A Neural Model for Self Organizing Feature Detectors and Classifiers in a Network Hierarchy
A neural model for self organizing feature detectors and classifiers in a network hierarchy

James R. Williamson

November 1998

Technical Report CAS/CNS-98-033

Permission to copy without fee all or part of this material is granted provided that: 1. The copies are not made or distributed for direct commercial advantage; 2. the report title, author, document number, and release date appear, and notice is given that copying is by permission of the BOSTON UNIVERSITY CENTER FOR ADAPTIVE SYSTEMS AND DEPARTMENT OF COGNITIVE AND NEURAL SYSTEMS. To copy otherwise, or to republish, requires a fee and/or special permission.

Copyright © 1998

Boston University Center for Adaptive Systems and Department of Cognitive and Neural Systems

677 Beacon Street

Boston, MA 02215
A NEURAL MODEL FOR SELF ORGANIZING FEATURE DETECTORS 
AND CLASSIFIERS IN A NETWORK HIERARCHY

James R. Williamson*
Department of Cognitive and Neural Systems
and Center for Adaptive Systems
Boston University

Submitted: November, 1998
Technical Report CAS/CNS TR-98-033

Submitted to Neural Computation

All correspondence should be addressed to:

Professor James R. Williamson
Department of Cognitive and Neural Systems
Boston University
677 Beacon Street
Boston, MA 02215
Phone: 617-353-6743
Fax: 617-353-7755
E-mail: jrw@cns.bu.edu

*Supported in part by the Defense Advanced Research Projects Agency and the Office of Naval Research (ONR N00014-95-1-0409).
Abstract

Many models of early cortical processing have shown how local learning rules can produce efficient, sparse-distributed codes in which nodes have responses that are statistically independent and low probability. However, it is not known how to develop a useful hierarchical representation, containing sparse-distributed codes at each level of the hierarchy, that incorporates predictive feedback from the environment. We take a step in that direction by proposing a biologically plausible neural network model that develops receptive fields, and learns to make class predictions, with or without the help of environmental feedback. The model is a new type of predictive adaptive resonance theory network called Receptive Field ARTMAP, or RAM. RAM self organizes internal category nodes that are tuned to activity distributions in topographic input maps. Each receptive field is composed of multiple weight fields that are adapted via local, on-line learning, to form smooth receptive fields that reflect the statistics of the activity distributions in the input maps. When RAM generates incorrect predictions, its vigilance is raised, amplifying subtractive inhibition and sharpening receptive fields until the error is corrected. Evaluation on several classification benchmarks shows that RAM outperforms a related (but neurally implausible) model called Gaussian ARTMAP, as well as several standard neural network and statistical classifiers. A topographic version of RAM is proposed, which is capable of self organizing hierarchical representations. Topographic RAM is a model for receptive field development at any level of the cortical hierarchy, and provides explanations for a variety of perceptual learning data.
1 Introduction

Two goals for models of learning in neural networks are (i) to explain how real, biological neural networks learn, and (ii) to solve applied problems by learning to make accurate predictions. We refer to models with these alternative goals as biological and functional models. Biological models are constrained to have nodes that obey neurally plausible activation and learning rules, and are typically used to explain early sensory processing and development. Examples of these are neural network models of how simple cells and complex cells develop their orientationally tuned receptive fields within smoothly varying maps of orientation preference and ocular dominance (Durbin and Mitchison, 1990; Obermayer et al., 1990, 1992; Swindale, 1992; Siros and Miikkulainen, 1994; Olson and Grossberg, 1998).

These models share similar computational themes in their neural activation and synaptic learning rules: (i) center-surround connections within a layer cause cells that are nearby to be correlated and cells that are further apart to be anticorrelated; (ii) synaptic weights change their strength based on local information, typically obeying a variant of Hebbian learning, in which weights increase or decrease based on pre- and post-synaptic activity correlations. These models share the computational goals of local correlation and global decorrelation: converting input signals that contain high-order correlations into sparse-distributed codes that contain statistically independent outputs (Field, 1994; Olshausen and Field, 1996), where each output is a lump of (locally correlated) activity in a topographic map.

These models decompose the input signal into a representation that is most efficient in general for later stages of processing. However, the representation may not be optimal for the particular needs (which change over time) of the system in which it is embedded. In other words, a major limitation of unsupervised learning models is that they do not take advantage of predictive feedback, which evaluates the environmental usefulness of the representation they learn. Therefore, unsupervised learning alone is insufficient for modeling the sensory/perceptual cortical hierarchy.

What we need is to incorporate biologically plausible mechanisms for supervised learning into unsupervised learning models. However, standard supervised learning networks are functional models that are concerned with the computational goal of learning — generalization — rather than the biological plausibility of its implementation. For example, two popular supervised learning networks of this type are multilayer perceptron (MLP) and radial basis function (RBF) networks (Rumelhart, Hinton, and Williams, 1986; Poggio and Girosi, 1989), which learn mappings from an $M$-dimensional real-valued input space into a $C$-dimensional real- or binary-valued output space via an $N$-dimensional hidden layer. Learning in these networks typically involves updating parameters in the hidden layer, which determine the receptive fields of the hidden nodes, via gradient descent-based update rules that use error signals computed in the output layer. In general, updating parameters in the hidden layer requires $CNP$ signals from the output layer, where $P$ is the number of parameters per node in the hidden layer. Therefore, these learning algorithms are neurally implausible because they require massive feedback connections, as well as neurally implausible computations in the hidden layer (Grossberg, 1987; Poggio and Girosi, 1989).

Predictive adaptive resonance theory, or ARTMAP, networks are a class of supervised learning networks that do not require massive feedback projections (Carpenter, Grossberg, and Reynolds,
ARTMAP networks are primarily unsupervised, but can incorporate predictive feedback via changes in an internal vigilance parameter. Raising vigilance sharpens category receptive fields, causing the network to learn to make finer discriminations in those regions of the input space that lead to erroneous predictions. ARTMAP networks learn on-line, adjusting weights after each training sample is presented, and constructively, adding new categories when the current set of categories is insufficient. ARTMAP networks thereby self organize, on-line, a representation of sufficient size and complexity to learn an input/output mapping.

