Communicated by Robert Jacobs

A Constructive, Incremental-Learning Network for Mixture Modeling and Classification

James R. Williamson
Center for Adaptive Systems and Department of Cognitive and Neural Systems, Boston University, Boston, MA 02215, U.S.A.

Gaussian ARTMAP (GAM) is a supervised-learning adaptive resonance theory (ART) 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. GAM's representation is a gaussian mixture model of the input space, with learned mappings from the mixture components to output classes. We show a close relationship between GAM and the well-known expectation-maximization (EM) approach to mixture modeling. GAM outperforms an EM classification algorithm on three classification benchmarks, thereby demonstrating the advantage of the ART match criterion for regulating learning and the ARTMAP match tracking operation for incorporating environmental feedback in supervised learning situations.

1 Introduction

Adaptive resonance theory (ART) networks construct stable recognition categories for unsupervised clustering using fast, incremental learning (Carpenter & Grossberg, 1987). The size of clusters coded by ART categories is determined by a global match criterion. ART networks have been extended into supervised-learning ARTMAP networks, which use predictive feedback to regulate the ART clustering mechanism in order to learn multidimensional input-output mappings (Carpenter, Grossberg, & Reynolds, 1991). If an ARTMAP network's prediction is incorrect, then a match tracking signal from the network's output layer raises the match criterion and thus alters clustering in the ART module. In this way, ARTMAP networks realize perhaps the minimal possible extension to ART networks to enable supervised learning while preserving the ART design constraint of fast, incremental learning using only local update rules. In contrast, many online supervised-learning networks, such as multilayer perceptrons and adaptive radial basis function networks, are less local in nature because their gradient-descent learning algorithms require that error signals computed at each of the parameters in the output layer be fed back to each of the parameters in the hidden layer (Rumelhart, Hinton, & Williams, 1986; Poggio & Girosi, 1989).

Copyright © 2001 All Rights Reserved

A new ARTMAP network called gaussian ARTMAP (GAM), which uses internal recognition categories that have gaussian-defined receptive fields, has recently been introduced and applied to several classification problems (Williamson, 1996a, 1996b; Grossberg & Williamson, 1997). GAM's recognition categories learn a gaussian mixture model of the input space as well as mappings to the output classes. When GAM makes an incorrect prediction, its match tracking operation is triggered. The network's vigilance level is raised by adjusting the match criterion, which restricts activation to only those categories that have a sufficiently good match to the input. Match tracking continues until a correct prediction is made, after which the network learns. Thus, match tracking dynamically regulates learning based on predictive feedback. In addition, if no committed categories satisfy the match criterion, a new, uncommitted category is chosen. By this process GAM incrementally constructs a representation of sufficient complexity to solve a classification problem.

GAM is closely related to the expectation-maximization (EM) approach to mixture modeling (Dempster, Laird, & Rubin, 1977). We show that the EM algorithm for unsupervised density estimation using (separable) gaussian mixtures is essentially the same as the GAM equations for modeling the density of its input space, except that GAM is set apart by three features that are standard for ART networks:

1. GAM uses incremental learning, in which the parameters are updated after each input sample, whereas EM uses batch learning, which requires the entire data set. Incremental variants of the EM algorithm will also be discussed.
2. GAM restricts learning of the current data sample to the subset of categories that satisfy its match criterion, whereas EM allows all mixture components to be affected by all data samples.
3. GAM is a constructive network that chooses new, uncommitted categories during training when no committed categories satisfy its match criterion, whereas EM uses a constant, preset number of components.

A straightforward extension of the unsupervised EM mixture modeling algorithm to supervised classification problems involves modeling the class label as a multinomial variable. In this way, the mixture components represent the I $\rightarrow$ O mapping from a real-valued input space to a discrete-valued output space by modeling joint gaussian and multinomial densities in the I/O space (Ghahramani & Jordan, 1994). We show a close relationship between this EM classification algorithm and GAM. However, GAM is set apart by match tracking, which causes GAM to "pay attention" to its training errors and devote more resources to troublesome regions of its I/O space. GAM thereby learns a more effective representation of the I $\rightarrow$ O mapping than EM, as demonstrated by GAM's superior performance to EM on three classification benchmarks.

Copyright © 2001 All Rights Reserved

2 Gaussian ARTMAP

2.1 Category Match and Activation. GAM consists of an input layer, F_1, and an internal category layer, F_2, which receives input from F_1 via adaptive weights. Activations at F_1 and F_2 are denoted, respectively, by $\vec{x} = (x_1, \ldots, x_M)$ and $\vec{y} = (y_1, \ldots, y_N)$ where M is the dimensionality of the input space and N is the current number of committed F_2 category nodes. Each F_2 category, j, models a local density of the input space with a separable gaussian receptive field and maps to an output class prediction. The category's receptive field is defined with a separable gaussian distribution parameterized by two M-dimensional vectors: its mean, $\vec{\mu}_j$, and standard deviation, $\vec{\sigma}_j$. A scalar, n_j, also represents the amount of training data for which the node has received credit. Category j is activated only if its match, G_j, satisfies the match criterion, which is determined by a vigilance parameter, ρ. Match is a measure, obtained from the category's unit-height gaussian distribution, of how close an input is to the category's mean, relative to its standard deviation:

$$G_j = \exp\left(-\frac{1}{2}\sum_{i=1}^{M}\left(\frac{x_i - \mu_{ji}}{\sigma_{ji}}\right)^2\right). \tag{2.1}$$

The match criterion is a threshold: the category is activated only if $G_j > \rho$; otherwise, the category is reset. 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 $(\prod_{i=1}^{M} \sigma_{ji})^{-1}$, which normalizes its gaussian distribution:

$$g_j = \frac{n_j}{\prod_{i=1}^{M} \sigma_{ji}} G_j \text{ if } G_j > \rho; \qquad g_j = 0 \text{ otherwise.} \tag{2.2}$$

The category's activation, y_j, represents its conditional probability for being the "source" of the input vector: $P(j \mid \vec{x})$. This is obtained by normalizing the category's input strength,

$$y_j = \frac{g_j}{\sum_{l=1}^{N} g_l}. \tag{2.3}$$

As originally proposed, GAM used a choice activation rule at F_2: $y_j = 1$ if $g_j > g_l \ \forall\, l \neq j$; $y_j = 0$ otherwise (Williamson, 1996a). In this version, only a single, chosen category learned on each trial. Here, we describe a distributed-learning version of GAM, which uses the distributed F_2 activation equation (2.3). Distributed GAM was introduced in Williamson (1996b), where it was shown to obtain 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

Copyright © 2001 All Rights Reserved

image classification system that use rule-based, multilayer perceptron, and k-nearest neighbor classifiers (Grossberg & Williamson, 1997).

2.2 Prediction and Match Tracking. Equations 2.1 through 2.3 describe activation of category nodes in an unsupervised-learning gaussian ART module. The following equations describe GAM's supervised-learning mechanism, which incorporates feedback from class predictions made by the F_2 category nodes and thus turns gaussian ART into gaussian ARTMAP. When a category, j, is first chosen, it learns a permanent mapping to the output class, k, associated with the current training sample. All categories that map to the same class prediction belong to the same ensemble: $j \in E(k)$. Each time an input is presented, the categories in each ensemble sum their activations to generate a net probability estimate, z_k, of the class prediction k that they share:

$$z_k = \sum_{j \in E(k)} y_j. \tag{2.4}$$

The system prediction, K, is obtained from the maximum probability estimate,

$$K = \arg\max_k (z_k), \tag{2.5}$$

which also determines the chosen ensemble. On real-world problems, the probability estimate z_K has been found to predict accurately the probability that prediction K is correct (Grossberg & Williamson, 1997). Note that category j's initial activation, y_j, represents $P(j \mid \vec{x})$. Once the class prediction K is chosen, we obtain the category's "chosen-ensemble" activation, y_j^*, which represents $P(j \mid \vec{x}, K)$:

$$y_j^* = \frac{y_j}{\sum_{l \in E(K)} y_l} \quad \text{if } j \in E(K); \qquad y_j^* = 0 \ \text{ otherwise.} \tag{2.6}$$

If K is the correct prediction, then the network resonates and learns the current input. If K is incorrect, then match tracking is invoked. As originally conceived, match tracking involves raising ρ continuously, causing categories j, such that $G_j \leq \rho$, to be reset until the correct prediction is finally selected (Carpenter, Grossberg, & Rosen, 1991). Because GAM uses a distributed representation at F_2, each z_k may be determined by multiple categories, according to equation 2.6. Therefore, it is difficult to determine numerically how much ρ needs to be raised in order to select a different prediction. It is inefficient (on a conventional computer) to determine the exact amount to raise ρ by repeatedly resetting the active category with the lowest match value G_j, each time reevaluating equations 2.3, 2.4, and 2.5, until a new prediction is finally selected.

