_{max}$ does not satisfy the vigilance criterion disqualify the node from the competition by setting $T(w^a_{j_{max}} \mid \tilde{x}^r) = -1$, and then go to back to the beginning of Step 4.

5. After the vigilance test, we distinguish two cases:

a. If node $j_{max}$ is uncommitted then the output of the presented test pattern is designated as "unknown". Go to Step 6.

b. If node $j_{max}$ is a committed node, and $w^{ab}_{j_{max}, k_{max}} = 1$, while the rest of the $w^{ab}_{j_{max}, k}$'s are equal to 0, then designate the output of the network $O'$ as the vector $W^{ab}_{j_{max}}$. For example in the four class case, if $k_{max} = 4$, then the output of $W^{ab}_{j_{max}}$ is $[0 \ 0 \ 0 \ 1]$. Set the output $O'$ to be equal to $W^{ab}_{j_{max}}$, which is $[0 \ 0 \ 0 \ 1]$. Go to Step 6.

6. If all the test patterns in the test set have not been applied to the network then go back to Step 2 and present the next input/output test pair in the sequence. If we have presented all the input/output test pairs then the
results can be analyzed to find the misclassification error and other such statistics.
6. GAUSSIAN ARTMAP (GAM)

This architecture is a modification of Fuzzy ARTMAP proposed by Williamson. Fuzzy ARTMAP performs well in some cases when compared to other learning systems. Though it is an efficient learning system in addition to the many advantages it has, it also suffers with disadvantages like sensitivity to noise and inefficiency of fuzzy categories. When the training data are noisy, so that regions of feature space essentially map randomly to different predictions, FA proliferates categories. This category proliferation problem is partly due to the fact that the choice and match functions are flat within a category’s hyper-rectangle, and partly due to the use of fast learning. To deal more efficiently with the problems of category proliferation in noise and category, a new ART module called Gaussian ART is introduced, which uses categories defined as Gaussian distributions. Gaussian ART is incorporated into ARTMAP architecture to create Gaussian ARTMAP. The structure of Gaussian ARTMAP (GAM) is very similar to Fuzzy ARTMAP except that the ART\textsuperscript{A} module is replaced by the Gaussian ART\textsuperscript{A} module, and no complement coding is done. GAM’s categories do not have a geometric representation in the same fashion as FAM categories do, they still correspond to the hyper-ellipsoidal regions embedded in the input space, which signify the set of patterns that constantly update the related category. These regions can also be thought of summarizing the data they include in some loose sense and they accomplish to form non-linear decision boundaries. GAM is a supervised – learning adaptive resonance theory network that uses Gaussian
defined receptive fields. Like other ART networks, GAM incrementally learns and constructs a representation of sufficient complexity to solve a problem it is trained on. The GAM classification algorithm bases the probability that input training samples belong to specific classes on the parameters of its Gaussian distributions: the means, standard deviations, and a priori probabilities. GAM accommodates choice and distributed learning and it can fit data that vary between dimensions, but not that co-vary. GAM is essentially an incremental learning Gaussian classifier in which each output class is determined during training to correspond to any number of sources of Gaussianly distributed data. One limitation in this analogy is that GAM can only define its categories with separable Gaussian distributions. This limitation is necessary so that GAM uses only simple operations that can be implemented in parallel.
Figure 6: Gaussian distribution that fit the independently varying data and co-varying data.

GA categories, which are defined by the separable Gaussian distributions, can capture independent variable well, figures (a) and (b) in the above figure shows the Gaussian distributions that fit the independently varying data. GA categories cannot capture the co-varying data; two possibilities for fitting these data are two smaller distributions or one larger distribution. Figures (c) and (d) in the above figure shows the two smaller distributions and one larger distribution respectively.

Each Gaussian ART category $j$ is defined by an $M$-dimensional vector $\mu_j$ representing its mean, $\sigma_j$ representing its standard deviation, and a scalar $n_j$ representing its count, the number of training samples it has coded. Thus each
Gaussian ART category requires $2M+1$ components to represent $M$-dimensional input. The main feature of the GAM is that it preferably identifies clusters with Gaussian distribution, in which the co-variance (off-diagonal) coefficients in the co-variance matrix describing the cluster are fixed to zero. The restriction was imposed on the GAM system for computational purposes, the reason being that with this kind of representation each cluster—identifying node is described by $2^*M+1$ parameters, where $M$ is the dimensionality of feature space. This network has the familiar properties of ART networks because categories are incrementally formed to represent clusters of input samples, and the inclusivity of the categories is inversely related to a vigilance parameter. The Gaussian ART activation function evaluates the probability that an input belongs to a category’s distribution, as well as the category’s a priori probability.

GAM operates in two phases, training phase and performance (prediction or testing) phase. In the training phase GAM is presented with set of input/output pairs $\{(I^{1}, O^{1})... (I^{r}, O^{r}) ... (I^{PT}, O^{PT})\}$, and is expected to match each input with its corresponding output. The input/output pairs are presented to the GAM one after the other in a serial pattern. The first input $I^{1}$ is presented at the $F^{a}_{1}$ layer and the corresponding output $O^{1}$ is presented at the $F^{b}_{2}$ layer and then the second input $I^{2}$ is presented at the $F^{a}_{2}$ layer and the corresponding output $O^{2}$ is presented at the $F^{b}_{2}$ layer and so on until the last input/output pair is presented. These training patterns are presented to the GAM until all the inputs are correctly matched with their corresponding outputs and no weights updated has been done during the
list presentation. This process of presentation of all the input/output pairs is called as off-line learning.

In the testing phase GAM presented with only the input pairs unlike in the training phase where it is presented with the input/output pairs, and the outputs are observed. The inputs are presented at the $F_1^{i\!}$ layer and the outputs are observed at the $F_2^{o\!}$ layer.

There are two parameters that are set by the user. These are the initial standard deviation $\gamma$ and the base line vigilance parameter $\rho_a$. The initial standard deviation $\gamma$ takes the values from $(0, 1]$ and affects the standard deviation of the network. The initial baseline vigilance takes the values in the interval $[0,1]$. The vigilance parameter $\rho_a$ and the number of nodes $N_a$ in the category representation layer are the two network parameters that are tuned by the GAM by itself. The vigilance parameter $\rho_a$ takes the values from $[\rho_a, 1]$. It is by default set to $\rho_a$ prior to the presentation of an input/output pair to GAM. It is important to note that $\rho_a$ is always reverted back to the baseline vigilance value $\rho_a$ after the end of presentation of an input/output pair. Suppose $\rho_a$ is incremented to $\rho_a + y$ during the presentation of input/output pair $(I^t, O^t)$, then $\rho_a$ is reverted back to its original value of $\rho_a$ when the next input/output pair $(I^{t+1}, O^{t+1})$ is presented. The last network parameter to be discussed is $N_a$, the number of nodes in the category representation layer plus one. The ‘one’
increment represents the one uncommitted node in the $F_2^u$ layer. The uncommitted node corresponds to a node that has not encoded any of the input patterns presented so far. Before the training commences the value of this parameter is initialized to 1 and thereafter it increases, as the rules of the GAM’s training phase require. For GAM, high vigilance ($\rho$) means that more internal categories are created by the network to match input data to output categories, i.e. that the categories are less broad in the feature space.
Training Phase of GAM:

The step-by-step implementation of the off-line training phase of GAM is as follows:

1. Set the network parameter $\bar{\rho}_a$ to a value from the interval $[0, 1]$. Also, initialize the parameter $\gamma$. The weight values corresponding to a node $j$ in $F_2^a$ are: $\mu_j$ (mean of the data that have activated and were encoded by node $j$), $\sigma_j$ (the standard deviation vector of the data that have activated and were encoded by node $j$), $n_j$ (the number of training input patterns that were encoded by node $j$ in $F_2^a$), and the inter-ART weights $W_{ab}^j$. The initial weight values for the inter-ART weights $(W_{jk}^{ab} ; j = 1, \ldots, N_a, k = 1, \ldots, N_b)$ are set equal to 0. As training progresses ever vector $W_{jk}^{ab}$ that has been committed has one of its components equal to 1 and the other components equal to zero. The component of $W_{jk}^{ab}$ that is equal to 1 designates the label that node $j$ is mapped to. The number of nodes in the $F_1^a$ layer is set to $2M_a$. The number of nodes in the $F_2^a$ layer is set to 1. Before training the only node in the $F_2^a$ layer is the uncommitted node. The number of nodes in the $F_2^h$ layer ($N_h$) is set to the number of output classes. If there are 4 output classes the number of
nodes in the $F_2^b$ output layer is 4. The index $r$ of the input/output pairs is set to 1.