For these reasons, the ARTMAP paradigm is a promising approach for bridging the gap between, on one hand, biologically plausible unsupervised models that learn cortical maps, and, on the other hand, biologically implausible supervised models that learn input/output mappings. We take a step in this direction by proposing a new neural network model called Receptive Field ARTMAP, or RAM, which develops receptive fields and learns to make class predictions, with or without predictive feedback, using only simple, local computations that are biologically plausible.

The rest of this article is organized as follows. In Section 2 the RAM network is described heuristically, and in Section 3 its equations are described in detail. In Section 4 RAM's receptive field properties are described and examples are illustrated of the decision regions and receptive fields that it develops. In Section 5 a detailed comparison is made of RAM and Gaussian ARTMAP, which is a related but neurally implausible network. In Section 6 the functional competence of RAM is demonstrated by evaluating it on several classification benchmarks. On the whole, RAM outperforms Gaussian ARTMAP as well as several other neural network and statistical classifiers. In Section 7 RAM's memory requirements are discussed. It is shown that the majority of its weights can be pruned with little or no effect on performance. Finally, in Section 8 a topographic version of RAM is proposed, which allows RAM to self organize a hierarchical representation consisting of topographic maps at each level of the hierarchy. Explanations are given for how topographic RAM could produce many perceptual learning phenomena. Topographic RAM's relation to a laminar model of visual cortex is also discussed.

2 Receptive Field ARTMAP Network

Unlike traditional supervised learning networks, which operate on arbitrary real-valued input vectors, RAM operates on topographic maps and takes advantage of their local activity correlations. The generalizability of standard neural networks, such as RBFs, depends on how well their receptive fields (or basis functions) interpolate between training samples in the input space. The key is to learn both the optimal placement and amount of spread of the receptive fields. This involves adjusting the basis function parameters, such as the means and variances of Gaussian distributions. RAM, on the other hand, utilizes the fact that topographic maps, due to their locally correlated activities, generate input signals that already encode a good deal of spread in the input space, and thereby provide a certain amount of regularization. RAM is able to learn smooth distributions in the input space by updating its weight fields using simple, correlational learning rules that encode the sample average of the input distributions it learns.

Figure 1 diagrams the RAM network, which consists of an $F_1$ layer containing input fields, an
$F_2$ layer containing category nodes, and a $C_2$ layer containing class output nodes. Distributions depicted in each layer represent patterns of activity among sets of neurons, and solid arrows represent distributed, learned connections, as shown in the key in Figure 1 (bottom). The $F_1$ fields are topographic maps, as indicated by their smooth distributions of locally correlated activities. The $F_2$ and $C_2$ layers are nontopographic, as indicated by their jagged distributions of uncorrelated activities. The $F_1$ fields map into the $F_2$ layer via learned connections, producing an initial activity pattern in $F_2$ (dotted line). $F_2$ categorizes the input with a distributed code. Learned connections from $F_2$ to $C_2$ produce an initial activity pattern in $C_2$ (dotted line). $C_2$ codes the network's class prediction, which is determined by winner-take-all competition (bold line). In the supervised learning case, the winner of this competition is determined by a supervised signal (not shown) as to the correct output class. However, if the bottom-up $C_2$ pattern does not match this supervised signal sufficiently well, then match tracking is triggered: $F_2$ vigilance (represented by $\rho$) is raised, altering the $F_2$ activity pattern via learned, subtractive inhibition, which in turn alters the $C_2$ activity pattern, until match at $C_2$ is obtained. Once match at $C_2$ is obtained, $C_2$ resonates, with the correct class node winning the competition (bold line). $C_2 \rightarrow F_2$ feedback then creates a resonant activity distribution at $F_2$ (bold line). The resonant activities at $F_2$ and $C_2$ control correlational learning in all the adaptive weights.

How much does RAM deviate from a purely unsupervised network? Imagine what would happen if match tracking were disabled. This could be done by using a supremely inclusive match rule in $C_2$, in which match is always obtained regardless of the initial $F_2$-induced pattern of activity at $C_2$ (dotted line). Then, because the resonant $C_2$ activity (bold line) is determined by the supervised signal, and because the resonant $F_2$ activity is determined by input from both $F_1$ and $C_2$, the $F_2 \rightarrow C_2$ signals would become irrelevant and the network would become, in effect, unsupervised. That is, the supervised learning problem would be treated as an unsupervised one in which the supervised signal from $C_2$ is treated as just another input. By learning a categorization in $F_2$ of the joint input/output, or I/O, distribution, the network could then (via its symmetrical $F_2 \rightarrow C_2$ connections) predict the output at $C_2$ given input from only $F_1$.

Similarly, Ghahramani and Jordan (1994) used the unsupervised expectation-maximization, or EM, algorithm to learn a model of the joint I/O density for supervised learning problems. EM is a batch learning algorithm that guarantees convergence to a local maximum in the model’s likelihood. An EM classifier can be learned by modeling the real-valued input signals with a fixed number of Gaussian distributions, and modeling the binary-valued output signals with a multinomial distribution. The EM algorithm can then learn an unbiased model of the joint I/O distribution, without regard to the direction, $I \rightarrow O$ or $O \rightarrow I$, of the desired mapping. However, maximizing the likelihood of this joint I/O density model is not the same thing as maximizing the accuracy of its $I \rightarrow O$ predictions. This raises the question: can the model's classification performance be improved by biasing its unsupervised learning during training, based on evaluations of its predictive accuracy in the desired direction ($I \rightarrow O$)?

Williamson (1997) showed that the Gaussian ARTMAP, or GAM, network does precisely this, by using match tracking to bias its otherwise unsupervised learning rules, and thereby improving upon the classification performance of the unsupervised EM algorithm. GAM has the same structure as RAM, but learns explicit Gaussian receptive fields in the $F_2$ layer, using on-line approximations of the EM update equations. Without match tracking, GAM learns a less likely
Figure 1: Receptive Field ARTMAP network. See text for details.
mixture model than EM (on-line approximations of EM do not guarantee convergence to a local maximum in likelihood), and as a result is a worse classifier. With match tracking, however, GAM becomes a better classifier than EM even though it still learns a less likely mixture model. The RAM network was developed out of a desire to obtain similar computational performance as GAM, but to do so with a network that uses neurally plausible local computations throughout its architecture, as described below.