Copyright © 2001 All Rights Reserved

Instead, a one-shot match tracking algorithm has been developed for GAM and used successfully on several classification problems (Williamson, 1996b; Grossberg & Williamson, 1997). This algorithm involves raising ρ to the average match value of the chosen ensemble:

$$\rho = \exp\left(-\frac{1}{2}\sum_{j\in E(K)} y_j^* \sum_{i=1}^{M}\left(\frac{x_i-\mu_{ji}}{\sigma_{ji}}\right)^2\right). \tag{2.7}$$

In addition, all categories in the chosen ensemble are reset: $g_j = 0 \ \forall\ j \in E(K)$. Equations 2.2 through 2.5 are then reevaluated. Based on the remaining non-reset categories, a new prediction K in equation 2.5, and its corresponding ensemble, is chosen. This automatic search cycle continues until the correct prediction is made or until all committed categories are reset, $G_j \leq \rho \ \forall\ j \in \{1,\dots,N\}$, and an uncommitted category is chosen. Match tracking ensures that the correct prediction comes from an ensemble with a better match to the training sample than all reset ensembles. Upon presentation of the next training sample, ρ is reassigned its baseline value: $\rho = \bar{\rho}$.

2.3 Learning. The F_2 parameters $\vec{\mu}_j$ and $\vec{\sigma}_j$ are updated to represent the sample statistics of the input using local learning rules that are related to the instar, or gated steepest descent, learning rule (Grossberg, 1976). Instar learning is an associative rule in which the postsynaptic activity y_j^* modulates the rate at which the weight w_{ji} tracks the presynaptic signal $f(x_i)$,

$$\epsilon\frac{d}{dt}w_{ji} = y_j^*[f(x_i) - w_{ji}]. \tag{2.8}$$

The discrete-time version of equation 2.8 is

$$w_{ji} := (1-\epsilon^{-1}y_j^*)w_{ji} + \epsilon^{-1}y_j^* f(x_i). \tag{2.9}$$

GAM's learning equations are obtained by modifying equation 2.9. The rate constant ϵ is replaced by n_j, which is incremented to represent the cumulative chosen-ensemble activation of node j and thus the amount of training data the node has been assigned credit for:

$$n_j := n_j + y_j^*. \tag{2.10}$$

Modulation of learning by n_j causes the inputs to be weighted equally over time, so that their sample statistics are learned. The presynaptic term $f(x_i)$ is set to x_i and x_i^2, respectively, for learning the first and second moments of the input. The standard deviation is then derived from these statistics:

$$\mu_{ji} := (1-y_j^* n_j^{-1})\mu_{ji} + y_j^* n_j^{-1} x_i, \tag{2.11}$$

Copyright © 2001 All Rights Reserved

$$\nu_{ji} := (1 - y_j^* n_j^{-1})\nu_{ji} + y_j^* n_j^{-1} x_i^2, \tag{2.12}$$

$$\sigma_{ji} = \sqrt{\nu_{ji} - \mu_{ji}^2}. \tag{2.13}$$

In Williamson (1996a, 1996b) and Grossberg and Williamson (1997), σ_{ji}, rather than ν_{ji}, is incrementally updated via:

$$\sigma_{ji} := \sqrt{(1 - y_j n_j^{-1})\sigma_{ji}^2 + y_j n_j^{-1}(x_i - \mu_{ji})^2}. \tag{2.14}$$

Unlike equations 2.12 and 2.13, equation 2.14 biases the estimate of σ_{ji} because the incremental updates are based on current estimates of μ_{ji}, which vary over time. This bias appears to be insignificant, however, as our simulations have not revealed a significant advantage for either method. Equations 2.12 and 2.13 are used here solely because they describe a simpler learning rule.

GAM is initialized with $N = 0$. When an uncommitted category is chosen, N is incremented, and the new category, indexed by N, is initialized with $y_N^* = 1$ and $n_N = 0$, and with a permanent mapping to the correct output class. Learning then proceeds via equations 2.10 through 2.13, with one modification: a constant, γ^2, is added to ν_{Ni} in equation 2.12, which yields $\sigma_{Ni} = \gamma$ in equation 2.13. Initializing categories with this nonzero standard deviation is necessary to make equation 2.1 and equation 2.2 well defined. Varying γ has a marked effect on learning: as γ is raised, learning becomes slower, but fewer categories are created. Generally, γ is much larger than the final standard deviation that a category converges to. Intuitively, a large γ represents a low level of certainty for, and commitment to, the location in the input space coded by a new category. As γ is raised, the network settles into its input space representation in a slower and more graceful way. Note that best results have generally been obtained by preprocessing the set of input vectors to have the same standard deviation in each dimension, so that γ has the same meaning in all the dimensions.

3 Expectation-Maximization

Now we show the relationship between GAM and the EM approach to mixture modeling. EM is a general iterative optimization technique for obtaining maximum likelihood estimates of observed data that are in some way incomplete (Dempster et al., 1977). Each iteration of EM consists of an expectation step (E-step) followed by a maximization step (M-step). We start with an "incomplete-data" likelihood function of the model given the data and then posit a "complete-data" likelihood function, which is much easier to maximize but depends on unknown, missing data. The E-step finds the expectation of the complete-data likelihood function, yielding a deterministic function. The M-step then updates the system parameters to

Copyright © 2001 All Rights Reserved

maximize this function. Dempster et al. (1977) proved that each iteration of EM yields an increase in the incomplete-data likelihood until a local maximum is reached.

3.1 Gaussian Mixture Modeling. First, let us consider density estimation of the input space using (separable) gaussian mixtures. We model the training set, $X = \{\vec{x}_t\}_{t=1}^{T}$, as comprising independent, identically distributed samples generated from a mixture density, which is parameterized by $\Theta = \{\alpha_j, \vec{\theta}_j\}_{j=1}^{N}$. The incomplete-data density of X given Θ is

$$P(X|\Theta) = \prod_{t=1}^{T} P(\vec{x}_t|\Theta) = \prod_{t=1}^{T}\sum_{j=1}^{N} \alpha_j P(\vec{x}_t|\vec{\theta}_j), \tag{3.1}$$

where $\vec{\theta}_j$ parameterizes the distribution of the jth component, and α_j represents its a priori probability, or mixture proportion: $\alpha_j \geq 0$ and $\sum_{j=1}^{N} \alpha_j = 1$. The incomplete-data log likelihood of Θ given X is

$$l(\Theta|X) = \sum_{t=1}^{T} \log \sum_{j=1}^{N} \alpha_j P(\vec{x}_t|\vec{\theta}_j), \tag{3.2}$$

which is difficult to maximize because it includes the log of a sum. Intuitively, equation 3.2 contains a credit assignment problem, because it is not clear which component generated each data sample. To get around this problem, we introduce "missing data" in the form of a set of indicator variables, $Z = \{\vec{z}_t\}_{t=1}^{T}$, such that $z_{tj} = 1$ if component j generated sample t and $z_{tj} = 0$ otherwise. Now, using the complete data, $\{X, Z\}$, we can explicitly assign credit and thus decouple the overall maximization problem into a set of simple maximizations by defining a complete-data density function,

$$P(X, Z|\Theta) = \prod_{t=1}^{T}\prod_{j=1}^{N} [\alpha_j P(\vec{x}_t|\vec{\theta}_j)]^{z_{tj}}, \tag{3.3}$$

from which we obtain a complete-data log likelihood,

$$l_c(\Theta|X, Z) = \sum_{t=1}^{T}\sum_{j=1}^{N} z_{tj} \log[\alpha_j P(\vec{x}_t|\vec{\theta}_j)], \tag{3.4}$$

which does not include a log of a sum. However, note that $l_c(\Theta|X, Z)$ is a random variable because the missing variables Z are unknown. Therefore, the EM algorithm finds the expected value of $l_c(\Theta|X, Z)$ in the E-step:

$$Q(\Theta|\Theta^{(p)}) = E[l_c(\Theta|X, Z)|X, \Theta^{(p)}], \tag{3.5}$$

Copyright © 2001 All Rights Reserved

where $\Theta^{(p)}$ is the set of parameters at the pth iteration. The E-step yields a deterministic function $Q(\Theta|\Theta^{(p)})$, and the M-step then maximizes this function with respect to Θ to obtain new parameters, $\Theta^{(p+1)}$:

$$\Theta^{(p+1)} = \arg\max_{\Theta} Q(\Theta|\Theta^{(p)}). \tag{3.6}$$

Dempster et al. (1977) proved that each iteration of EM yields an increase in the incomplete-data likelihood until a local maximum is reached:

$$l(\Theta^{(p+1)}|X) \geq l(\Theta^{(p)}|X). \tag{3.7}$$

The E-step in equation 3.7 simplifies to computing (for all t, j) $y_{tj}^{(p)} = E[z_{tj}|\vec{x}_t, \Theta]$, the probability that component j generated sample t. For comparison with GAM, we define the density $P(\vec{x}_t|\vec{\theta}_j)$ as a separable gaussian distribution, yielding

$$y_{tj}^{(p)} = \frac{\alpha_j^{(p)} \left(\prod_{i=1}^{M} \sigma_{ji}^{(p)}\right)^{-1} \exp\left(-\frac{1}{2}\sum_{i=1}^{M}\left(\frac{x_{ti}-\mu_{ji}^{(p)}}{\sigma_{ji}^{(p)}}\right)^2\right)}{\sum_{l=1}^{N}\left[\alpha_l^{(p)} \left(\prod_{i=1}^{M} \sigma_{li}^{(p)}\right)^{-1} \exp\left(-\frac{1}{2}\sum_{i=1}^{M}\left(\frac{x_{ti}-\mu_{li}^{(p)}}{\sigma_{li}^{(p)}}\right)^2\right)\right]}. \tag{3.8}$$

For distributions in the exponential family, the M-step simply updates the model parameters based on their reestimated sufficient statistics, which are computed in a batch procedure that weights each sample by its probability, $y_{tj}^{(p)}$,

$$\alpha_j^{(p+1)} = \frac{1}{T}\sum_{t=1}^{T} y_{tj}^{(p)}, \tag{3.9}$$

$$\mu_{ji}^{(p+1)} = \frac{\sum_{t=1}^{T} y_{tj}^{(p)} x_{ti}}{\sum_{t=1}^{T} y_{tj}^{(p)}}, \tag{3.10}$$

$$\sigma_{ji}^{(p+1)} = \sqrt{\frac{\sum_{t=1}^{T} y_{tj}^{(p)} x_{ti}^2}{\sum_{t=1}^{T} y_{tj}^{(p)}} - \left(\mu_{ji}^{(p+1)}\right)^2}. \tag{3.11}$$

Note that $y_{tj}^{(p)}$ in equation 3.8 is equivalent to GAM's category activation term y_j in equation 2.3 provided that vigilance is zero ($\rho = 0$). Also, the EM parameter reestimation equations (3.9 through 3.11) are essentially the same as GAM's learning equations (2.10 through 2.13), except that EM uses batch learning with a constant number of components, while GAM uses incremental learning, updating the parameters after each input sample, and recruiting new categories as needed.

Copyright © 2001 All Rights Reserved

3.2 Extension to Classification. The EM mixture modeling algorithm is extended to classification problems by modeling the class label as a multinomial variable (Ghahramani & Jordan, 1994). Therefore, each mixture component consists of a gaussian distribution for the "input" features and a multinomial distribution for the "output" class labels. Thus, the classification problem is cast as a density estimation problem, in which the mixture components represent the joint density of the input-output mapping. Specifically, the joint probability that the tth sample has input features $\vec{x}_t$ and output class $k(t)$ is denoted by

$$\begin{aligned} P(\vec{x}_t, K = k(t)|\Theta) &= \sum_{j=1}^{N} \alpha_j P(\vec{x}, K = k(t)|\vec{\theta}_j, \vec{\lambda}_j) \\ &= \sum_{j=1}^{N} \lambda_{jk(t)} \alpha_j P(\vec{x}|\vec{\theta}_j), \end{aligned} \tag{3.12}$$

where the multinomial distribution is parameterized by $\lambda_{jk} = P(K = k|j; \vec{\theta}_j)$ and $\sum_k \lambda_{jk} = 1$. This classification algorithm is trained the same way as the gaussian mixture algorithm, except that equation 3.8 becomes:

$$y_{tj}^{(p)} = \frac{\lambda_{jk(t)}^{(p)} \alpha_j^{(p)} \left(\prod_{i=1}^{M} \sigma_{ji}^{(p)}\right)^{-1} \exp\left(-\frac{1}{2}\sum_{i=1}^{M}\left(\frac{x_{ti}-\mu_{ji}^{(p)}}{\sigma_{ji}^{(p)}}\right)^2\right)}{\sum_{l=1}^{N}\left[\lambda_{lk(t)}^{(p)} \alpha_l^{(p)} \left(\prod_{i=1}^{M} \sigma_{li}^{(p)}\right)^{-1} \exp\left(-\frac{1}{2}\sum_{i=1}^{M}\left(\frac{x_{ti}-\mu_{li}^{(p)}}{\sigma_{li}^{(p)}}\right)^2\right)\right]}, \tag{3.13}$$

and the multinomial parameters are updated via

$$\lambda_{jk}^{(p+1)} = \frac{\sum_{t=1}^{T} y_{tj}^{(p)} \delta[k - k(t)]}{\sum_{t=1}^{T} y_{tj}^{(p)}}, \tag{3.14}$$

where $\delta[\omega] = 1$ if $\omega = 0$ and $\delta[\omega] = 0$ if $\omega \neq 0$. During testing, the class label is missing and its expected value is "filled in" to determine the system prediction:

$$K = \arg\max_k \left(\sum_{j=1}^{N} y_{tj}\lambda_{jk}\right), \tag{3.15}$$

where y_{tj} is computed via equation 3.8. The parameter λ_{jk} plays an analogous role to GAM's membership function, $j \in E(k)$. GAM's equation (2.5) performs the same computation as equation 3.15, provided that $\lambda_{jk} = 1$ if $j \in E(k)$ and $\lambda_{jk} = 0$ otherwise. Note that if each EM component is initialized

Copyright © 2001 All Rights Reserved

so that $\lambda_{jk} = 1$ for some k, then λ_{jk} will never change according to update equation 3.14. Therefore, with this restriction, along with the restriction that vigilance is always zero ($\rho \equiv 0$), EM becomes a batch-learning version of GAM.

Thus, GAM and EM use similar learning equations and obtain a similar final representation: a gaussian mixture model, with mappings from the mixture components to class labels. However, the learning dynamics of the two algorithms are quite different due to GAM's match tracking operation. The EM algorithm is a variable metric gradient ascent algorithm, in which each step in parameter space is related to the gradient of the log likelihood of the mixture model (Xu & Jordan, 1996). With each step, the likelihood of the I/O density estimate increases until a local maximum is reached. In other words, the parameterization at each step is represented by a point in the parameter space, which has a constant dimensionality. The system is initialized at some point in this parameter space, and the point moves with each training epoch, based on the gradient of the log likelihood, until a local maximum of the likelihood is reached.

GAM's parameters are updated using an incremental approximation to the batch-learning EM algorithm. However, the most important respect in which GAM's learning procedure differs from that of EM is that the former uses predictive feedback via the match tracking process. When errors occur in the $I \rightarrow O$ mapping, match tracking reduces, by varying amounts, the number of categories that learn and thus restricts the movement of GAM's parameterization to a parameter subspace. Match tracking also causes uncommitted categories to be chosen, which expands the dimensionality of the parameter space. Newly committed categories have small a priori probabilities and large standard deviations, and thus a weak but ubiquitous influence on the gradient.

3.3 Incremental Variants of EM. One of the practical advantages of GAM over the standard EM algorithm described in sections 3.1 and 3.2 is that the former learns incrementally whereas the latter learns using a batch method. However, incremental variants of EM have also been developed. Most notably, Neal and Hinton (1993) showed that EM can incrementally update the model parameters if a separate set of sufficient statistics is stored *for each* input sample. That is, a separate set of the statistics computed in equations 3.9 through 3.11 and 3.14, corresponding to each input sample, can be saved. In this way, the E-step and M-step can be computed following the presentation of each sample, with the maximization affecting only the statistics associated with the current sample. Because this incremental EM recomputes expectations and maximizations following each input sample, it incorporates new information immediately into its parameter updates and thereby converges more quickly than the standard EM batch algorithm.