2. Present the $r^{th}$ input pattern $(I^r, O^r)$ is presented to Gaussian-ARTMAP. That is, the input pattern $I^r$ is presented at the input layer $F_1^a$ and the output $O^r$ is presented at the output layer $F_2^b$. The vigilance parameter $\rho_a$ is set to the baseline vigilance value $\overline{\rho}_a$.

3. Calculate the bottom-up inputs to all the committed nodes $F_2^a$ of Gaussian ARTMAP due to the presentation of input pattern $I^r$ at the input layer. The bottom-up input for a node $j$ in $F_2^a$ is calculated according to the following equation.

$$g_j(I^r) = \frac{n_j}{\prod_{i=1}^{M_r} \sigma_{ji}} G_j(I^r)$$

where

$$G_j(I^r) = \exp \left( -\frac{1}{2} \sum_{i=1}^{M_r} \left( \frac{I_i - \mu_{ji}}{\sigma_{ji}} \right)^2 \right)$$

4. Choose the node in $F_2^a$ which receives the maximum bottom-up input. In other words, choose the node with the highest $g_j(I^r)$ value. Let us assume that the index of the chosen node is $J$. Once the node is chosen check to see if the node passes the vigilance test. Now we have three cases to consider.
a. If node $J$ is a committed node and it satisfies the vigilance criterion we can go on to Step 5. A node $J$ satisfies the vigilance criterion if

$$
\rho_j(I^r) = G_j(I^r) \geq \bar{\rho}_a
$$

b. If node $J$ does not satisfy the vigilance criterion disqualify the node from the competition by setting $g_j(I^r) = -1$ and go back to Step 4.

c. If no committed node can be found that meets the vigilance, then an uncommitted node category $J$ will be chosen.

5. At this point after the vigilance test, we can again distinguish three cases:

a. If node $J$ is an uncommitted node and the output $O^r$ of the input/output pair $(I^r, O^r)$ is such that its component $K$ is one and the rest are zeros, then set $W_{jk}^{ab} = 1$. For example in the four class classification problem (‘A’, ‘B’, ‘C’ & ‘D’) discussed earlier, if $O^r$ represents ‘D’, that is $O^r = [0 \ 0 \ 0 \ 1]$, then $K$ would be 4 and $W_{j4}^{ab} = 1$. The rest of the components of $W_{jk}^{ab}$ would be set to zero.

Furthermore,

$$
n_j = n_j + 1
$$

$$
\mu_{j_{\text{max}}} = I^r
$$

$$
\sigma_{j_{\text{max}}} = \gamma
$$
b. If node \( J \) is a committed node and the output \( \mathbf{O}^r \) of the input/output pair \((I^r, \mathbf{O}^r)\) is such that its component \( K \) is one and the rest are zeros, and at the same time \( W_{ij}^{ab} \) is such that its component \( K \) is one and the rest are zeros, then the desired output \( \mathbf{O}^r \) matches the actual output, represented by \( W_{ij}^{ab} \) (e.g., for the four class letter recognition problem case both \( \mathbf{O}^r \) and \( W_{ij}^{ab} \) are equal to \([0 0 0 1]\)). Furthermore,

\[
n_j = n_j + 1
\]

\[
\mu_j = \left(1 - \frac{1}{n_j}\right)\mu_j + \frac{1}{n_j}I^r
\]

\[
\sigma_{ji} = \sqrt{\left(1 - (n_j)^{-1}\right)\sigma_{ji}^2 + \left(\mu_j - I_i\right)^2}
\]

c. If node \( J \) is a committed node and the output \( \mathbf{O}^r \) of the input/output pair \((I^r, \mathbf{O}^r)\) is such that its component \( K \) is one and the rest are zeros, while at the same time \( W_{ij}^{ab} \) is such that a different than the \( K \) component is one and the rest are zeros, then the desired output \( \mathbf{O}^r \) does not match the actual output, represented by \( W_{ij}^{ab} \). In this case node \( J \) is reset by setting \( g_{ij}(I^r) = -1 \), the vigilance level \( \rho_a \) is increased to the value of \( G_{ij}(I^r) \), and we return back to Step 4 to find another node \( J \) that maximizes the bottom-up input and satisfies the vigilance, while predicting at the same time the correct output.
6. Unless all input/output pairs have been presented and we have reached
the end of an epoch (i.e., presentation of all the input/output pairs), \( r \) is
incremented to \( r+1 \) and we go back to Step 2 to present the \( r+1 \)th
input/output pair. If all input/output pairs have been presented then two
cases can be distinguished.

a. In the previous list presentation at least one component of the top-
down weights or the inter-ART weights have been changed. In this
case we go back to Step 2 and present the first input/output pair in
the set of input/output pairs, by setting \( r \) to 1.

b. In the previous list presentation no weight changes occurred in the
top-down weights and the inter-ART weights. Hence training is
considered to be complete and the network is considered to have
learnt the training patterns perfectly.
Performance Phase of GAM:

The step-by-step implementation of Gaussian ARTMAP’s performance phase is described below:

1. Initialize the weights $\mathbf{\mu}_j, \mathbf{\sigma}_j, n_j; \ j = 1,...,N_a, \ W_{jk}^{ab}; \ j = 1,...,N_a, \ k = 1,...,N_b$, to the values that they had at the end of the training phase of Gaussian ARTMAP.

2. Present the $r$-th test pattern (i.e., test pattern $\tilde{I}^r$) to Gaussian ARTMAP. That is the test pattern is applied to the $F_1^a$ input layer.

3. Calculate the bottom-up inputs from $F_1^a$ to all nodes in $F_2^a$ layer due to the presentation of the test pattern $\tilde{I}^r$. During the computation only the nodes that pass the vigilance are considered (i.e., nodes for which $\rho_j(\tilde{I}^r) = G_j(\tilde{I}^r) \geq \rho_a$). The bottom-up inputs are computed based on the formulas

$$g_j(\tilde{I}^r) = \frac{n_j}{\prod_{i=1}^{N_a} \sigma_{ji}} G_j(\tilde{I}^r)$$

where

$$G_j(\tilde{I}^r) = \exp \left( -\frac{1}{2} \sum_{i=1}^{N_a} \left( \frac{\tilde{I}^r_i - \mathbf{\mu}_j}{\sigma_{ji}} \right)^2 \right)$$
4 Find the node $K$ in $F_2^b$ node that maximizes the sum of the bottom-up inputs of nodes in $F_2^u$ that have the same label. That is $K$ is defined as follows:

$$K = \arg \max_k \left( g_j(I) \right)$$

5 If all the test patterns in the test set have not been applied to the network then go back to Step 2 and present the next input/output test pair in the sequence. If we have presented all the input/output test pairs then the results can be analyzed to find the misclassification error and other such statistics.
Category Choice Function (CCF):

Each Gaussian ART output node is represented by its simple mean, standard deviation along each dimension, and the number of patterns it encloses. From the Baye’s theorem the posterior probability of cluster $j$ given pattern $x$ is

$$P(j \mid x) = \frac{p(x \mid j)P(j)}{p(x)}$$

Since the clusters are defined by separable Gaussian distribution, the conditional probability density of $x$ given cluster $j$ is

$$p(x \mid j) = \frac{1}{(2\pi)^{M/2} \prod_{i=1}^{M} \sigma_{ji}} e^{-\frac{1}{2} \sum_{i=1}^{M} \left( \frac{x_i - w_{ji}}{\sigma_{ji}} \right)^2}$$

The priori probability of cluster $j$ is the ration of patterns it encompasses to the total number of patterns,

$$P(j) = \frac{n_j}{\sum_{j=1}^{N} n_j}$$

where $N$ is the number of categories.

If the match criterion is satisfied, the category’s net input signal, $g_j$, is determined by modulating its match value by $n_j$, which is proportional to the category’s a priori probability, and by $\left( \prod_{i=1}^{M} \sigma_{ji} \right)^{-1}$ which normalizes its Gaussian distribution. The
choice function of the template is proportional to the logarithm of its posterior probability, which is

\[
g_j(x) = \log \left( (2\pi)^{M/2} p(x | j) P(j) \right)
\]

\[
= -\frac{1}{2} \sum_{i=1}^{M} \left( \frac{\mu_{ji} - x_i}{\sigma_{ji}} \right)^2 - \log \left( \prod_{i=1}^{M} \sigma_{ji} \right) + \log(P(j))
\]

The density \(p(x)\) is ignored because it is the same for all categories and so is the dimensional scaling factor \((2\pi)^{M/2}\). The non-reset ART category \(J\) with maximum discriminant function is chosen. The cluster with maximum choice function is

\[
J = \arg \max_{j} \left( g_j(x) \right)
\]
Category Match Function (CMF):

The Gaussian ARTMAP match function is:

\[
g_j^1(x) = \log \left( (2\pi)^{M/2} \prod_{i=1}^{M} \sigma_{ji} \right) p(x \mid J) = -\frac{1}{2} \sum_{i=1}^{M} \left( \frac{\mu_{ij} - x_i}{\sigma_{ji}} \right)^2
\]

\[
= g_j^1(x) + \log \left( \prod_{i=1}^{M} \sigma_{ji} \right) - \log(P(J))
\]

If a chosen category’s match value does not satisfy the ART vigilance parameter \( \rho_a \), then the category is reset. Category match is determined by how well input \( I \) matches with the shape of category \( J \)'s distribution, which is normalized to a unit height \( g_j^1(I) \). If \( g_j^1(I) > \rho_a \), then the category resonates, otherwise it is reset.