3 Receptive Field ARTMAP Equations

3.1 Input Conversion

RAM is a model for the development of receptive fields at one layer of a cortical hierarchy ($F_2$), based on converging input from multiple topographic activity maps lower in the hierarchy ($F_1$; see Figure 1). It rests upon the assumption that these activity maps are characterized by two properties which arise naturally from competitive dynamics in center-surround neural architectures (Grossberg, 1973; von der Malsburg, 1973):

1. Maps contain locally correlated "lumps" of activity;
2. A map's net activity level is roughly constant.

To demonstrate that RAM learns efficient and useful internal representations, its performance is evaluated in Section 6 on standard classification benchmarks, for which the results of many neural network and statistical classifiers are available. In order to test RAM on these benchmarks, the real-valued input features need to be converted into topographic activity maps, or input fields, with the two properties listed above. Thus, the magnitude of the $i^{th}$ input feature, $I_i$, is converted into a Gaussianly distributed pattern of activity in the $i^{th}$ $F_1$ field, $x_i = \{ x_{ih} \}_{h=1}^L$:

$$x_{ih} = \frac{\exp \left[ - \frac{(I_i - b_i/L)^2}{2(\sigma_i)^2} \right]}{\sum_{h'=1}^L \exp \left[ - \frac{(I_{i} - b_{h'}/L)^2}{2(\sigma_{h'})^2} \right]}.$$  

(1)

For example, Figure 2 illustrates how a 3-dimensional input vector is converted into three $F_1$ input fields. The input vector, $I = \{ I_i \}_{i=1}^M$, corresponds to a point in an $M$-dimensional input space. Equation (1) produces a representation in the $M$ $F_1$ fields, $\{ x_{ih} \}_{i=1}^M$, that synthesizes a Gaussian distribution around that point, $G(I, \alpha \sigma)$. In all simulations reported here, the input vectors, $\{ I^{(t)} \}_{t=1}^T$, were first normalized (across the entire data set) to a range of [0:1] in each dimension. After this normalization, the $\sigma_i$ values in equation (1) --- the standard deviations from each dimension $i$ of the training set --- were computed. These $\sigma_i$ values cause the width of each lump of activity produced by equation (1) to be proportional to the standard deviation of that feature.
3.2 Category Match and Activation

The $F_2$ layer consists of $N$ committed category nodes, each of which responds to a region of the $M$-dimensional input space. In the beginning of training, all $F_2$ nodes are uncommitted. As the network is trained the nodes become committed, one by one, until a sufficiently complex categorization of the input space is achieved. The $i^{th}$ weight field of the $j^{th} F_2$ node, $w_{ji}$, is a weighted average of an initial, uniform distribution, and of all the subsequent Gaussian distributions in the $i^{th} F_1$ field that the node has learned. The dot product of $w_{ji}$ and $x_i$ defines the match between this weight distribution and the current Gaussian distribution at the $i^{th} F_1$ field. The match computed at each weight field is multiplied across the dimensions to determine a net, multidimensional match:

$$g_j = \prod_{i=1}^{M} \sum_{h=1}^{L} x_{ih} w_{jih} = \prod_{i=1}^{M} x_i^T \cdot w_{ji}. \quad (2)$$

Equation (2) thus defines the match between $G(I, \alpha \sigma)$, which is the $M$-dimensional Gaussian activity distribution in $F_1$, and $\{w_{ji}\}_{i=1}^{M}$, which is the $M$-dimensional distribution synthesized by the weight fields of the $j^{th} F_2$ node.

The computation in equation (2) can plausibly be implemented with neurons (Poggio and Girosi, 1989). Figure 3 (top) shows a compact, single-neuron model of equation (2), in which weighted inputs are linearly summed in dendritic branches, and then multiplied at the base of the dendritic tree. However, we are not aware of evidence for successive sum and multiply operations.
Figure 3: Two neural models for computing match in equation (2). The single-neuron model (top) computes linear summations followed by multiplication in a single dendritic tree. The likelier, multiple-neuron model (bottom) computes linear summations in intermediate cells, followed by multiplication in the transduction from these cells to the final $F_2$ cell.
within a single dendritic tree. Figure 3 (bottom) shows a likelier model, in which intermediate cells compute linear summations, and the transduction from these cells to the final $F_2$ cell produces the multiplication. This multiplication can arise from successive log, sum, and exponentiate computations using integrate-and-fire presynaptic cells (Tal and Schwartz, 1997). Alternatively, it can arise from local dendritic mechanisms in the postsynaptic cell (Mel, 1994). One advantage of the multiple-cell model in Figure 3 (bottom) is that it allows for possible weight sharing (Poggio and Girosi, 1989). That is, each intermediate cell could output to multiple final $F_2$ cells, thereby conserving neural resources. Another advantage is that each final $F_2$ cell could receive the sum of different sets of multiplied inputs (Mel, 1994). We do not pursue these possibilities further in this paper.

The final $F_2$ cell is activated only if $g_j$ is large enough. In other words, if $g_j$ is too small then the input is treated as an outlier and ignored. How large does $g_j$ need to be? This is determined by a global vigilance parameter, $\rho$, and a local match threshold, $T_j$. The $j^{th}$ $F_2$ node's bottom-up activation obeys

$$y_j = [g_j - \rho T_j]^+, \quad (3)$$

where $[\cdot]^+$ is the half-wave rectification operator. Figure 3 shows that the local match threshold could be implemented via an inhibitory connection either at the $F_2$ cell's soma or at the base of its dendritic tree — regions where inhibitory synapses are common (McGuire et al., 1991).

The vigilance parameter, $\rho$, is typically set to a baseline value of zero ($\rho = \bar{\rho} = 0$). It is only when the network makes incorrect predictions during training, as described below in Section 3.4, that $\rho$ is raised above its baseline value. The local match threshold, $T_j$, provides a measure of a node's expected input match. Section 3.5 describes how, during learning, $T_j$ converges to its node's average match value. Thus, in equation (3) a node is activated only if the ratio of its current match with its expected match, $g_j/T_j$, is larger than $\rho$. The ratio $g_j/T_j$ is the node's normalized match.