Incremental EM illustrates the statistics that need to be maintained in order to ensure monotonic convergence of the model likelihood in an online

Copyright © 2001 All Rights Reserved

setting. However, the need to store separate statistics for each input sample makes incremental EM extremely nonlocal and, moreover, quite impractical for use on large data sets. On the other hand, there also exist incremental approximations of EM that use local learning but do not guarantee monotonic convergence of the model likelihood. For example, Hinton and Nowlan (1990) used an incremental equation for updating variance estimates that is identical to equation 2.14, except that a constant learning rate coefficient was used rather than the decaying term, n_j^{-1}.

GAM differs from standard EM due to both GAM's match tracking procedure and its incremental approximation to EM's learning equations. To make comparisons of GAM and EM on real-world classification problems more informative, the effects of each of these differences should be isolated. Therefore, in the following section GAM is compared to an incremental EM approximation as well as to the standard EM algorithm. Furthermore, in order to isolate the role played by match tracking, we use GAM's learning equations as the incremental EM approximation.

4 Simulations: Comparisons of GAM and EM

4.1 Methodology. All three classification tasks are evaluated using the same procedure. The data sets are normalized to have unit variance in each dimension. EM is tested with several different N. For each setting of N, EM is trained five times with different initializations, and the five test results are averaged. EM's performance often peaks quickly and then declines due to overfitting, particularly when N is large. Therefore, EM's performance is plotted following two training epochs, when its best performance is generally obtained (for large N), and also following equilibration. GAM uses $\overline{\rho} \approx 0$ (precisely, $\overline{\rho} = 10^{-7M}$) for all simulations, and γ is varied. For each setting of γ, GAM is trained five times with different random orderings of the data, and the data order is also scrambled between each training epoch. GAM is trained for 100 epochs, and the test results are plotted after each epoch. As the results below illustrate, GAM often begins with relatively poor performance for the first few training epochs (particularly when γ is large because it biases the initial category standard deviations), after which performance improves. Performance sometimes peaks and then declines due to overfitting. To convey a full picture of GAM's behavior, GAM's average performance is plotted for each of its 100 training epochs and for each setting of γ.

Several initialization procedures for EM were evaluated and found to produce widely different results. The most successful of these is reported here. As it happens, this procedure initializes EM components in essentially the same way that GAM categories are initialized, except that the former are all initialized prior to training. Specifically, each mixture component is assigned to one of N randomly selected samples, denoted by $\{\vec{x}_t, k(t)\}_{t=1}^{N}$,

Copyright © 2001 All Rights Reserved

and initialized as follows: $\alpha_j = 1/N$, $\mu_{ji} = x_{ti}$, $\sigma_{ji} = \gamma$, and $\lambda_{jk} = \delta[k - k(t)]$. In addition, it is guaranteed that at least one component maps to each of the output classes. Because each EM component maps only to a single output class, EM and GAM use the same representation and thus have the same storage requirement for a given N.

It is fortuitous that EM's best initialization procedure corresponds so closely to that of GAM. This makes the comparison between the two algorithms more revealing because it isolates the role played by match tracking. Apart from their batch–incremental-learning distinction, this EM algorithm operates identically to a GAM network that has a constant $\rho \equiv 0$. This is because EM equation 3.13, in which "feedback" from the class label directly "activates" the mixture components, is functionally equivalent to the GAM process of resetting all ensemble categories that make a wrong prediction, and finally basing learning on the chosen-ensemble activations in equation 2.6 that correspond to the correct prediction. Thus, GAM is set apart only by match tracking, which raises ρ according to equation 2.7 when an incorrect prediction is made.

The contribution of match tracking can be further isolated by removing the batch–incremental-learning distinction. This is done by using a static-GAM (S-GAM) network, which is identical to GAM except that it has a fixed vigilance ($\rho \equiv \overline{\rho}$), which prevents S-GAM from committing new categories during training because the baseline vigilance, $\overline{\rho} = 10^{-7M}$, is too small to reset any of the committed categories. Therefore, S-GAM needs to be initialized with a set of N categories prior to training. S-GAM is initialized the same way as EM (described above), with each of the N categories assigned to one of N randomly selected training samples and with an ensemble that maps to each of the output classes. If S-GAM makes an incorrect prediction during training, then the chosen ensemble *is* reset, but vigilance *is not* raised. Because vigilance is never raised, match tracking has no effect on learning other than to ensure that the correct prediction is made before learning occurs. Therefore, S-GAM's procedure is functionally identical to the EM procedure of directly activating categories based on both the input and the supervised class label. By including comparisons to S-GAM, therefore, the effects of match tracking alone (GAM versus S-GAM), the effects of incremental learning alone (S-GAM versus EM), and the effects of match tracking and incremental learning together (GAM versus EM) are revealed.

4.2 Letter Image Recognition. EM, GAM, and S-GAM are first evaluated on a letter image recognition task developed in Frey and Slate (1991). The data set, which is archived in the UCI machine learning repository (King, 1992), consists of 16-dimensional vectors derived from machine-generated images of alphabetical characters (A to Z). The classification problem is to predict the correct letter from the 16 features. Classification difficulty stems from the fact that the characters are generated from 20 different fonts, are randomly warped, and only simple features such as the

Copyright © 2001 All Rights Reserved

Table 1: Letter Image Classification.

Algorithm	Error Rate (%)
k-NN	4.4
HAC (a)	19.2
HAC (b)	18.4
HAC (c)	17.3
GAM-CL ($\gamma = 2$)	6.0
GAM-CL ($\gamma = 4$)	6.3
FAM ($\alpha = 1.0$)	8.1
FAM ($\alpha = 0.1$)	13.5

Source: Results are adapted from Frey and Slate (1991) and Williamson (1996a).
Notes: k-NN = nearest-neighbor classifier ($k = 1$). HAC = Holland-style adaptive classifier. Results of three HAC variations are shown here (see Frey & Slate, 1991). GAM-CL = Gaussian ARTMAP with choice learning. FAM = Fuzzy ARTMAP.

total number of "on" pixels and the size and position of a box around the "on" pixels are used. The data set consists of 20,000 samples, the first 16,000 of which are used for training and the last 4000 for testing. For comparison, the results of several other classifiers are shown in Table 1 (see Frey & Slate, 1991; Williamson, 1996a).

Figure 1 shows the classification results of EM and GAM on the letter image recognition problem. EM's average error rate is plotted as a function of N ($N = 100, 200, \ldots, 1000$). For each N, the error rate is shown after two training epochs (solid line) and after EM has equilibrated (dashed line). Note that with $N < 600$, EM performs better following equilibration, but with $N > 600$, EM performs better following two epochs, and then performance declines, presumably due to overfitting. EM's best performance (7.4 ± 0.2 percent error) is obtained with $N = 1000$ following two training epochs.

GAM's average error rate is plotted for $\gamma = 1, 2, 4$. Each point along one of GAM's error curves corresponds to a different training epoch, with the first epoch plotted at the left-most point of a curve. As training proceeds, the number of categories increases, and the error rate decreases. After a certain point, the error rate may increase again due to overfitting. As γ is raised, the error curves shift to the left. The initial performance becomes progressively worse, and it takes longer for GAM's performance to peak. However, fewer categories are created, and there is less degradation due to