Once a category is reset, it remains inactive until presentation of next input. If no committed ART category meets the vigilance condition, then an uncommitted category \( J^1 \), with \( n_{ji} = 0 \), is chosen.

When an ART category \( J \) is chosen for the first time during training, it is assigned the prediction \( K \), of the current training sample. If category \( J \) is again chosen in response to another training sample, and its prediction \( K' \) is incorrect \((K' \neq K)\), then match tracking is invoked. The vigilance parameter is raised to the value of the category’s match function, and category \( J \) is reset. Match tracking assures that a correct prediction comes from a category whose distribution is
much better to the training sample than all reset categories. Upon presentation of
the next training sample, $\rho_a$ is reassigned to its baseline vigilance value, $\rho_a = \bar{\rho}_a$.

When a category $J$ learns an input sample $I$ its count, mean and standard
deviation are updated to represent the sample count, mean and standard
deviation.

$$n_J = n_J + 1$$

$$\mu_J = \left(1 - \frac{1}{n_J}\right)\mu_J + \frac{1}{n_J}I'$$

$$\sigma_J = \sqrt{\left[(1 - (n_J)^{-1})\sigma_J^2 + \left(|n_J|^{-1}(\mu_J - I)^2\right]\right] \text{ if } n_J > 1}$$

$$= \gamma \quad \text{ otherwise.}$$
Williamson introduced distributed GAM in 1996. In his paper it was shown that the distributed GAM obtained a more efficient representation than GAM with choice learning. Distributed GAM has also been applied as part of an image classification system, where it outperformed existing state-of-the-art image classification system that use rule-based, multiplayer perceptron, and k-nearest neighbor classifiers. The distributed version of the GAM presented in this thesis uses only the distributed prediction. dGAM is almost similar to GAM the only difference arises during the prediction as dGAM uses distributed prediction and during the training dGAM follows exactly the same method as that of GAM. As the patterns are trained in exactly the same way that they are trained in GAM the number of nodes and categories created will be the same. In the distributed prediction each time an input is presented, the categories belonging to the same node sum their activation in order to generate a net probability of the class prediction that they share. And finally the network prediction is obtained from the maximum probability estimate, which decides the class label that the pattern belongs to.
Performance Phase of dGAM:

The step-by-step implementation of distributed Gaussian ARTMAP’s performance phase is described below:

1. Initialize the weights $\mu_j, \sigma_j, n_j; \ j = 1,...,N_a$, $W_{jk}^{ab}; \ j = 1,...,N_a, k = 1,...,N_b$, to the values that they had at the end of the training phase of Gaussian ARTMAP.

2. Present the $r$-th test pattern (i.e., test pattern $\tilde{I}^r$) to Gaussian ARTMAP. That is the test pattern is applied to the $F^a_1$ input layer.

3. Calculate the bottom-up inputs from $F^a_1$ to all nodes in $F^a_2$ layer due to the presentation of the test pattern $\tilde{I}^r$. During the computation only the nodes that pass the vigilance are considered (i.e., nodes for which $\rho(J | \tilde{I}^r) = G(J | \tilde{I}^r) \geq \rho_a$). The bottom-up inputs are computed based on the formulas

$$g(j | \tilde{I}^r) = \frac{n_j}{\prod_{i=1}^{M} \sigma_{ji}} G(j | \tilde{I}^r)$$

where

$$G(j | \tilde{I}^r) = \exp \left( -\frac{1}{2} \sum_{i=1}^{M} \left( \frac{I_i - \mu_{ji}}{\sigma_{ji}} \right)^2 \right)$$
4. Find the node $K$ in $F_2^b$ node that maximizes the bottom-up inputs of nodes in $F_2^a$. That is $K$ is defined as follows:

$$K = \arg \max_k \left( \sum_{j \in \text{input}} \exp(g(j \mid I')) \right)$$

5. If all the test patterns in the test set have not been applied to the network then go back to Step 2 and present the next input/output test pair in the sequence. If we have presented all the input/output test pairs then the results can be analyzed to find the misclassification error and other such statistics.
8. SEMI SUPERVISED GAM

Semi supervised learning in GAM is incorporated by performing an additional computational task in the training phase of the supervised version of GAM. The performance phase of the semi-supervised GAM (ssGAM) is identical to the performance phase of its supervised counterpart. Although only GAM is discussed in this section all the comments holds good with dGAM. In semi-supervised learning a user-defined amount of error is incorporated during the training phase of the neural network in order to improve the generalization performance.

In GAM during training, after selecting a winner category node to encode a pattern, we check to see if the class labels of the pattern and the category’s match. If the actual class label does not match with the predicted class label then the chosen node is eliminated and a different node is selected. In the semi-supervised version of GAM if the class labels do not match a test called Prediction Test (PT) is performed. This Prediction test is a test for the amount of error that the winner category node has already absorbed. If the amount of error already absorbed by the winner category is less than a user-defined amount of “allowable error” then the “erroneous pattern” is encoded by the category, otherwise, the algorithm proceeds as in the supervised case to search for a new category node that encode it. Erroneous patterns are defined as the patterns that were encoded by the category and their class label was different than the initial
class label. The Prediction test computes the number of erroneous patterns that the category node has encoded, meaning the number of patterns from classes other than the initial class label.

The amount of acceptable error is user defined and is controlled by the tunable error tolerance parameter $\varepsilon$, which takes values between 0 and 1. A value of 0 for $\varepsilon$ would mean it allows no error while constructing categories whereas a value of 1 for $\varepsilon$ would mean 100% error is allowed. The former case ($\varepsilon = 0$) reduces the semi-supervised algorithms to supervised algorithms (that is, ssGAM to GAM) and the latter case ($\varepsilon = 1$) reduces the semi-supervised algorithms into unsupervised algorithms or clustering algorithms. If, for example, the value for $\varepsilon$ is set as 0.1, it amounts to allowing a maximum of 10% error during the construction of categories. Similarly, $\varepsilon = 0.3$ equals 30% maximum error, $\varepsilon = 0.7$ equals 70% maximum error and so on. It has to be noted that 30% or 70% error refers to the maximum error that is allowed and does not imply that all category nodes will be constructed with an error of 30% or 70%. The parameter only sets the roof on the error tolerance and does not impose error on the network. In other words, the amount of error each category will allow is never greater than $\varepsilon \times 100\%$. 
Prediction Test:

In order to understand the Prediction Test it is necessary to address how the connectivity between the first hidden layer and the output layer is constructed in exemplar based classifiers. These weights are called inter-ART weights in ART neural networks. These weights are represented as $W_{jk}^{ab}$, where $j$ refers to the index of the node in the hidden layer and $k$ refers to the index of the node in the output layer. Each node in the output layer represents an output class. (Each node can be thought of as having a class label.) In supervised classifiers there exists just one connection from a node $j$ in the hidden layer to a node $k$ in the output layer. Node $j$ is not connected to any other node in the output layer. However in semi-supervised algorithms a node $j$ in the hidden layer can have connections to many nodes in the output layer. Each connection has a weight associated with it. This weight is equal to the number of patterns that a node in the hidden layer has encoded for that particular class that the output layer node represents. Suppose a connection between a node $j$ in the hidden layer and a node $k$ in the hidden layer has a weight of 3, it means that node $j$ has encoded 3 patterns from class $k$ (or the class that node $k$ represents).

The Prediction test for node $J$ is defined as follows.

$$\frac{W_{J,J(J)}}{1 + \sum_{c=1}^{C} W_{J,c}} \geq 1 - \varepsilon$$
where

c = 1,2…C represents the classes in the dataset,

\( I(J) \) = the initial class label and

\( w_{J,c} \) = the number of patterns encoded by the category node \( J \) with the class label \( c \).