### 3.3 Output Prediction

Each RAM category node has connections to nodes in the $C_2$ layer, which determine the prediction of an output class. The connection from the $j^{th}$ category node to the $k^{th}$ class node, $p_{jk}$, represents the node's estimate of the probability of that class. The class nodes are activated by summing the inputs from all the $F_2$ nodes:

$$z_k = \sum_{j=1}^{N} y_j p_{jk}, \quad (4)$$

During testing, the class prediction, $K$, is determined by the most active class node, which is chosen via winner-take-all competition:

$$K = \arg \max_k (z_k). \quad (5)$$

Winner-take-all competition is obtained in a competitive shunting network that has faster-than-linear signal functions (Grossberg, 1973).
During training, the winner of the $C_2$ competition is determined by the supervised signal, $s_k$, where $s_k = 1$ if $k$ is the correct output class, and $s_k = 0$ otherwise. These competitive dynamics are not modeled in detail, but an approximation is adopted that determines whether or not the match between the $F_2$ signal and the supervised signal is good enough for the network to resonate. The $C_2$ vigilance parameter, $\rho^{(e)}$, determines the inclusiveness of the match rule in the $C_2$ layer. If $\rho^{(e)} = 0$, then match is always obtained at $C_2$, and the network becomes unsupervised, as discussed in Section 2. Therefore, the value of $\rho^{(e)}$, within the range $[0:1]$, determines the network’s level of supervision.

Let $K$ be the index of the class node that has maximum $F_2$ input via equation (4), and let $K^*$ be the index of the correct output class, as determined by the supervised signal. Then, if

$$
\frac{z_K}{z_{K^*}} > \rho^{(e)},
$$

the network is said to \textit{resonate}, and learning is triggered. When this happens, the supervised signal determines the winner of the competition:

$$
K = K^*.
$$

The chosen class then feeds back signals to the $F_2$ layer, altering the category activities based on the new information as to the correct class. These altered, resonant, activities then determine the rate at which the categories learn.

### 3.4 Match Tracking

If there is no match in $C_2$ (i.e., if equation (6) is not satisfied), then match tracking is triggered (Carpenter, Grossberg, and Reynolds, 1991). The $F_2$ vigilance parameter, $\rho$, is raised from its baseline value of $\rho = \bar{\rho}$ until either the network resonates (i.e., equation (6) is satisfied), or until all the committed categories become inactive. Raising $\rho$ inactivates, or \textit{resets}, committed categories via the subtractive inhibition in equation (3). Each time a category is reset, the class activities computed via equation (4) are altered, and the $C_2$ match condition in equation (6) may become satisfied.

This simple match tracking operation is similar to the one originally proposed by Carpenter, Grossberg, and Reynolds (1991). In an ideal parallel implementation, $\rho$ would be raised slowly and equations (3)–(6) evaluated continuously until a match is obtained in the $C_2$ layer (or all committed categories are reset). Our serial implementation approximates this continuous-time approach by iteratively raising $\rho$ to the lowest normalized match value among those active categories that have a positive connection to the chosen output class (thereby resetting one category that affects the current class prediction), and then reevaluating equations (3)–(6).

### 3.5 Learning

When RAM achieves a match at the $C_2$ layer (i.e., equation (6) is satisfied), its active categories \textit{resonate} and learn by an amount proportional to their resonant activity, which represents their
credit given both the input pattern and the chosen class prediction. Resonant activity is determined by feedback from the chosen class, $K$, which reinforces categories in proportion to the strength of the bottom-up weights, $p_{jK}$, via symmetrical top-down weights, which are also denoted by $p_{jK}$. The $F_2$ layer normalizes its activities via competitive shunting dynamics with linear signal functions (Grossberg, 1973). Hence, the resonant activity is

$$y_j^* = \frac{p_{jK} y_j}{\sum_{j=1}^{N} p_{jK} y_j},$$

which is the network's estimate of the probability of category $j$ given both the input and the chosen class: $P(j|I, K)$. In a parallel hardware implementation, $F_2$ competition could be initiated immediately following $F_2$ activation in equation (3), and $C_2 \rightarrow F_2$ feedback could be initiated immediately following activation of the class nodes in equation (4), even while match tracking is taking place. However, winner-take-all competition in the $C_2$ layer would need to take place sufficiently quickly that the $C_2 \rightarrow F_2 \rightarrow C_2$ feedback loop (approximated by equations (4)–(8)) would not alter the ordinal relations among the class activations before the class winner is chosen.

The $F_2$ categories learn by amounts proportional to their resonant activity. For each category $j$, three different types of weights are learned: (i) the weight field weights, $\{w_{ji}\}_{i=1}^{M}$, (ii) the bottom-up and top-down class weights, denoted by $p_{jK}$, and (iii) the match threshold, $T_j$. The weights are updated via differential equations that operate on a slow time scale relative to changes in the activity variables described above. When a category is first committed, its weight field and class weights learn at a relatively fast rate which slows down as the category becomes more experienced. This experience level is represented by a "count" variable, $n_j$, which is initialized to zero and updated via the differential equation,

$$\frac{1}{\delta_n} \frac{d}{dt} n_j = (1 - n_j) y_j^*.$$ 

The learning rate $\delta_n$ is chosen very small so that equation (9) is essentially linear, and $n_j$ integrates the resonant activity of node $j$ over all the training inputs. In our simulations the first-order Euler approximation of equation (9) is implemented:

$$n_j := n_j + \delta_n (1 - n_j) y_j^*.$$ 

Similarly, all the learning equations below are approximations of ideal continuous-time differential equations.

The learning rate that is used to update weight field and class weights, denoted by $\delta_{n_j}$, is a decreasing function of experience,

$$\delta_{n_j} = \frac{\delta_n}{\beta \delta_n + n_j},$$

which (approximately) converges to $\delta_n$ as $n_j \rightarrow 1$. This experience-dependent learning rate allows weight field and class weights to quickly learn an initial, well-defined template, and then to slowly adapt to the statistics of a large number of inputs.

Weight field weights are updated using an instar equation (Grossberg, 1976, 1980; Kohonen,
in which the weights track the presynaptic $F_1$ signals, sampled by the resonant postsynaptic $F_2$ activities:

$$w_{jih} := w_{jih} + \delta_{nj}(x_{ih} - w_{jih})y_j^*.$$  \hfill (12)

The bottom-up class weights are updated using an outstar equation (Grossberg, 1968, 1980), in which the weights track the resonant postsynaptic activities in $C_2$, sampled by resonant presynaptic $F_2$ signals:

$$p_{jk} := p_{jk} + \delta_{nj}(z_k^* - p_{jk})y_j^*.$$  \hfill (13)

Following equilibration of winner-take-all competition in the $C_2$ layer, the resonant $C_2$ activities, $z_k^*$, are:

$$z_k^* = 1 \text{ if } k = K; \quad z_k^* = 0 \text{ otherwise}.$$  \hfill (14)

The top-down $C_2 \rightarrow F_2$ weights are denoted by $p_{jk}$, just like the bottom-up $F_2 \rightarrow C_2$ weights. The top-down weights are updated via an instar equation which is identical to equation (13) because the weights track the presynaptic signal from $C_2$, sampled by the postsynaptic activity in $F_2$.