Copyright © 2001 All Rights Reserved

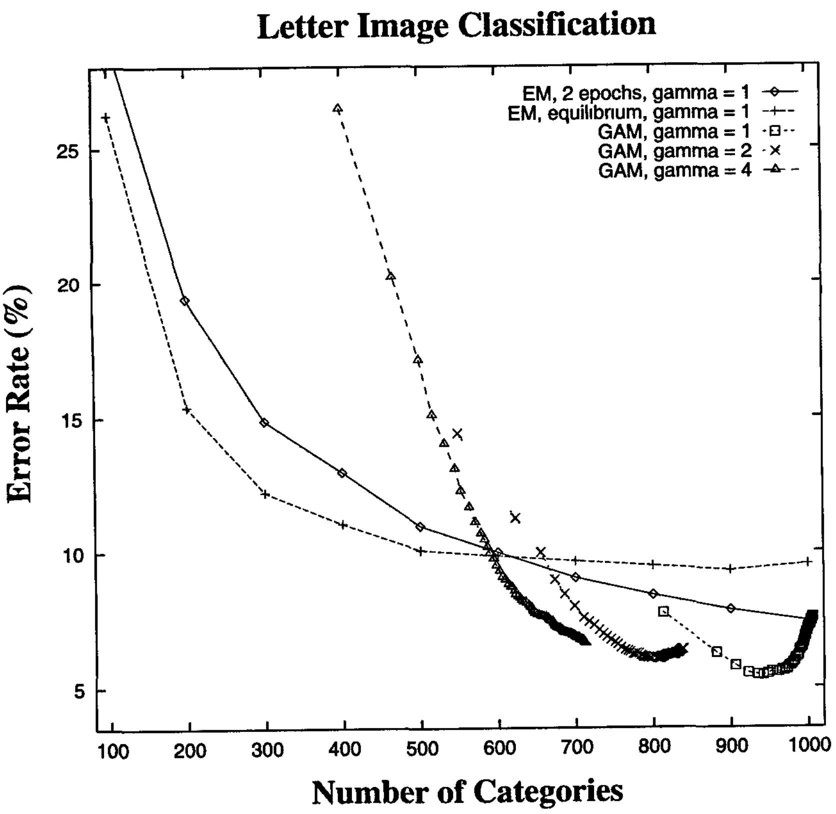

Figure 1: The average error rates of EM and GAM on the letter image classification problem plotted as a function of the number of categories, N. EM's error rates are shown after two training epochs and after equilibration. EM is trained with $N = 100, 200, \ldots, 1000$. GAM is trained with $\gamma = 1, 2, 4$. GAM's error rates are plotted after each training epoch. Each of GAM's error curves corresponds to a different value of γ, with the left-most point on a curve corresponding to the first epoch. From left to right along a curve, each successive point corresponds to a successive training epoch.

overfitting. GAM's best performance (5.4 ± 0.3 percent error) is obtained with $\gamma = 1$ following six training epochs.

For all settings of γ, GAM achieves lower error rates than EM for all N. As γ is raised, GAM requires more training epochs to surpass EM. However, EM does achieve reasonably low error rates with N much smaller than that created by GAM for any value of γ. Figure 1 also suggests a general pattern

Copyright © 2001 All Rights Reserved

Table 2: The Trade-Offs Entailed by the Choice of γ for GAM for Three Classification Problems.

γ	Number of Epochs	Error Rate (%)	Number of Categories	Storage Rate[a] (%)
a.	Letter image classification			
1	1	7.7	812.8	10.5
1	6	5.4	941.2	12.1
2	38	6.0	807.2	10.4
4	100	6.6	712.4	9.2
8	100	8.8	593.8	7.7
b.	Satellite image classification			
1	1	12 4	125.0	5.7
1	100	10.0	255.2	11.7
2	100	11.2	212.0	9.7
4	100	12.2	149.2	6.8
8	100	13.9	93.4	4.3
c.	Spoken vowel classification			
1	1	49.7	72.4	28.8
2	5	48.7	62.2	24.7
4	6	44.0	53.8	21.4
8	19	43.9	46.6	18.5

Notes: The lowest error rates (averaged over five runs) obtained for each of four settings of γ ($\gamma = 1, 2, 4, 8$) are shown. The error rate obtained with $\gamma = 1$ after only one training epoch is also shown to illustrate GAM's fast-learning capability.

[a]The amount of storage used by GAM divided by the amount used by the training set.

in GAM's performance: in a plot of error rate as a function of number of categories, there exists (roughly) a U-shaped envelope. For different values of γ, GAM approaches and then follows a different portion of that envelope. As the remaining simulations illustrate, however, the relationship between γ and the placement of this envelope varies between different tasks. To illustrate further the trade-offs entailed by the choice of γ, Table 2a lists the lowest error rate obtained on this problem for different values of γ, along with the number of training epochs used to reach that point and the number of categories created. Finally, the storage rate, which is the amount of storage used by GAM relative to that used by the training set (and hence, by a nearest-neighbor classifier), is listed. For an M-dimensional input, each GAM category stores $2M + 1$ values, so the storage rate is calculated as:

$$\frac{N(2M+1)}{TM}.$$

In Figure 2, the error rates are plotted as a function of the number of training epochs. GAM's results are shown given its best parameter setting

Copyright © 2001 All Rights Reserved

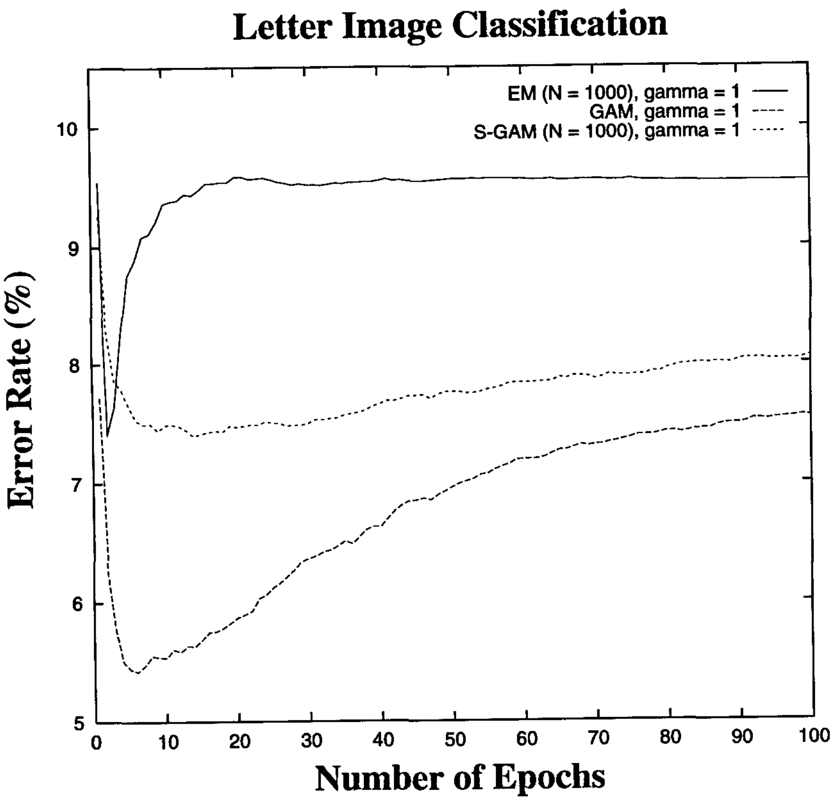

Figure 2: Average error rates of EM, GAM, and S-GAM plotted as a function of the number of training epochs.

($\gamma = 1$), and the results of EM and S-GAM are shown given similar parameters ($\gamma = 1, N = 1000$). GAM quickly achieves its best performance at six epochs, after which performance slowly degrades due to overfitting. EM quickly achieves its best performance at two epochs, after which performance quickly degrades and then stabilizes. S-GAM's performance improves more slowly than EM, but S-GAM eventually obtains lower error rates than EM. For other values of N, the relative performance of EM and S-GAM is similar to that shown in Figure 2. Therefore, on this problem S-GAM's incremental approximation to EM's batch learning seems to confer a small advantage. Much of this advantage is probably due to the fact that S-GAM's standard deviations decrease less than EM's due to the lingering effect of γ. Additional simulations have shown that EM suffers less from

Copyright © 2001 All Rights Reserved

overfitting if the shrinkage of its standard deviations is attenuated during learning, although this variation does not appear to improve its best results.

Figures 1 and 2 show that match tracking causes GAM to construct an appropriate number of categories to support the I $\rightarrow$ O mapping and to learn the mapping more accurately than EM or S-GAM. However, these results do not indicate why this is the case. There are two possible explanations for why match tracking gives GAM an advantage: (1) match tracking causes GAM to obtain a better estimate of the I/O density (i.e., to obtain a mixture model with a higher likelihood) (2) match tracking biases GAM's density estimate so as to reduce its predictive error in the I $\rightarrow$ O mapping. The results shown in Figures 3 and 4 indicate that the latter explanation is correct.

Figure 3 shows the error rate over 25 training epochs for a single run of GAM, on which GAM created 985 categories: the error rate on the training set (top) and the test set (bottom). Figure 3 also shows the corresponding error rates for EM and S-GAM initialized with the same number ($N = 985$) of categories as were created by GAM. On both the training set and the test set, GAM obtains much lower error rates than EM and S-GAM. Next, Figure 4 shows the log likelihoods of the mixture models formed by EM, GAM, and S-GAM. EM obtains a much higher log likelihood than GAM and S-GAM on both the training set and the test set, whereas the log likelihoods obtained by GAM and S-GAM are similar. Therefore, GAM outperforms EM and S-GAM because match tracking biases its density estimate such that the predictive error in its I $\rightarrow$ O mapping is reduced, despite the fact that GAM obtains an estimate of the I/O density with a lower likelihood than that of EM.