\( \varepsilon \) = the tunable error tolerance parameter of the network, which determines how much error is permissible. For a fixed value of \( \varepsilon \) the network will allow only \( \varepsilon \times 100\% \) error in the construction of category nodes of the neural network.

In other words, the prediction test guarantees that the number of legitimate patterns encoded by the category divided by the total number of patterns encoded by the category plus one, is greater than or equal to \( 1-\varepsilon \). Legitimate patterns are the patterns that were encoded by the category and their class label was the same as the category’s initial label.

\[
\frac{\text{legitimate\_patterns\_encoded}}{1 + \text{total\_patterns\_encoded}} \geq 1 - \varepsilon
\]
GAM and ssGAM Pseudo Code

**GAM Pseudo Code:**

For (every Training pattern)

```
{ 
  1. \( S = \{ \text{Category Choice function (CCF) value for every category in } F_2^a \} \); 
  2. Select the category with max CCF value from S; 
  3. Perform Vigilance test on selected category; 
  4. If (selected category passes vigilance test) 
    a. If (selected category has same class label) 
      allow category to encode pattern; continue with next pattern; 
    b. else 
      Initiate Match Tracking mechanism; 
  5. If (no category passes Vigilance test) 
    Commit Uncommitted node 
}
```

**ssGAM Pseudo Code:**

For (every Training pattern)

```
{ 
  1. \( S = \{ \text{Category Choice function (CCF) value for every category in } F_2^a \} \); 
  2. Select the category with max CCF value from S; 
  3. Perform Vigilance test on selected category; 
  4. If (selected category passes vigilance test) 
    a. if (selected category has same class label) 
      allow category to encode pattern; continue with next pattern; 
```
b. else

   i. Perform PREDICTION TEST (PT);

      if (selected category passes PT)

         allow category to encode pattern; continue with next

         pattern;

   ii. Else

       Initiate Match tracking mechanism;

5. If (no category passes Vigilance test)

   Commit Uncommitted node

}
Training Phase of ssGAM:

The ssGAM in addition to the GAM parameter values it has one extra parameter, denoted by $\varepsilon$, which sets the limit to the error tolerance for the categories in the $F^a_i$. It takes values from 0 to 1. It has to be set to an appropriate value prior to beginning of training. There is also additional consideration for ssFAM. Each category node in the $F^a_i$ layer also has a parameter called the initial class label $I(w^a_j)$. This parameter stores the class label of the category when it was first created. This is necessary in ssGAM since the inter-ART weights could have more than one non-zero components.

The step-by-step implementation of the off-line training phase of GAM is as follows:

1. Set the network parameter $\bar{\rho}_a$ to a value from the interval $[0, 1]$. Also, initialize the parameter $\gamma$. The weight values corresponding to a node $j$ in $F^a_i$ are: $\mu_j$ (mean of the data that have activated and were encoded by node $j$), $\sigma_j$ (the standard deviation vector of the data that have activated and were encoded by node $j$), $n_j$ (the number of training input patterns that were encoded by node $j$ in $F^a_i$), and the inter-ART weights $W^a_{jk}$. The initial weight values for the inter-ART weights $(W^a_{jk}; j = 1,...,N_a, k = 1,...,N_b)$ are set equal to 0. As training progresses
ever vector $W_{jk}^{ab}$ that has been committed has one of its components equal to 1 and the other components equal to zero. The component of $W_{jk}^{ab}$ that is equal to 1 designates the label that node $j$ is mapped to. The number of nodes in the $F_1^a$ layer is set to $2M_a$. The number of nodes in the $F_2^a$ layer is set to 1. Before training the only node in the $F_2^a$ layer is the uncommitted node. The number of nodes in the $F_2^b$ layer ($N_b$) is set to the number of output classes. If there are 4 output classes the number of nodes in the $F_2^b$ output layer is 4. The index $r$ of the input/output pairs is set to 1.

2. Present the $r^{th}$ input pattern $(I^r, O^r)$ is presented to Gaussian-ARTMAP. That is, the input pattern $I^r$ is presented at the input layer $F_1^a$ and the output $O^r$ is presented at the output layer $F_2^b$. The vigilance parameter $\rho_a$ is set to the baseline vigilance value $\bar{\rho}_a$.

3. Calculate the bottom-up inputs to all the committed nodes $F_2^a$ of Gaussian ARTMAP due to the presentation of input pattern $I^r$ at the input layer. The bottom-up input for a node $j$ in $F_2^a$ is calculated according to the following equation.

$$g_j(I^r) = \frac{n_j}{\prod_{i=1}^{M_a} \sigma_{ji}} \cdot G_j(I^r)$$

where...
\[ G_j(I^r) = \exp \left( -\frac{1}{2} \sum_{i=1}^{M} \left( \frac{I_i - \mu_{ji}}{\sigma_{ji}} \right)^2 \right) \]

4. Choose the node in \( F^a_2 \) which receives the maximum bottom-up input. In other words, choose the node with the highest \( g_j(I^r) \) value. Let us assume that the index of the chosen node is \( J \). Once the node is chosen check to see if the node passes the vigilance test. Now we have three cases to consider.

a. If node \( J \) is a committed node and it satisfies the vigilance criterion we can go on to Step 5. A node \( J \) satisfies the vigilance criterion if
\[ \rho_j(I^r) = G_j(I^r) \geq \overline{\rho}_a \]

b. If node \( J \) does not satisfy the vigilance criterion disqualify the node from the competition by setting \( g_j(I^r) = -1 \) and go back to Step 4.

c. If no committed node can be found that meets the vigilance, then an uncommitted node category \( J \) will be chosen.

5. At this point after the vigilance test, we can again distinguish three cases:

a. If node \( J \) is an uncommitted node and the output \( O^r \) of the input/output pair \((I^r, O^r)\) is such that its component \( K \) is one and the rest are zeros, then set \( W_{jk}^{ah} = 1 \). For example in the four class classification problem ('A', 'B', 'C' & 'D') discussed earlier, if \( O^r \) represents 'D', that is \( O^r = [0 \ 0 \ 0 \ 1] \), then \( K \) would be 4 and \( W_{j4}^{ah} = 1 \). The rest of the components of \( W_{jk}^{ah} \) would be set to zero. Furthermore,
\[ n_j = n_j + 1 \]

\[ \mu_{j_{\text{max}}} = I^r \]

\[ \sigma_{j_{\text{max}}} = \gamma \]

b. If node \( J \) is a committed node and the output \( O^r \) of the input/output pair \((I^r, O^r)\) is such that its component \( K \) is one and the rest are zeros, and at the same time \( W_{j}^{ab} \) is such that its component \( K \) is one and the rest are zeros, then the desired output \( O^r \) matches the actual output, represented by \( W_{j}^{ab} \) (e.g., for the four class letter recognition problem case both \( O^r \) and \( W_{j}^{ab} \) are equal to \([0 \ 0 \ 0 \ 1])\). Furthermore,

\[ n_j = n_j + 1 \]

\[ \mu_j = \left(1 - \frac{1}{n_j}\right) \mu_j + \frac{1}{n_j} I^r \]

\[ \sigma_{j} = \sqrt{\left(1 - (n_j)^{-1}\right) \sigma_{j}^2 + \left(1 - (n_j)^{-1} (\mu_j - I_j)^2\right)} \]

c. If node \( J \) is a committed node and the Output \( O^r \) of the input/output pair \((I^r, O^r)\) is such that its component \( K \) is one and the rest are zeros, while at the same time \( W_{j}^{ab} \) is such that a different than the \( K \) component is one and the rest are zeros, then the desired output \( O^r \) does not match the actual output, represented by \( W_{j}^{ab} \).
Prediction Test is performed on the category J with initial label equal to \( I(W_{j_{\text{max}}}^u) \), to find out if the category encountered the maximum number of erroneous patterns it is allowed to.

\[
\frac{W_{j_{\text{max}}}^u I(w_{j_{\text{max}}}^u)}{1 + \sum_{\varepsilon = 1}^{\varepsilon_{\text{max}}} W_{j_{\text{max}}}^\varepsilon} \geq 1 - \varepsilon
\]

where, \( \varepsilon \in [0, 1] \) = error tolerance parameter set by the user prior to beginning of experimentation.