The match threshold $T_j$ tracks the excitatory input signal to the $j^{th} F_2$ node, at a rate determined by the node’s resonant activity:

$$T_j := T_j + \delta_T(g_j - T_j)y_j^*,$$  \hfill (15)

where $\delta_T$ is a constant rate parameter. Thus, over time $T_j$ converges to the average input match of its node.

**Category Instantiation.** If RAM does not make a correct prediction, and match tracking has reset all the committed categories, then a new category is committed. This is the only way that categories become committed. When this happens, the new category’s weights are initialized as follows:

$$N := N + 1,$$

$$j := N,$$

$$n_j = 0,$$

$$T_j = 0,$$

$$w_{jih} = (1 + r^*)/L \text{ where } r^* \text{ is a random number selected from } U[-0.00005 : 0.00005],$$

$$p_{jk} = 1/C \text{ where } C \text{ is the number of nodes in the } C_2 \text{ layer}.$$  

The resonant activity of the newly committed node is $y_j^* = 1$, and learning equations (10)-(15) are applied.

**Weight Pruning.** In order to conserve the number of weights used by the network, weight field and class weights are pruned after each weight update. If any weight becomes smaller than the pruning threshold, $\Gamma$, then the weight is permanently set to zero. With this rule, RAM produces a pattern of synaptic proliferation followed by refinement that is analogous to cortical developmental
data (Calloway and Katz, 1990; Kandel and O'Dell, 1992; Antonini and Stryker, 1993). Newly committed categories have uniformly distributed weight field and class weights. As the categories become more experienced, their weight distributions become much narrower, and many weights (i.e., synaptic connections) disappear.

**Parameters.** The same network parameters were used to obtain all simulation results:

\[ p = 0.0, \quad \rho(z) = 0.85, \quad \delta_n = 0.0000001, \quad \beta = 4.0, \quad \delta_T = 0.0001, \quad \Gamma = 0.005. \]

The only parameters that were varied in the simulations are the input field parameters used in equation (1). On the smaller datasets wide input field distributions were used: \( \alpha = 0.75, L = 7 \). On the larger datasets narrow input field distributions were used: \( \alpha = 0.375, L = 10 \). These choices for the input field parameters are discussed below.

## 4 RAM Receptive Fields

### 4.1 Input Fields, Weight Fields, and Receptive Fields

The width of the receptive fields that RAM learns is determined largely by the width of the input field distributions defined in equation (1). Equation (12) prevents a weight field from converging to a narrower distribution than the input distributions. Thus, the fixed parameter \( \alpha \) in equation (1) defines the minimum amount of variance that a category may learn to represent in any input dimension.

Figure 4 shows the two types of input distributions, dubbed “narrow” and “wide”, that were used in all our benchmark simulations. The choice of \( \alpha \) is dictated by the level of resolution that is necessary to obtain good discrimination in the input space. Not surprisingly, a high resolution (and thus a small \( \alpha \)) yielded the best results on large datasets, while a low resolution (and thus a large \( \alpha \)) yielded the best results on the small datasets. Once \( \alpha \) is chosen, two considerations influence the choice of \( L \): If \( L \) is too small with respect to \( \alpha \), then the Gaussian distribution will be undersampled, and receptive fields will become distorted due to aliasing; if \( L \) is too large, then many redundant weights will be stored, which is a waste of memory resources. Figure 4 illustrates how these opposing constraints are balanced in our choice of \( \alpha \) and \( L \) for narrow and wide input field distributions.

Figure 4 (top left) shows a narrow input distribution \( (\alpha = 0.375, L = 10) \), and Figure 4 (bottom left) shows a wide input distribution \( (\alpha = 0.75, L = 7) \), given \( I = 0.475 \). If a node were to learn only one of these input patterns, then its weight field would converge to that pattern via equation (12). Therefore, these distributions depict the minimum possible width, or variance, of a learned weight field distribution, given input fields generated with these sigmas. Figure 4 (right) shows the minimum-width 1-D receptive fields that are obtained with weight fields that are identical to the input fields on the left.

Even though the weight fields on the left are very nonsmooth (having only 4 or 5 positive
Figure 4: Top left: narrow input distribution ($\alpha = 0.375, L = 10$) given $I = 0.475$. Top right: narrow minimum-width receptive field resulting from weight field distribution that is identical to the input distribution on the left. Bottom left: wide input distribution ($\alpha = 0.75, L = 7$). Bottom right: wide minimum-width receptive field. For this plot the training set standard deviation in equation (1) is set to a typical value of $\sigma = 0.195$. 

14
values), the receptive fields they yield on the right are very smooth because they are produced via convolutions with shifted versions of the input field Gaussian in equation (2). However, if $\alpha$ or $L$ were reduced very much below their present values, then distortions in the receptive fields due to aliasing would start becoming noticeable, and start degrading the network's performance.

Suppose that two nodes learn the minimum-width receptive fields shown in Figure 4 (right) and their inhibitory weights, $T_j$, converge to their average match values. In this case, their average match values are the same as their maximum match values since each node only learns the single input distribution that optimally matches its receptive field. Figure 5 shows that, if vigilance is raised to $\rho = 0.14$ (left), both receptive fields would respond only to inputs within $\pm 2\sigma$, whereas if vigilance is raised to $\rho = 0.68$ (right), both receptive fields would respond only to inputs within $\pm 1\sigma$. Thus, raising $\rho$ has a similar effect on all receptive fields, independent of their width, once $T_j$ has converged.

4.2 2-Dimensional Example

A simple classification task illustrates how RAM's receptive fields and decision boundaries develop given both narrow and wide input distributions. The task, a variant of the noisy nested spirals benchmarks in Carpenter and Ross (1995) and Williamson (1996), involves learning a mapping from a 2-dimensional input space to two output classes. The inputs corresponding to each class are generated from 27 gaussian distributions that are configured in a spiral pattern. Figure 6 (top left) shows the correct output classes (black = class 1, white = class 2) as a function of the input values on the $x$ and $y$ axes. Figure 6 (top middle) shows the probability density functions, or pdfs, for the two classes, with the class 1 pdf subtracted from the class 2 pdf. Figure 6 (top right) shows the set of 10,000 training samples, with class 1 samples in black and class 2 samples in white. The nested spirals problem was chosen because it contains a highly nonlinear decision boundary which is beyond the ability of RAM to resolve given the input distributions defined in Section 4.1. However, it is instructive to see how well RAM approximates the ideal decision boundary at the two input resolutions.