4.3 Landsat Satellite Image Segmentation. EM, GAM, and S-GAM are evaluated on a real-world task: segmentation of a Landsat satellite image (Feng, Sutherland, King, Muggleton, & Henery, 1993). The data set, which is archived in the UCI machine learning repository (King, 1992), consists of multispectral values within nonoverlapping 3×3 pixel neighborhoods in an image obtained from a Landsat satellite multispectral scanner. At each pixel are four values, corresponding to four spectral bands. Two of these are in the visible region (corresponding approximately to green and red regions of the visible spectrum) and two are in the (near) infrared. The input space is thus 36-dimensional (9 pixels and 4 values per pixel). The spatial resolution of a pixel is about $80\,\mathrm{m} \times 80\,\mathrm{m}$. The center pixel of each neighborhood is associated with one of six vegetation classes: red soil, cotton crop, gray soil, damp gray soil, soil with vegetation stubble, and very damp gray soil. The data set is partitioned into 4435 samples for training, and 2000 samples for testing. For comparison, the results of several other classifiers are shown in Table 3 (see Feng et al., 1993; Asfour, Carpenter, & Grossberg, 1995).

Figure 5 shows the classification results of EM and GAM on the satellite image segmentation problem. EM's error rate is plotted for $N = 25, 50, \ldots,$

Copyright © 2001 All Rights Reserved

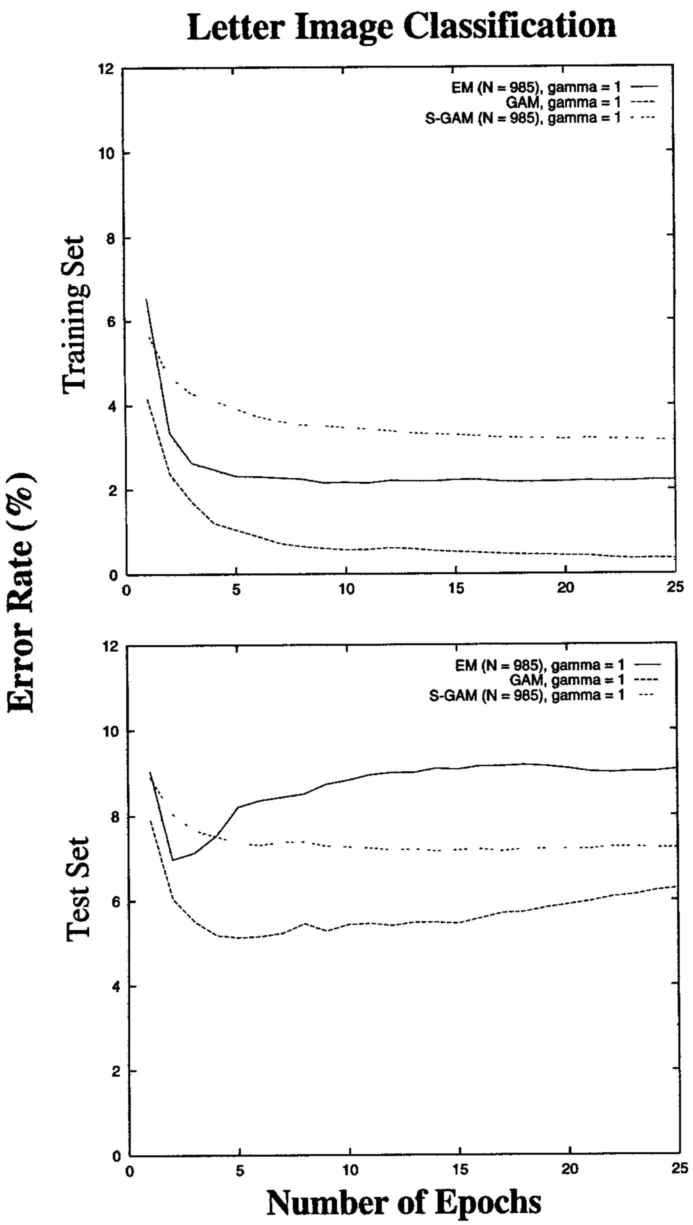

Figure 3: Error rates of EM, GAM, and S-GAM on the training set (top) and the test set (bottom) plotted as a function of the number of training epochs.

Copyright © 2001 All Rights Reserved

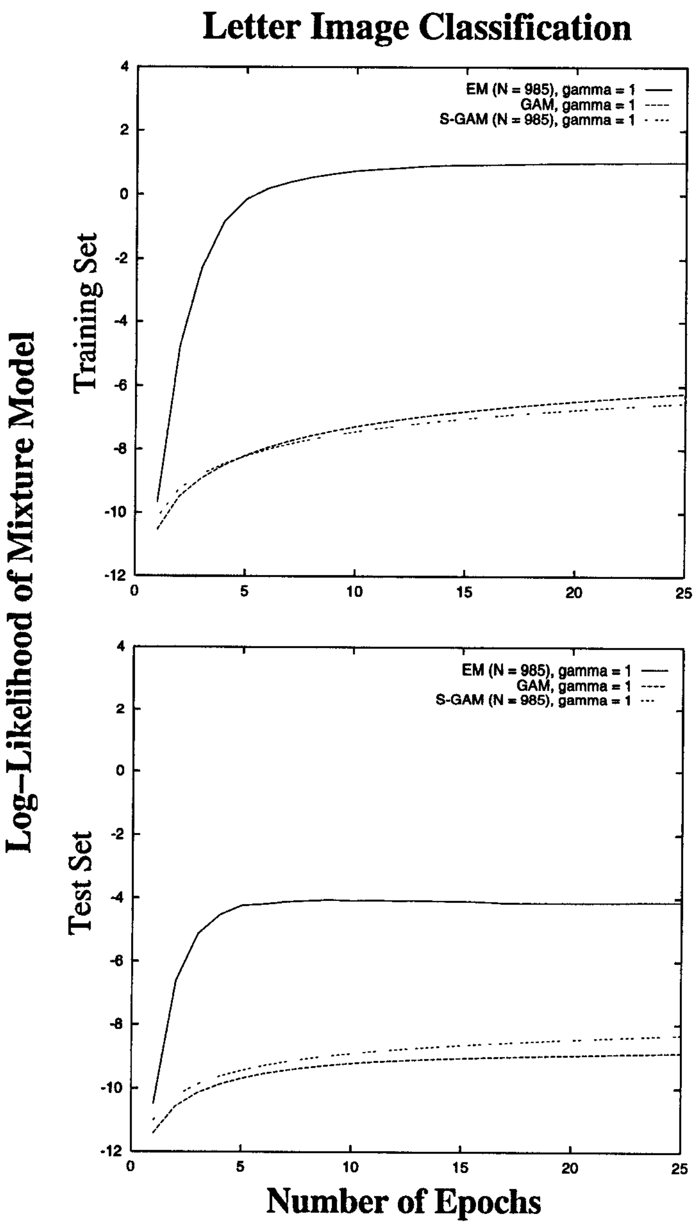

Figure 4: Log likelihoods of EM, GAM, and S-GAM on the training set (top) and test set (bottom) plotted as a function of the number of training epochs.

Copyright © 2001 All Rights Reserved

Table 3: Satellite Image Classification.

Algorithm	Error Rate (%)
k-NN	10.6
FAM	11.0
RBF	12.1
Alloc80	13.2
INDCART	13.7
CART	13.8
MLP	13.9
NewID	15.0
C4.5	15.1
CN2	15.2
Quadra	15.3
SMART	15.9
LogReg	16.9
Discrim	17.1
CASTLE	19.4

Source: Results adapted from Feng et al. (1993) and Asfour et al. (1995).
Note: The *k*-NN result reported here, which we obtained, is different from the *k*-NN result reported in Feng et al. (1993).

275, following two epochs and following equilibration. For $N < 100$ EM performs better after equilibration, whereas for $N > 100$ EM performs better after two epochs. EM's best results (10.6 ± 0.4 percent error) are obtained with $N = 275$. GAM's error rate is plotted for $\gamma = 1, 2, 4$. Once again, GAM achieves the lowest overall error rate (10.0 ± 0.4 percent error), although on this problem GAM outperforms EM only with $\gamma = 1$. It is difficult to distinguish GAM's error curves because they overlap each other so much. Unlike the letter image recognition results, none of GAM's error curves tails up due to overtraining. Therefore, all the curves appear to be on the left side of our hypothetical U-shaped envelope. Table 2b reports the lowest error rate for different values of γ, along with the relevant statistics: number of training epochs, number of categories, and storage rate.