Here we distinguish two cases:

i. If the category passes the prediction test, then the inter-ART weights are updated such that \( W_{j_{\text{max}}}^{ab} = W_{j_{\text{max}}}^{ab} + 1 \). That is if \( W_{j_{\text{max}}}^{ab} = [0 \ 0 \ 1 \ 0] \), then after the inter-ART weight update \( W_{j_{\text{max}}}^{ab} = [0 \ 0 \ 1 \ 1] \). Furthermore, the top-down weight vector \( w_j^a \) becomes equal to \( w_j^a \land I' \). After effecting these weight changes go to Step 6.

ii. If the category fails the prediction test, In this case node J is reset by setting \( g_j(I') = -1 \), the vigilance level \( \rho_a \) is increased to the value of \( G_j(I') \), and we return back to Step 4 to find another node J that maximizes the bottom-up input and satisfies the vigilance, while predicting at the same time the correct output.
6. Unless all input/output pairs have been presented and we have reached the end of an epoch (i.e., presentation of all the input/output pairs), \( r \) is incremented to \( r+1 \) and we go back to Step 2 to present the \( r+1^{\text{th}} \) input/output pair. If all input/output pairs have been presented then two cases can be distinguished.

a. In the previous list presentation at least one component of the top-down weights or the inter-ART weights have been changed. In this case we go back to Step 2 and present the first input/output pair in the set of input/output pairs, by setting \( r \) to 1.

b. In the previous list presentation no weight changes occurred in the top-down weights and the inter-ART weights. Hence training is considered to be complete and the network is considered to have learnt the training patterns perfectly.
Performance Phase of ssGAM:

The step-by-step implementation of semi-supervised Gaussian ARTMAP's performance phase is described below. It is exactly same as the performance phase of GAM.

1. Initialize the weights \( \mu_j, \sigma_j, n_j; \ j = 1,\ldots,N_a, \ W_{jk}^{ab}; \ j = 1,\ldots,N_a, \ k = 1,\ldots,N_b, \) to the values that they had at the end of the training phase of Gaussian ARTMAP.

2. Present the \( r \)-th test pattern (i.e., test pattern \( \tilde{I}^r \)) to Gaussian ARTMAP. That is the test pattern is applied to the \( F_1^a \) input layer.

3. Calculate the bottom-up inputs from \( F_1^a \) to all nodes in \( F_2^a \) layer due to the presentation of the test pattern \( \tilde{I}^r \). During the computation only the nodes that pass the vigilance are considered (i.e., nodes for which \( \rho_j(\tilde{I}^r) = G_j(\tilde{I}^r) \geq \bar{\rho}_a \)). The bottom-up inputs are computed based on the formulas

\[
 g_j(\tilde{I}^r) = \frac{n_j}{\prod_{i=1}^{N_a} \sigma_{ji}} G_j(\tilde{I}^r)
\]

where

\[
 G_j(\tilde{I}^r) = \exp \left( -\frac{1}{2} \sum_{i=1}^{M_a} \left( \frac{\tilde{I}_i - \mu_{ji}}{\sigma_{ji}} \right)^2 \right)
\]
4. Find the node $K$ in $F_2^b$ node that maximizes the sum of the bottom-up inputs of nodes in $F_2^a$ that have the same label. That is $K$ is defined as follows:

$$K = \arg \max_k \left( g_j(I) \right)$$

5. If all the test patterns in the test set have not been applied to the network then go back to Step 2 and present the next input/output test pair in the sequence. If we have presented all the input/output test pairs then the results can be analyzed to find the misclassification error and other such statistics.
9. EXPERIMENTS

The experiments were performed by using two types of databases.

1. Artificial Databases
2. Real Databases

Artificial databases were used because we can generate as many training, cross-validation, and test data, as we desire. Artificial databases were used also because they allow us to change the dimensionality of the input patterns and the number of output classes, at will. In this thesis all the data used is restricted to be 2 dimensional data. Furthermore, data with 2, 4 and 6 output classes were used. For each of the 2-dimensional, 2, 4 and 6 output class data generated the overlap of data belonging to different classes was chosen to be 5%, 15%, 25% and 40%. The amount of overlap amongst data belonging to different classes affects the difficulty of the problem (higher overlap results in more difficult problem).

The databases are trained with all variations of parameters, i.e. 121,000 experiments with different combinations of parameters in the case of ssGAM. The trained networks are then tested for their performance against the cross-validation set. The effect of each of the network parameters is closely examined. 100 best networks, (networks are considered to be best depending on the percent correct classification of the networks on the cross-validation set) are
selected and their performance is tested against the testing set. Finally the network which maximizes the generalization performance on the cross-validation set is considered to be the best network.
Artificial Databases

Data Generation:
All patterns are 2 dimensional (2-D), Gaussianly distributed data, belonging to 2, 4 and 6 different classes. For the 2-class problem the centers of the two Gaussian populations are at \((-d/2, 0)\) and at \((+d/2, 0)\). For the 4-class problem the centers of the four Gaussian populations are located at \((+d/2, +d/2)\), \((-d/2, +d/2)\), \((-d/2, -d/2)\), \((+d/2, -d/2)\), for class 1, 2, 3, and 4, respectively. For the 6-class problem the Gaussian populations are located at the circumference of a circle of radius \(d/2\). The first center is at location \((d/2, 0)\), and the rest of the centers are found by traversing the trajectory of the circle’s circumference and defining subsequent centers separated by 60 degrees from the previously defines center. All the Gaussian populations have a common covariance matrix equal to \(I\), where \(I\), stands for the unit matrix in 2-D. The data have equal probability of being drawn from any of the Gaussian populations of the 2-class, 4-class or 6-class problems. The parameter \(d\) that determines the center separation is thus chosen so that the overlap between the data is equal to 5%, 15%, 25%, and 40%. The overlap of the data is defined to be equal to the error that the best (Bayesian classifier) commits when confronted with the aforementioned datasets. In the following figures we are showing a pictorial illustration of the 2-class, 4-class and 6-class datasets, when the class overlap is 5%.
Figure 7: The figure showing the 2-dimensional 2-class Gaussian data

Figure 8: figure showing the 2-dimensional, 4-class Gaussian data
Figure 9: figure showing the 2-dimensional, 6-class Gaussian data
Real Databases

To verify the results obtained with artificial databases, experiments were conducted using real databases. Three databases were chosen from the UCI Machine Learning repository. They are the Abalone database, the Page-blocks database and the Iris database.

For each one of the real databases the training data set consisted of 500 data points selected from the database in a way that reflects the distribution of classes in the database (see more details in the sequel regarding of how we chose 500 training data points for the Iris database). The selection of the training points within each class is however arbitrary. The number of points remaining in the database after extracting the training data set is equally divided to form the testing data set and the cross validation data set. In all of our experiments we used the training set to design the ART classifier. Then, we observed its performance (in terms of network size and classification accuracy) either on the validation set or on the test set. If we were attempting to find the best-trained network over a range of network parameter values, we used the validation set to determine the set or sets of optimal network parameter values and then we observed the network’s performance on the test set. In the following we describe in more detail each of the real datasets used in our experiments.
**Abalone Database:**

This database is used to predict the age of abalone. The data set has 8 attributes and 4177 instances. The eight attributes represent the sex, length, diameter, height, weight, etc. The number of rings in the shell of the abalone indicates the age. The database was treated as a classification problem by grouping the number of rings into three categories (1-8, 9-10, and 11-greater) and using the three groups as output classes. Experiments with the Abalone database have been conducted using the aforementioned grouping of categories strategy (see Clark, 96). In our experiments, the first attribute representing the sex of the abalone was discarded since it was non-numerical. We conjecture that the Abalone dataset has high overlap among its output classes since the best generalization results reported in the literature are around 60%.

**Pageblocks Database:**

The problem behind the pageblocks database consists of classifying the blocks of the page layout of a document that has been detected by a segmentation process. This classification process is seen as an essential step in document analysis in order to separate text from graphic areas. The database has 10 attributes, 5473 points with five output classes, which are text, horizontal line, picture, vertical line and graphic.
**Iris Database:**

It is a 4 dimensional database consisting of 4 attributes. Each dimension represents the sepal length in cm, sepal width in cm, petal length in cm and petal width in cm. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other. Taking cue from [Keller ‘85], data points representing the two non-linearly separable classes are extracted and features 1-2 are eliminated. The three classes represent Iris Setosa, Iris Versicolour and Iris Virginica, which are designated as class-0, class-1 and class-2 respectively. The Iris dataset is expanded by generating 100 more points around each point in the original Iris dataset, with these points as means and a small variance (0.1) around the points. This operation is performed to increase the size of the dataset without adversely affecting the nature of the database. With this expanded Iris data set we could choose 500 data-points (as with the previous databases) for training and have enough points left to perform reliable cross-validation and testing. The small variance added justifies our claim that the nature of the database is not adversely affected.
Performance Comparisons of dGAM and GAM, and Effect of Network Parameters

GAM vs. dGAM:

Table 1: Comparison of GAM and dGAM for 2, 4 and 6 class problems with 5, 15, 25 and 40% overlap.