1-D input fields. Figure 6 (middle left) shows a contour plot of a minimum-width receptive field in the 2-D input space given a narrow input distribution, and Figure 6 (bottom left) shows the same for a wide input distribution. To the right of these contour plots are the decision regions formed by RAM given narrow and wide input distributions. In the narrow case, RAM forms more refined decision regions that more closely approximate the ideal decision regions than it does in the wide case. In both cases, RAM's decision regions appear to be reasonable approximations given the resolution limits dictated by its input distributions.

2-D input fields. The above results were generated using 1-D input fields, as described in equation (1). However, input fields in cortex should generally consist of 2-D topographic activity maps. RAM was therefore also evaluated with 2-D input fields, which were created by taking the outer product of the 1-D input fields used above. By directly learning 2-D maps, each receptive
Figure 5: Effect on receptive fields of raising \( \rho \). Once \( T_j \) has converged to the average match values of the minimum-width receptive fields shown in Figure 4, raising \( \rho \) has the same effect on both the narrow and wide receptive fields, shrinking them to a width of \( 4\sigma \) (left) and \( 2\sigma \) (right).
Figure 6: **Top:** correct decision regions of noisy nested spirals problem (left); difference of probability density functions (middle); 10,000 training samples (black = class 1, white = class 2) (right). **Middle:** using a small scale parameter ($\alpha = 0.375, L = 10$), contour plot of a minimum-width receptive field, with contours at 25%, 50%, and 75% of the receptive field height (left); RAM decision regions with 1-D input fields (middle), and a 2-D input field (right). **Bottom:** using a large scale parameter ($\alpha = 0.75, L = 7$), contour plot of minimum-width receptive field (left); RAM decision regions with 1-D input fields (middle), and a 2-D input field (right).
field is able to capture higher order statistical relationships between the two dimensions, albeit less economically in terms of the number of weights. Figure 6 (right) shows that RAM forms more accurate decision regions using 2-D input fields than it does using 1-D input fields. In the narrow 2-D case (middle right), RAM does not generate any predictions in some of the corners of the input space because the weights that respond to these regions were pruned due to a lack of input support.

Figure 7 shows contour plots of the receptive fields for the first 4 categories in the narrow 1-D case (top) and the narrow 2-D case (bottom). Each category has positive weights, \( p_{jk} \), to both output classes. The contour plots in Figure 7 are weighted by the difference in the class weights, \( p_{j1} - p_{j2} \). The receptive fields are in general agreement with the decision regions RAM produces; note that in the 2-D case, the individual receptive fields can capture higher-order relationships between the two input dimensions.

5 Comparison with Gaussian ARTMAP

RAM is similar to the Gaussian ARTMAP, or GAM, network (Williamson, 1996, 1997; Grossberg and Williamson, 1998a). GAM self organizes a Gaussian mixture model of its input space, with learned mappings from the mixture components to output classes. As mentioned in Section 2, GAM's learning algorithm is closely related to the expectation-maximization, or EM, approach for
mixture modeling (Williamson, 1997). GAM differs from EM in terms of three properties which are standard for ARTMAP networks: (i) learning online versus in batch mode; (ii) restricting learning to a subset of categories via the match threshold; and (iii) learning constructively by committing new categories when no currently committed categories satisfy the match threshold. Because of these three properties, GAM outperformed EM on several benchmark databases. In this section, the similarities and differences between RAM and GAM are summarized. In Section 6, detailed comparisons are made of their performance on several classification benchmarks.

5.1 Category Match and Activation

In GAM, an input vector $I$ is treated as a single point (rather than a distribution) in input space. Each category computes the match of the input vector with its explicit Gaussian distribution, which is defined with a learned mean vector $\mu$ and standard deviation vector $\sigma$:

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

(16)

GAM's match, $G_j$, is analogous to RAM's normalized match, $g_j/T_j$. Therefore, in GAM, just like in RAM, $\rho$ has the same effect on all categories, causing them to disregard inputs that are a fixed distance from their mean in units of their standard deviation. If an input is close enough to the mean to satisfy the match criterion, then the bottom-up activation function,

$$Y_j = \begin{cases} \frac{n_j}{\prod_{i=1}^{M} \sigma_{ji}} G_j & \text{if } G_j > \rho, \\ 0 & \text{otherwise}, \end{cases}$$

(17)

weights the match value, $G_j$, by two factors. First, $G_j$ is divided by $\prod_{i=1}^{M} \sigma_{ji}$, which normalizes the volume under the Gaussian distribution. The result is similar to RAM's match, $g_j$, which is computed using weight fields that are automatically normalized due to learning. Second, $G_j$ is multiplied by a count variable, $n_j$, which is essentially identical to RAM's $n_j$ (see equation (10)). Unlike GAM, RAM does not factor $n_j$ into its $F_2$ activation in equation (3), and therefore does not weight its category activations by their prior probabilities.

Weighting $F_2$ activations by $n_j$ causes frequently activated categories (i.e., categories that already have a large $n_j$) to increase their dominance during learning, resulting in a skewed distribution of prior probabilities. It appears to be more effective for RAM to make the contribution of all the categories as equal as possible, a goal which is consistent with the principle of sparse coding (Field, 1994). In Section 7.1 it is demonstrated that factoring $n_j$ into equation (3) degrades RAM's performance on a classification database.

5.2 Output Prediction

In GAM, each category node maps to only a single output class. A newly committed GAM category, $j$, has little influence on determining the output predictions because $n_j$ is small and
therefore \( Y_j \) tends to be small. The category’s influence grows due to learning because (i) \( n_j \) increases, and (ii) the category’s receptive field becomes narrower, and therefore the denominator in (17) becomes smaller. A newly committed RAM category, \( j \), has relatively stronger activations than a new GAM category because \( n_j \) is not factored into its activation. However, because the RAM category starts with uniformly distributed \( F_2 \rightarrow C_2 \) weights, its influence on determining the output prediction is also weak. The RAM category’s impact grows due to learning because (i) its receptive field becomes narrower, and therefore its optimal match becomes bigger, and (ii) its \( F_2 \rightarrow C_2 \) weights become refined, and it forms a strong connection to a single output class.