Figure 6 plots the error rates as a function of the number of training epochs for GAM ($\gamma = 1$), EM ($\gamma = 1, N = 250$), and S-GAM ($\gamma = 1, N = 250$). GAM's performance generally increases throughout, whereas EM's performance again peaks at two epochs and then quickly degrades and stabilizes. On this problem EM outperforms S-GAM.

Copyright © 2001 All Rights Reserved

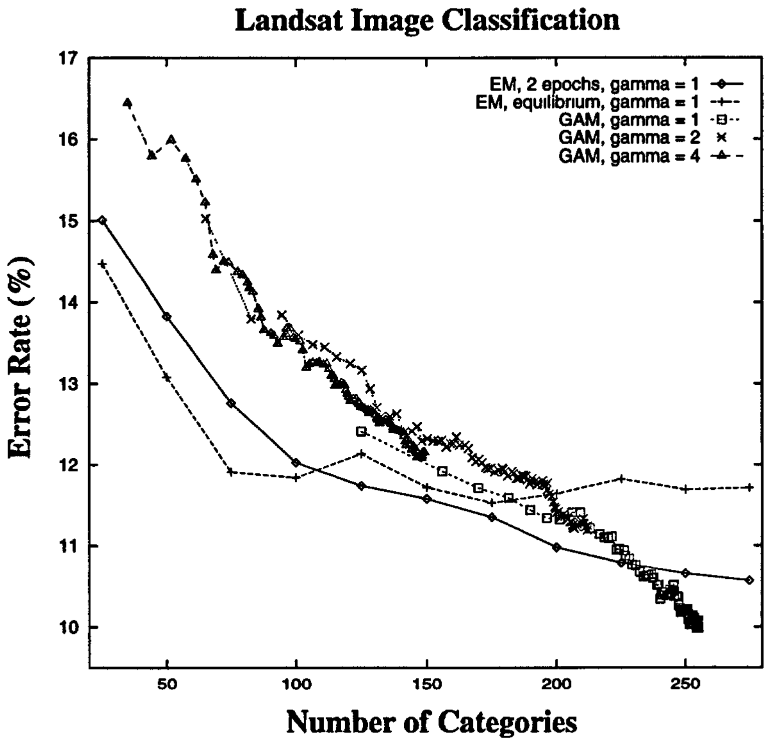

Figure 5: Average error rates of EM and GAM plotted as a function of the number of categories.

4.4 Speaker-Independent Vowel Recognition. Finally, EM, GAM, and S-GAM are evaluated on another real-world task: speaker-independent vowel recognition (Deterding, 1989). The data set is archived in the CMU connectionist benchmark collection (Fahlman, 1993). The data were collected by Deterding (1989), who recorded examples of the 11 steady-state vowels of English spoken by 15 speakers. A word containing each vowel was spoken once by each of the 15 speakers (7 females and 8 males). The speech signals were low pass filtered at 4.7 kHz and then digitized to 12 bits with a 10-kHz sampling rate. Twelfth-order linear predictive analysis was carried out on six 512-sample Hamming windowed segments from the steady part of the vowel. The reflection coefficients were used to calculate 10 log area parameters, giving a 10-dimensional input space. Each speaker thus

Copyright © 2001 All Rights Reserved

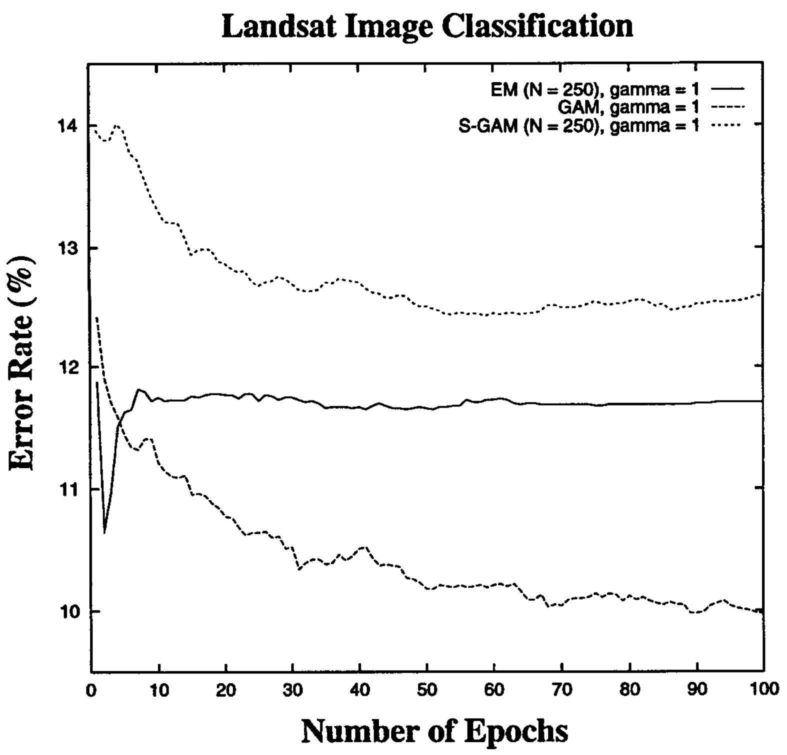

Figure 6: Average error rates of EM, GAM, and S-GAM plotted as a function of the number of training epochs.

yielded six samples of speech from the 11 vowels, resulting in 990 samples from the 15 speakers. The data are partitioned into 528 samples for training, from four male and four female speakers, and 462 samples for testing, from the remaining four male and three female speakers. For comparison, the results of several other classifiers are shown in Table 4 (see Robinson, 1989; Fritzke, 1994; Williamson, 1996a).

Figure 7 shows the classification results of EM and GAM on the vowel recognition problem. EM's error rate is plotted for $N = 15, 20, \ldots, 70$, following two epochs and following equilibration. For all N, EM obtains better results after two epochs than after equilibration. EM's best results (45.4 ± 1.1 percent error) are obtained with $N = 40$. GAM's error rate is plotted for $\gamma = 2, 4, 8$. By a small margin, GAM achieves the lowest overall

Copyright © 2001 All Rights Reserved

Table 4: Spoken Vowel Classification.

Algorithm	Error Rate (%)
k-NN	43.7
MLP	49.4
MKM	57.4
RBF	52.4
GNN	46.5
SNN	45.2
3-D GCS	36.8
5-D GCS	33.5
GAM-CL ($\gamma = 2$)	43.3
GAM-CL ($\gamma = 4$)	41.8
FAM ($\alpha = 1.0$)	48.9
FAM ($\alpha = 0.1$)	50.4

Source: Results adapted from Robinson (1989), Fritzke (1994), and Williamson (1996a).
Notes: Gradient descent networks (using 88 internal nodes) are: MLP (multilayer perceptron), MKM (modified Kanerva model), RBF (radial basis function), GNN (gaussian node network), and SNN (square node network). Constructive networks are: GCS (growing cell structures), GAM-CL, and FAM.

error rate (43.9 ± 1.4 percent error), but outperforms EM only with $\gamma = 4$ and $\gamma = 8$. The data set is quite small, and hence both algorithms have a strong tendency to overfit the data. Table 2c reports the lowest error rate for different values of γ, along with the relevant statistics: the number of training epochs, number of categories, and storage rate.

Figure 8 plots the error rates as a function of the number of training epochs for GAM ($\gamma = 3$), EM ($\gamma = 3, N = 40$), and S-GAM ($\gamma = 3, N = 40$). GAM's performance peaks at 19 epochs and then slowly degrades. EM's performance peaks at 2 epochs and then degrades quickly and severely before stabilizing. S-GAM's error rate, which stabilizes near 50 percent, never reaches EM's best performance.