<table>
<thead>
<tr>
<th></th>
<th>2 Class</th>
<th></th>
<th>4 Class</th>
<th></th>
<th>6 Class</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GAM</td>
<td>NC</td>
<td>dGAM</td>
<td>NC</td>
<td>GAM</td>
<td>NC</td>
</tr>
<tr>
<td>5%</td>
<td>95.0%</td>
<td>20</td>
<td>95.1%</td>
<td>72</td>
<td>94.8%</td>
<td>16</td>
</tr>
<tr>
<td>15%</td>
<td>84.9%</td>
<td>30</td>
<td>84.7%</td>
<td>17</td>
<td>84.7%</td>
<td>26</td>
</tr>
<tr>
<td>25%</td>
<td>75.1%</td>
<td>10</td>
<td>75.0%</td>
<td>22</td>
<td>72.7%</td>
<td>57</td>
</tr>
<tr>
<td>40%</td>
<td>61.1%</td>
<td>19</td>
<td>61.4%</td>
<td>500</td>
<td>59.1%</td>
<td>47</td>
</tr>
</tbody>
</table>

Table 1 above compares the performance of GAM and dGAM on the artificial databases. In our experiments we have trained GAM and dGAM networks for different values of the GAM network parameters. In particular, we experimented with 10 different values of the baseline vigilance (all values starting from 0.0 and increasing to 0.9 with step of 0.1), 10 different values of the initial standard deviation parameter $\gamma$ (all values starting from 0.1 and increasing to the value of 1.0 with step 0.1) and 100 different order of pattern presentations within the training set. All these 10,000 trained GAMs and dGAMs had their performance evaluated on the validation set and the best performing (in terms of classification accuracy GAM and dGAM were chosen). For these best performing GAM and dGAM we are reporting in the above table the network size and the classification
accuracy of the network on the test set. Similarly, the results are also reported, in a pictorial fashion, in Figures 10 and 11 below. The pertinent observation from this table and figures is that although dGAM occasionally exhibits slightly better performance than GAM it does so at the expense of creating networks with more (sometimes significantly more) categories than its GAM counterpart.

Figure 10: Graph showing the best performances of GAM and dGAM for 2, 4 and 6, class problems in each case. PCC stands for the percentage of Correct Classification.
Figure 11: Graph showing the comparison of number of categories created in each case by the best GAM and dGAM networks. Best is defined in terms of highest Percentage of Correct Classification. NC stands for Numbers of Categories.

Figure 12: Graph showing the comparison of average of the Best 100 results of GAM and dGAM in each case. Best means in terms of highest classification accuracy on the cross-validation set.
Figures 12 and 13 show the average classification of the best 100 networks and all the GAM and dGAM networks. In Figure 12, we observe that the average classification performance of the best 100 networks is identical for GAM and dGAM. These two networks seem to appreciably differentiate when their classification performance is averaged over all the trained networks. This seems to indicate that if we have enough data to reliably cross-validate to find the best network parameters and enough computational resources to perform exhaustive experimentation to find these best parameters there is no appreciable difference (in classification performance) between GAM and dGAM.
Figure 14: Graph shows the average PCC, std. PCC, average number of categories created and std NC of the networks for every rho value of the 2 class 15% overlap problem for the GAM and dGAM respectively.

The graph in the figure above shows the performance of the GAM and dGAM networks for every rho (\(\rho\)) value. The figures above show the effect of the rho value on the network size and the classification accuracy of the network. The graph shows that the average classification performance of the networks is good for almost all rho values smaller than or equal to 0.8. But, the average network size and the standard deviation of the network size monotonically increase as the rho value increases. Consequently, it seems that rho values closer to a rho value of 0 is working the best.
Figure 15: Graph showing the performance of the 2 class 15% problem for each particular gamma for the GAM and dGAM respectively.

The graph in the above figure shows that the average performance of the networks for a variety of gamma values. Here we observe that the average classification performance stays pretty unaffected by the specific gamma value used. But the average size of the network created decreases monotonically as the gamma value increases. This leads us to the conclusion that larger gamma values (closer to the value of 1.0) seem to be producing the best (on the average) networks.
Semi-supervised GAM and dGAM:

When we compare the performance of GAM, dGAM (the supervised ART versions) with their corresponding semi-supervised counterparts we can make the following observations. Table 2 depicts the best performing (in terms of generalization performance) GAM, dGAM, ssGAM, and ssdGAM networks. One observation from this table is that the generalization performance of the semi-supervised networks is not improved by semi-supervision. Another observation from this table is that the size of the semi-supervised networks is not significantly reduced by semisupervision (except for the case of the 2-class Gaussian dataset). Table 3 reaffirms these observations too. In table 3 the average generalization performance of GAM, dGAM (corresponds to epsilon equal to 0) and ssGAM, ssdGAM (corresponds to positive epsilon values) is reported. From these results it is obvious that the best average generalization performance is attained for GAM and dGAM instead of ssGAM and ssdGAM.
Table 2: Summary of the results of the generalized performance of the networks produced on the cross validation set

### 2 Class Problem

<table>
<thead>
<tr>
<th></th>
<th>ssGAM</th>
<th></th>
<th>ssdGAM</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GAM</td>
<td>Best ssGAM</td>
<td>ε</td>
<td>comp</td>
<td>dGAM</td>
<td>Best ssdGAM</td>
<td>ε</td>
</tr>
<tr>
<td>5%</td>
<td>94.9%</td>
<td>94.9%</td>
<td>0.1</td>
<td>20/90</td>
<td>94.9%</td>
<td>94.9%</td>
<td>0.3</td>
</tr>
<tr>
<td>15%</td>
<td>85.4%</td>
<td>85.5%</td>
<td>0.1</td>
<td>30/8</td>
<td>85.3%</td>
<td>85.6%</td>
<td>0.2</td>
</tr>
<tr>
<td>25%</td>
<td>75.1%</td>
<td>75.2%</td>
<td>0.2</td>
<td>10/10</td>
<td>75.2%</td>
<td>75.3%</td>
<td>0.2</td>
</tr>
<tr>
<td>40%</td>
<td>59.2%</td>
<td>59.6%</td>
<td>0.7</td>
<td>19/3</td>
<td>59.6%</td>
<td>59.8%</td>
<td>0.4</td>
</tr>
</tbody>
</table>

### 4 Class Problem

<table>
<thead>
<tr>
<th></th>
<th>ssGAM</th>
<th></th>
<th>ssdGAM</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GAM</td>
<td>Best ssGAM</td>
<td>ε</td>
<td>comp</td>
<td>dGAM</td>
<td>Best ssdGAM</td>
<td>ε</td>
</tr>
<tr>
<td>5%</td>
<td>95.6%</td>
<td>95.7%</td>
<td>0.1</td>
<td>16/14</td>
<td>95.6%</td>
<td>95.7%</td>
<td>0.1</td>
</tr>
<tr>
<td>15%</td>
<td>85.4%</td>
<td>85.4%</td>
<td>0.0</td>
<td>26/26</td>
<td>85.1%</td>
<td>85.2%</td>
<td>0.1</td>
</tr>
<tr>
<td>25%</td>
<td>72.8%</td>
<td>72.8%</td>
<td>0.0</td>
<td>57/57</td>
<td>72.9%</td>
<td>72.9%</td>
<td>0.0</td>
</tr>
<tr>
<td>40%</td>
<td>60.5%</td>
<td>60.7%</td>
<td>0.2</td>
<td>47/37</td>
<td>60.9%</td>
<td>61.0%</td>
<td>0.3</td>
</tr>
</tbody>
</table>

### 6 Class Problem

<table>
<thead>
<tr>
<th></th>
<th>ssGAM</th>
<th></th>
<th>ssdGAM</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GAM</td>
<td>Best ssGAM</td>
<td>ε</td>
<td>comp</td>
<td>dGAM</td>
<td>Best ssdGAM</td>
<td>ε</td>
</tr>
<tr>
<td>5%</td>
<td>95.0%</td>
<td>95.0%</td>
<td>0.0</td>
<td>14/14</td>
<td>95.0%</td>
<td>95.2%</td>
<td>0.9</td>
</tr>
<tr>
<td>15%</td>
<td>84.8%</td>
<td>84.8%</td>
<td>0.0</td>
<td>69/69</td>
<td>84.7%</td>
<td>84.8%</td>
<td>0.1</td>
</tr>
<tr>
<td>25%</td>
<td>75.4%</td>
<td>75.5%</td>
<td>0.1</td>
<td>43/52</td>
<td>75.6%</td>
<td>75.6%</td>
<td>0.0</td>
</tr>
<tr>
<td>40%</td>
<td>59.5%</td>
<td>59.5%</td>
<td>0.0</td>
<td>90/90</td>
<td>59.9%</td>
<td>59.9%</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Table 3: Summary of the average performance of all the networks produced on the cross-validation set for every epsilon