5.3 Match Tracking

If GAM makes an incorrect prediction, then \( \rho \) is raised to the average match value of all the categories that contribute to the incorrect prediction:

\[
\rho = \exp \left( -\frac{1}{2} \sum_{j \in E(K)} \frac{Y_j}{Y} \sum_{i=1}^{M} \left( \frac{\mu_{ji} - I_i}{\sigma_{ji}} \right)^2 \right),
\]

after which all these categories are reset. In equation (18) the summation over \( j \in E(K) \) only includes those categories \( j \) that map to the incorrect prediction, and the normalization term \( Y \) is the sum of their activations. The functional advantage of equation (18) is that it always raises the value of \( \rho \), and usually by a very small amount. Thus, equation (18) helps prevent category proliferation, which occurs if match tracking generally raises \( \rho \) too much. From a modeling point of view, the disadvantage of equation (18) is that it is a nonlocal, neurally implausible operation. Moreover, it is undefined for a distributed \( F_2 \rightarrow C_2 \) mapping, which is what RAM has.

RAM, on the other hand, uses a simple match tracking operation --- similar to the one originally proposed by Carpenter, Grossberg, and Reynolds (1991) --- in which \( \rho \) is raised until match at the \( C_2 \) layer is obtained. RAM avoids category proliferation primarily by relaxing the match criterion at the \( C_2 \) layer. Equation (6) allows resonance if the bottom-up signal to the correct class node is strong enough (as determined by \( \rho^{(z)} \)), even if it is not the maximal bottom-up signal.

5.4 Learning

GAM’s receptive field parameters, \( \mu \) and \( \sigma \), are updated via discrete instar equations that are similar to equation (12), the equation for updating RAM’s weight fields. However, the effect of these update equations is very different for GAM because the meaning of the parameters is different. GAM’s parameters encode the sample mean and standard deviation with which to define its Gaussian receptive field, even though the underlying data may not be Gaussianly distributed. RAM’s weight fields, on the other hand, encode the sample average of its input distributions. Figure 8 depicts an extreme case wherein, given the same set of training inputs, the two networks learn completely different representations. In Figure 8, bimodally distributed inputs (top) lead to a bimodal RAM receptive field (middle) but a unimodal GAM receptive field (bottom). In practice, however, RAM typically learns unimodal receptive fields that are similar to GAM’s.
Gaussian receptive fields.

6 Classification Benchmarks

6.1 Methodology

All classification databases were preprocessed as follows. For RAM, the input vectors from the training and testing sets were normalized so that each feature value was in the range of [0:1]. Next, the feature values were mapped into $F_i$ input distributions via equation (1) with $\sigma_i$ equal to the standard deviation of values in the $i^{th}$ dimension of the training set. For GAM, the feature values were normalized to have a unit standard deviation in each dimension. GAM’s results on three of the following classification benchmarks have been reported elsewhere (Williamson, 1997; Grossberg and Williamson, 1998a).

The performance of both networks depends upon parameters that bias the spatial scale of the receptive fields they form. In RAM, the key parameter is $\alpha$, which scales the width of the $F_i$ distributions in equation (1). In GAM, the key parameter is $\gamma$, which defines a newly committed category’s initial standard deviation in each dimension. The effects of the scale parameters are similar for RAM and GAM on all of the benchmarks. On the first two benchmarks, which are small, the networks perform best with large scale parameters ($\alpha = 0.75$ for RAM, $\gamma = 4.0$ for GAM), whereas on the last two benchmarks, which are large, the networks perform best with small scale parameters ($\alpha = 0.375$ for RAM, $\gamma = 1.0$ for GAM). As was discussed in Section 3, RAM uses $L = 7$ with $\alpha = 0.75$ and $L = 10$ with $\alpha = 0.375$.

RAM and GAM are both sensitive to the order that training inputs are presented. Therefore, in order to obtain reliable classification measures, the results were averaged over several different training runs, with independently randomized orderings in each epoch. On the first two (small) datasets, the results of 25 runs were averaged, and on the other two (large) datasets, the results of 5 runs were averaged.

6.2 Speaker Independent Vowel Classification

In Williamson (1997), GAM was evaluated on a speaker independent vowel classification benchmark (Deterding, 1989), which is archived in the CMU connectionist benchmark collection (Fahlman, 1993). Using a large scale parameter ($\gamma = 4.0$), GAM outperformed a similar EM mixture modeling classifier, as well as several standard neural network classifiers, on this task. Here, RAM is evaluated on the same data set, also using a large scale parameter ($\alpha = 0.75$). 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, seven of whom were female and eight male. 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
Figure 8: Given bimodally distributed inputs (top), a RAM category learns a bimodal receptive field (middle), whereas a GAM category learns a unimodal receptive field (bottom).
Spoken Vowel Classification

Figure 9: Results of RAM and GAM, using large scale parameters, on the spoken vowel benchmark. Classification rate (top) and number of committed categories (bottom) are plotted as a function of the number of training epochs.
Each speaker thus 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.

Figure 9 (top) shows the average classification results of RAM and GAM on the vowel recognition benchmark as a function of the number of training epochs. RAM’s classification rate is plotted, with error bars, for epochs 1, 2, 5, 10, 15, ..., 60, while GAM’s classification rate, without error bars, is plotted for every epoch. RAM and GAM learn at about the same rate. GAM’s best performance of 56.0% correct is obtained after 6 epochs, after which its performance declines significantly, to about 51% after 60 epochs. RAM outperforms GAM throughout training, achieving its best performance of 59.9% correct after 10 epochs. Unlike GAM, RAM’s decline due to overtraining is small, with good performance (57.6% correct) maintained even after 60 training epochs. The primary reason that RAM suffers less than GAM from overtraining is that its input distributions define the minimum width that its receptive fields can form (e.g., see Figure 6), thereby preventing the receptive fields from shrinking too much around the training samples and overlearning the training data. Figure 9 (bottom) shows that RAM commits more categories than GAM, a trend that holds on all the following benchmarks as well. The number of categories that RAM commits can be reduced by lowering $\rho^{(z)}$, although this may result in a loss of accuracy.