5 Conclusions

GAM learns mappings from a real-valued space of input features to a discrete-valued space of output classes by learning a gaussian mixture model of the input space as well as connections from the mixture components to the output classes. The mixture components correspond to nodes in GAM's internal category layer. GAM is a simple neural architecture that em-

Copyright © 2001 All Rights Reserved

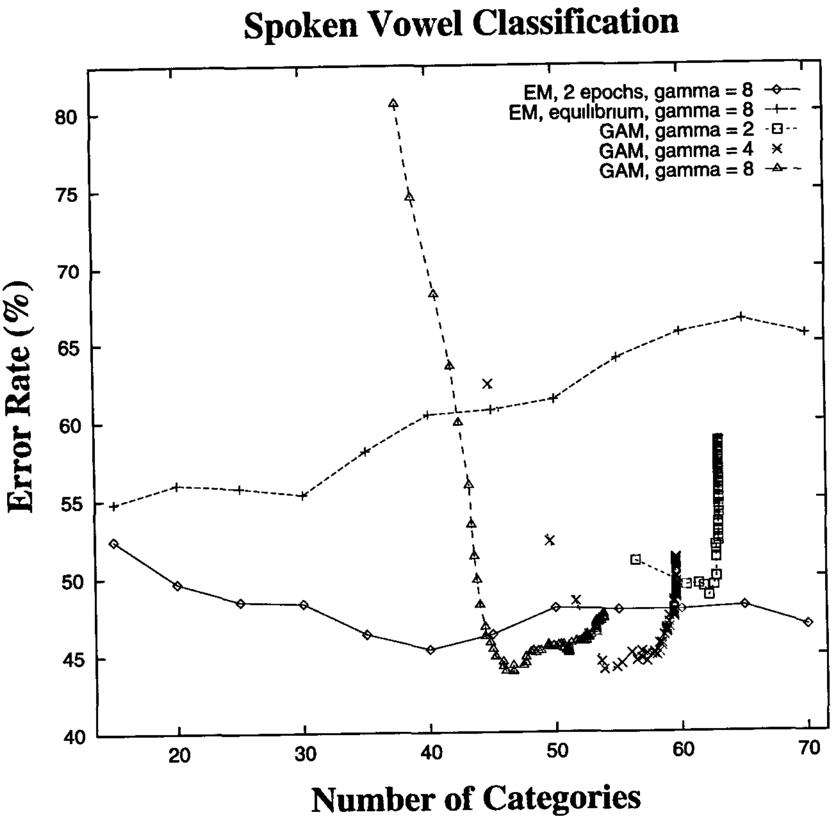

Figure 7: Average error rates of EM and GAM plotted as a function of the number of categories.

ploys constructive, incremental, local learning rules. These learning rules allow GAM to create a representation of appropriate size as it is trained online.

We have shown a close relationship between GAM and an EM algorithm that estimates the joint I/O density by optimizing a gaussian-multinomial mixture model. The major difference between GAM and EM is GAM's match tracking procedure, which raises a match criterion following incorrect predictions and prevents nodes from learning if they do not satisfy the raised match criterion. Match tracking biases GAM's estimate of the joint I/O density such that GAM's predictive error in the I $\rightarrow$ O map is reduced. With this biased density estimate, GAM outperforms EM on classification bench-

Copyright © 2001 All Rights Reserved

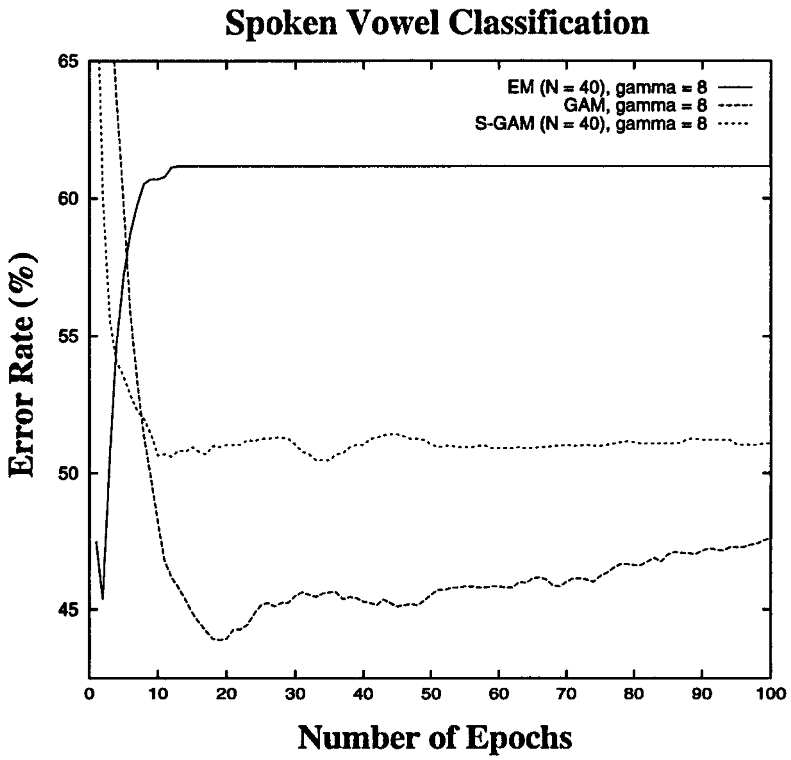

Figure 8: Average error rates of EM, GAM, and S-GAM plotted as a function of the number of training epochs.

marks despite the fact that GAM uses a suboptimal incremental approximation to EM's batch learning rules and learns mixture models that have lower likelihoods than those optimized by EM.

Acknowledgments

This work was supported in part by the Office of Naval Research (ONR N00014-95-1-0409).

References

Asfour, Y., Carpenter, G. A., & Grossberg, S. (1995). *Landsat satellite image segmentation using the fuzzy ARTMAP neural network* (Tech. Rep. No. CAS/CNS

Copyright © 2001 All Rights Reserved

TR-95-004). Boston: Boston University.

Carpenter, G. A., & Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing, 37,* 54–115.

Carpenter, G. A., Grossberg, S., & Reynolds, J. (1991). ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks, 4,* 565–588.

Carpenter, G. A., Grossberg, S., & Rosen, D. B. (1991). Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks, 4,* 759–771.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Satistical Society Series B, 39,* 1–38.

Deterding, D. H. (1989). *Speaker normalisation for automatic speech recognition.* Unpublished doctoral dissertation, University of Cambridge.

Fahlman, S. E. (1993). *CMU benchmark collection for neural net learning algorithms.* [Machine-readable data repository]. Pittsburgh: Carnegie Mellon University, School of Computer Science.

Feng, C., Sutherland, A., King, S., Muggleton, S., & Henery, R. (1993). Symbolic classifiers: Conditions to have good accuracy performance. In *Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics* (pp. 371–380) Waterloo, Ontario: University of Waterloo.

Frey, P. W. & Slate, D. J. (1991). Letter recognition using Holland-style adaptive classifiers. *Machine Learning, 6,* 161–182.

Fritzke, B. (1994). Growing cell structures—A self-organizing network for unsupervised and supervised learning. *Neural Networks, 7,* 1441–1460.

Ghahramani, Z. & Jordan, M. I. (1994). Supervised learning from incomplete data via an EM approach. In J. D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in Neural Information Processing Systems, 9.* San Mateo, CA: Morgan Kauffman.

Grossberg, S., (1976). Adaptive pattern classification and universal recoding. I: Parallel development and coding of neural feature detectors. *Biological Cybernetics, 23,* 121–134.

Grossberg, S., & Williamson, J. (1997). *A self-organizing neural system for learning to recognize textured scenes* (Tech. Rep. No. CAS/CNS TR-97-001). Boston, MA: Boston University.

Hinton, G. E., & Nowlan, S. J. (1990). The bootstrap Widrow-Hoff rule as a cluster-formation algorithm. *Neural Computation, 2,* 355–362.

King, R. (1992). Statlog databases. *UCI Repository of machine learning databases.* [Machine readable repository at ics.uci.edu:/pub/machine-learning-databases].

Neal, R. M., & Hinton, G. E. (1993). *A new view of the EM algorithm that justifies incremental and other variants.* Unpublished manuscript.

Poggio, T., & Girosi, F. (1989). *A theory of networks for approximation and learning* (A.I. Memo No. 1140). Cambridge, MA: M.I.T., 1989.

Robinson, A. J. (1989). *Dynamic error propagation networks.* Unpublished doctoral dissertation, Cambridge University.

Copyright © 2001 All Rights Reserved

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel Distributed Processing* (pp. 318–362). Cambridge, MA: MIT Press.

Williamson, J. R. (1996a). Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps. *Neural Networks, 9,* 881–897.

Williamson, J. R. (1996b). *Neural networks for image processing, classification, and understanding.* Unpublished doctoral dissertation, Boston University.

Xu, L. & Jordan, M. I. (1996). On convergence properties of the EM algorithm for gaussian mixtures. *Neural Computation, 8,* 129–151.

Received July 9, 1996, accepted January 29, 1997.

Copyright © 2001 All Rights Reserved