<table>
<thead>
<tr>
<th>Epsilon</th>
<th>2 Class Problem</th>
<th>4 Class Problem</th>
<th>6 Class Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>5% overlap</td>
<td>15% overlap</td>
<td>25% overlap</td>
<td>40% overlap</td>
</tr>
<tr>
<td>ssGAM</td>
<td>ssdGAM</td>
<td>ssGAM</td>
<td>ssdGAM</td>
</tr>
<tr>
<td>0.0</td>
<td>93.90</td>
<td>94.48</td>
<td>83.23</td>
</tr>
<tr>
<td>0.1</td>
<td>93.99</td>
<td>94.49</td>
<td>83.53</td>
</tr>
<tr>
<td>0.2</td>
<td>93.82</td>
<td>94.35</td>
<td>82.88</td>
</tr>
<tr>
<td>0.3</td>
<td>92.73</td>
<td>93.36</td>
<td>80.91</td>
</tr>
<tr>
<td>0.4</td>
<td>87.47</td>
<td>88.32</td>
<td>77.06</td>
</tr>
<tr>
<td>0.5</td>
<td>82.13</td>
<td>83.23</td>
<td>72.52</td>
</tr>
<tr>
<td>0.6</td>
<td>79.60</td>
<td>80.52</td>
<td>71.35</td>
</tr>
<tr>
<td>0.7</td>
<td>78.44</td>
<td>79.33</td>
<td>70.17</td>
</tr>
<tr>
<td>0.8</td>
<td>77.62</td>
<td>78.49</td>
<td>69.29</td>
</tr>
<tr>
<td>0.9</td>
<td>77.58</td>
<td>78.43</td>
<td>69.02</td>
</tr>
<tr>
<td>1.0</td>
<td>77.65</td>
<td>78.43</td>
<td>69.04</td>
</tr>
</tbody>
</table>
Comparison with other Networks:

Table 4: Summary of results for the 2 Class Artificial databases

<table>
<thead>
<tr>
<th></th>
<th>SsFAM</th>
<th>ssEAM</th>
<th>ssGAM</th>
<th>ssdGAM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FAM</td>
<td>ssFAM</td>
<td>Comp</td>
<td>EAM</td>
</tr>
<tr>
<td>5%</td>
<td>91.7%</td>
<td>94.9%</td>
<td>0.5+</td>
<td>16/2</td>
</tr>
<tr>
<td>15%</td>
<td>79.3%</td>
<td>84.9%</td>
<td>0.7+</td>
<td>62/4</td>
</tr>
<tr>
<td>25%</td>
<td>64.8%</td>
<td>74.3%</td>
<td>0.6+</td>
<td>89/2</td>
</tr>
<tr>
<td>40%</td>
<td>54%</td>
<td>61.5%</td>
<td>0.6</td>
<td>125/6</td>
</tr>
</tbody>
</table>

Table 5: Summary of results for the 4 Class Artificial databases

<table>
<thead>
<tr>
<th></th>
<th>SsFAM</th>
<th>ssEAM</th>
<th>ssGAM</th>
<th>ssdGAM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FAM</td>
<td>ssFAM</td>
<td>Comp</td>
<td>EAM</td>
</tr>
<tr>
<td>5%</td>
<td>88.8%</td>
<td>92.9%</td>
<td>0.4</td>
<td>16/5</td>
</tr>
<tr>
<td>15%</td>
<td>70.6%</td>
<td>81.4%</td>
<td>0.3</td>
<td>58/11</td>
</tr>
<tr>
<td>25%</td>
<td>56.9%</td>
<td>70.8%</td>
<td>1.0</td>
<td>114/9</td>
</tr>
<tr>
<td>40%</td>
<td>51.0%</td>
<td>58.1%</td>
<td>0.9+</td>
<td>176/18</td>
</tr>
</tbody>
</table>

- - 91
<table>
<thead>
<tr>
<th></th>
<th>ssFAM</th>
<th></th>
<th></th>
<th>ssEAM</th>
<th></th>
<th></th>
<th>ssGAM</th>
<th></th>
<th></th>
<th>ssdGAM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FAM</td>
<td>Best</td>
<td>ε</td>
<td>Comp</td>
<td>EAM</td>
<td>Best</td>
<td>ε</td>
<td>comp</td>
<td>GAM</td>
<td>Best</td>
</tr>
<tr>
<td>5%</td>
<td>80.6%</td>
<td>86.4%</td>
<td>0.1</td>
<td>20/17</td>
<td>92.0%</td>
<td>93.8%</td>
<td>0.3</td>
<td>32/7</td>
<td>94.5%</td>
<td>94.5%</td>
</tr>
<tr>
<td>15%</td>
<td>76.3%</td>
<td>81.0%</td>
<td>0.9</td>
<td>65/55</td>
<td>76.5%</td>
<td>81.8%</td>
<td>0.5</td>
<td>104/6</td>
<td>84.2%</td>
<td>84.2%</td>
</tr>
<tr>
<td>25%</td>
<td>65.7%</td>
<td>69.3%</td>
<td>0.8</td>
<td>105/57</td>
<td>65.0%</td>
<td>66.5%</td>
<td>0.7</td>
<td>147/25</td>
<td>74.0%</td>
<td>73.7%</td>
</tr>
<tr>
<td>40%</td>
<td>51.4%</td>
<td>56.0%</td>
<td>1.0</td>
<td>196/45</td>
<td>49.5%</td>
<td>51.7%</td>
<td>0.8</td>
<td>198/41</td>
<td>59.0%</td>
<td>59.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>94.7%</td>
<td>94.6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>14/14</td>
<td>95.0%</td>
<td>95.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.1</td>
<td>133/29</td>
<td>85.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>51/51</td>
<td>74.2%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>110/110</td>
<td>59.4%</td>
<td>59.4%</td>
</tr>
</tbody>
</table>

Table 6: Summary of results for the 6 Class Artificial databases
In the above tables we compare the performance of the best sGAM, and sdGAM networks with the performance of the best sFAM and sEAM networks. As was mentioned earlier in the thesis FAM and EAM are similar-type of ART networks as GAM, which cover the space of the input patterns with hyperrectangles and hyperellipsoids, respectively. In particular, in the above tables we have chosen as the best sFAM, sEAM, sGAM and sdGAM network as the one that maximizes the performance on the cross-validation set. For this network we are depicting at the above tables its generalization performance on the test set and its size (number of nodes in the F-2^a layer). Obviously higher generalization performance and smaller network size are preferred. The observations that can be extracted from the above tables is that the best sGAM and sdGAM networks achieve better generalization performance than their corresponding sFAM and sEAM counterparts. On the other hand the best sFAM and sEAM network has a smaller size than the best sGAM and sdGAM network.
Table 7: Summary of results for Real databases

<table>
<thead>
<tr>
<th>Real Databases</th>
<th>ssFAM FAM</th>
<th>Best ssFAM</th>
<th>ε</th>
<th>comp</th>
<th>ssEAM EAM</th>
<th>Best ssEAM</th>
<th>ε</th>
<th>comp</th>
<th>ssGAM GAM</th>
<th>Best ssGAM</th>
<th>ε</th>
<th>comp</th>
<th>ssdGAM dGAM</th>
<th>Best ssdGAM</th>
<th>ε</th>
<th>comp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abalone</td>
<td>49.7%</td>
<td>50.2%</td>
<td>0.8</td>
<td>37/19</td>
<td>50.0%</td>
<td>49.7%</td>
<td>0.7+</td>
<td>119/7</td>
<td>50.5%</td>
<td>55.1%</td>
<td>0.5</td>
<td>340/3</td>
<td>51.6%</td>
<td>55.1%</td>
<td>0.5+</td>
<td>96/3</td>
</tr>
<tr>
<td>Pageblocks</td>
<td>90.7%</td>
<td>87.7%</td>
<td>0.2+</td>
<td>116/12</td>
<td>75.7%</td>
<td>89.1%</td>
<td>0.7</td>
<td>133/6</td>
<td>87.8%</td>
<td>87.8%</td>
<td>0.0</td>
<td>17/17</td>
<td>87.8%</td>
<td>87.8%</td>
<td>0.0</td>
<td>17/17</td>
</tr>
<tr>
<td>Iris_DrG</td>
<td>92.8%</td>
<td>94.1%</td>
<td>0.6</td>
<td>38/13</td>
<td>92.8%</td>
<td>94.7%</td>
<td>0.6</td>
<td>35/10</td>
<td>94.8%</td>
<td>95.0%</td>
<td>0.1+</td>
<td>100/231</td>
<td>95.1%</td>
<td>95.1%</td>
<td>0.0+</td>
<td>20/20</td>
</tr>
</tbody>
</table>
**Abalone Database:**

Figure 16: Percentage correct classification on test set for ssFAM, ssEAM, ssGAM and ssdGAM against epsilon on abalone database.

Figure 17: Percentage correct classification on the cross-validation set and number of categories created by the ssGAM networks against Epsilon for abalone database.
From the results obtained on the Abalone database we can observe the following: The generalization of the best GAM networks is better than the generalization of the best FAM and EAM networks for the Abalone database. On the other hand, the number of categories created by the best FAM and EAM networks is smaller than the number of categories created by the GAM networks. Finally, it seems that the utilization of a positive epsilon (semi-supervision) has a positive effect on the GAM networks (despite the fact that it does not improve generalization it significantly reduces the number of categories created).

**Pageblocks Database:**

![Graph](image)

**Figure 18** Percentage correct classification on test set for ssFAM, ssEAM, ssGAM and ssdGAM against epsilon on pageblocks database
For the pageblocks database we observe that the best, in terms of generalization network, is the FAM network, while the best in terms of size (number of categories created) is the EAM network. Furthermore, the generalization performance of the GAM networks is inferior to the generalization performance of the FAM and EAM networks. Finally, the use of semi-supervision (positive epsilon) does not improve the performance of GAM networks.

**Iris Database:**

![Figure 19 Percentage correct classification on test set for ssFAM, ssEAM, ssGAM, ssdGAM against epsilon on Iris database](image-url)
For the Iris database we observe that the generalization of the best FAM, EAM and GAM networks is fairly similar. On the other hand the size of the best networks favors FAM and EAM compared to dGAM and surely compared to GAM (GAM creates too many categories). Once more semi-supervision does not seem to have a positive effect on the GAM networks, while it has a very positive effect on the FAM and EAM networks.
Observations from Artificial Databases:

The following observations are made after closely observing the results obtained from the artificial databases. Some of the observations that are the results of the previous researches done and are proved to be good for FAM and EAM also holds good for GAM and dGAM.

**Observation 1:** Higher overlap data domains are well suited for the semi-supervised learning approach.

![Graph showing percentage correct classification](image)

The downward shift in optimum PCC Train values indicates that for higher overlap in data better performances are achieved at greater training errors.

**Figure 20** Percentage of correct classification on the cross-validation set (PCC XV) versus percentage correct classification on the train set (PCC Train) for ssGAM on 5%, 25% and 40% overlapping data respectively.

Fig. 20 shows the percentage correct classification on the cross-validation set versus the percentage correct classification on the train set for ssGAM on 5%, 25% and 40% overlapping data. This figure holds good even with the ssFAM,
ssEAM and ssdGAM. From Fig. 20 we can see that better performances are achieved at lower training rates, i.e. the performance of the network is more when there are more training errors clearly indicating the issue of over-training. The issue of over-training becomes more pronounced in the case of higher overlapping data. Form Fig. 20 we can say that training the network to perfection affects more on the generalization performance of the higher overlap data than the lower overlap data.

**Observation 2:** *Higher compression ratios are achieved for higher overlap data*

From the tables 2, 3 and 4 we can see that higher compression rates for ssFAM, ssEAM, ssGAM and ssdGAM are achieved at the 40% overlapping data.

**Observation 3:** *The optimum epsilon value for one network is a good epsilon value for the other networks too*

![Figure 21](image.png)  
*Figure 21* Percentage correct classification on cross validation set versus epsilon for ssGAM and ssdGAM
Fig. 16 shows the percentage correct classification on the cross validation set for ssGAM and ssdGAM for the 2-class 25% overlap data. From the figure we can say that the epsilon value, that achieves the best generalization performance for a network is a good epsilon value for the other networks also.

Observations from Real Databases:

The following observations are made from the results obtained from the abalone, pageblocks and iris databases.

Observation 1: High overlap data domains are well suited for the semi-supervised learning approach

Figure 22: Percentage correct classification on training set versus percentage correct classification on cross-validation set for pageblocks and abalone database respectively
This observation that we made with artificial databases holds good even with real databases. The abalone database, which has high overlap shows clearly that high overlap data domains are well suited for semi-supervised learning approach than the pageblocks database, which has minimal data overlap. Fig. 22 clearly shows that high overlap data sets favor higher error tolerances during training.

**Observation 2:** Better compression rates are achieved at higher epsilon values

This observation is based on the table. 5. The positive epsilon value shows a good compression rates. In case of semisupervised learning, better generalized performance is achieved with minimal number of networks

**Observation 3:** Performance gains are universal

![Figure 23: Maximum and average percentage correct classification on cross-validation set against epsilon for the pageblocks and iris databases respectively](image)
The Fig. 23 shows that the peaks of the maximum percent correct classification on cross validation set and the average percent correct classification on the cross validation set against epsilon are the same showing that the performance gains are universal.

**Observation 4:** The optimum epsilon value for one network is a good epsilon value for other networks too

From the results of the real databases shown in table 5 and from the above-mentioned figures we can say that the peak epsilon value for one network is a good epsilon value for other networks too.
10. CONCLUSION

In this thesis, we have focused on two ART networks, the GAM and the dGAM. We have performed a number of experiments with artificial and real databases that allowed us to produce appropriate good default values (vigilance, and initial standard deviation) for the GAM and dGAM networks. This is a valuable discovery because when these networks are solicited to address a classification problem we would prefer to avoid performing extensive experimentation to discover a good solution (i.e., a network of good generalization performance and small size). Furthermore, we have implemented a variation of the GAM and dGAM networks referred to as semi-supervised GAM and dGAM networks. The incorporation of semi-supervision with the ART networks allows ART to code (within one category) patterns of mixed labels. As a result, semi-supervision has the benefit of effectively dealing with noisy and/or overlapping data. Although, semi-supervision has proven to be an effective approach in dealing with noisy and overlapping data for the FAM and EAM networks it did not demonstrate the same benefits for the Gaussian ART networks. One of the reasons for this behavior could be attributed to the fact of how categories are coded with Gaussian networks, where a category calculates the mean and standard deviation of all the patterns that are coded by it, and hence in essence has the ability to deal (somewhat) with noisy and overlapping data. Finally, we compared the classification performance of the best FAM, EAM and GAM networks. Our results here were mixed as well. That is although Gaussian networks exhibited
better (in most instances generalization performance) compared to their FAM, EAM counterparts, the size of the network architectures created favored in most cases FAM and EAM networks.

One research avenue that we are currently pursuing is to design a super-ART architecture that combines the good features of FAM, EAM and GAM. The obvious difficulties is how to appropriately combine the equations of each one of these architectures so that as new data arrive they can have a choice of being coded by a FAM, an EAM or a GAM category, in a way that optimizes the classifier’s performance.
APPENDIX: GAM Notations
x – input pattern

M – Dimensionality of x

I – Input vector applied at the input layer $F_1^a$

O – Output vector.

$F_0^a$ - Pre-processing layer where input pattern $x$ is converted to input vector $l$

$F_1^a$ - Input layer where $l$ is applied

$F_2^a$ - Category representation layer where categories are formed

$F_2^b$ - Output layer where outputs are extracted

$N_a$ - Number of nodes in the category representation layer $F_2^a$

$N_b$ - Number of nodes in the output layer $F_2^b$

$W_{ij}^a$ - Bottom-up weights originating in $F_2^a$ and terminating in $F_2^a$

$w_{ji}^a$ - Top-down weights originating in $F_2^a$ and terminating in $F_1^a$

$W_{jk}^{ab}$ - Inter-ART weights originating in $F_2^a$ and terminating in $F_2^b$

$P(j | x)$ - Posterior probability of cluster $j$ given pattern $x$

$p(x | j)$ - Conditional probability density of $x$ given cluster $j$

$P(j)$ - Priori probability of cluster $j$

$g_j(I^r)$ - Category Choice function of the input $r$

$G_j(I^r)$ - Category match function of the input $r$

$\rho_a$ - Vigilance parameter

$\overline{\rho_a}$ - Baseline vigilance parameter
\( \varepsilon \) - The tunable error tolerance parameter of the network

\( \mu_j \) - Mean of the category \( j \)

\( \sigma_j \) - Standard deviation of the category \( j \)

\( n_j \) - count
REFERENCES


Gomez-Sanchez, E.; Dimitriadis, Y.A.; Cano-Izquierdo, J.M.; Lopez-Coronado, J., “μARTMAP: use of mutual information for category reduction in