RAM was also evaluated on this benchmark using 2-D input fields, as in Section 4.3. This was done by combining pairs of the 10 1-D input fields, via outer products, into 5 2-D input fields. The dimensions that were least correlated with each other were paired up, in order to make maximal use of the 2-D feature spaces. This 2-D variation of RAM obtained similar, albeit slightly worse, results compared to 1-D RAM. Specifically, 2-D RAM obtained 59.5% correct using 56.7 categories after 10 epochs, and 55.8% correct using 63.4 categories after 60 epochs.

### 6.3 Waveform Classification

Using the same parameters as above, RAM and GAM were also tested on another small classification dataset, the waveform benchmark (Breiman et al., 1984), which is archived in the UCI machine learning repository (King, 1992). The problem is to map 21 real-valued input features into one of three output classes. The feature values for each class were generated by a random convex combination of two out of three basis functions, plus a random number. There is also a noise variation of this problem, in which 19 additional input dimensions, containing only random numbers, were added. For details, see pp. 49–55 of Breiman et al. (1984).

This benchmark has previously been evaluated under two conditions. Breiman et al. (1984) trained both a decision tree algorithm called CART, and a nearest-neighbor (NN) classifier, on 300 independent training samples and then tested them on the 5,000 samples that make up the database. Miller et al. (1996) have also reported the results of four RBF techniques after training on the first 2,500 samples, and then testing on the remaining 2,500 samples, of the database.

RAM and GAM were evaluated on both the clean (21-D) and noisy (40-D) data sets, after partitioning the database into both small (300) and large (2,500) training sets. For brevity, only the results obtained after 10 training epochs are reported. Figure 10 shows that RAM outperforms
Figure 10: Results of RAM and GAM, using large scale parameters, on the waveform benchmark. Classification rate (top) and number of committed categories (bottom) after 10 training epochs are plotted as a function of the size of the training set, for both the clean (21D) and noisy (40D) data sets.
GAM on all four variations (top), while committing more categories than GAM (bottom).

On the small training sets RAM outperforms CART and NN by a wide margin on both the clean variation (RAM: 83.9%; CART: 72%; NN: 78%) and the noisy variation (RAM: 80.2%; CART: 72%; NN: 38%). On the large training sets, both clean and noisy variations, RAM also easily outperforms the three standard RBF algorithms reported in Miller et al. (1996), regardless of the number of basis functions that were used. The fourth RBF technique is a sophisticated global optimization algorithm using deterministic annealing. RAM outperforms the deterministic annealing algorithm on the clean benchmark (86.6% versus 84% correct) and performs equally well on the noisy benchmark (86.9% versus 87% correct).

6.4 Letter Image Classification

In Williamson (1997), GAM was evaluated on a letter image classification task (Frey and Slate, 1991), which is archived in the UCI machine learning repository (King, 1992). Using a small scale parameter ($\gamma = 1.0$), GAM outperformed a similar EM mixture modeling classifier on this task, as well as several other classifiers. Here, RAM is evaluated on the same data set, also using a small scale parameter ($\alpha = 0.375$). The data consist 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 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 4,000 for testing.

Figure 11 (top) shows the average classification results of RAM and GAM on the letter image recognition benchmark as a function of the number of training epochs. RAM's classification rate is plotted, with error bars, for epochs 1, 2, 5, 10, 15, ..., 80, while GAM's classification rate, without error bars, is plotted for every epoch. Unlike the results on the first two benchmarks, here the two networks learn at very different rates. GAM's best performance of 94.6% is obtained after only 6 epochs, after which its performance declines to about 92.6% after 80 epochs. RAM, on the other hand, learns more slowly, requiring about 70 epochs to reach its best performance of 94.1% correct, after which its performance stabilizes. As in the first two benchmarks, RAM commits more categories than GAM (Figure 11, bottom).

The difference in RAM's and GAM's learning rates on this problem illustrates an important distinction between the two networks. In RAM, one parameter biases receptive field size and another parameter determines the learning rate. In GAM, on the other hand, both of these functions are, in effect, controlled by the same parameter. A RAM category begins with uniform weight fields, which are combined (via averaging) with the current input distributions during learning (see equation (12)). The width of the input distributions is determined by $\alpha$ (equation (1)), whereas the averaging coefficient is determined by $\beta$ (equation (11)). A GAM category begins with an initial width of $\gamma$ in its Gaussian receptive field. Therefore, $\gamma$ plays a role similar to $\alpha$ because it biases the receptive field size. However, reducing $\gamma$ also has an effect similar to reducing $\beta$, because it speeds up the process by which a receptive field shrinks to any particular size by making it start.
Figure 11: Results of RAM and GAM, using small scale parameters, on the letter image benchmark. Classification rate (top) and number of committed categories (bottom) are plotted as a function of the number of training epochs.
6.5 Natural Texture Classification

In Grossberg and Williamson (1998a), GAM, as part of a biologically motivated architecture for processing and classifying image regions, was evaluated on a natural texture classification task. The image processing front-end extracted, from each 8x8 pixel region of the input image, 16 oriented contrast features (four orientations and four spatial scales) as well as a single brightness feature. GAM was trained to use these 17-dimensional feature vectors to classify natural textures from the Brodatz album (Brodatz, 1966). For each texture, GAM was trained on three images (768 samples), and tested on a fourth image (256 samples). GAM was evaluated in this way on 6, 12, 18, ..., 42 textures. As the number of textures increased, and the classification problem became more difficult, GAM’s classification rate and category allocation (per texture) remained stable. Figure 12 summarizes GAM’s results, using $\gamma = 1.0$, obtained after two training epochs. On the whole, GAM’s performance did not improve with further training. As reported in Grossberg and Williamson (1998a), these results are superior to those obtained on a similar texture classification task by an alternative image classification architecture that used rule-based, multilayer perceptron, or $k$-nearest neighbor classifiers. GAM also outperformed this alternative architecture when they were both evaluated on an identical task involving the classification of 10 natural textures.

As with the results on the Letter Image Recognition benchmark reported above, RAM obtains comparable results to GAM, but requires more training epochs to do so. RAM obtains comparable results after about 15 training epochs, but continues to improve slightly with more training. For brevity, Figure 12 only shows RAM’s results after 35 training epochs, when its performance more or less stabilizes. While RAM and GAM obtain similar results on all seven variations, Figure 12 reveals an interesting trend: RAM does better than GAM when the number of textures is small, but worse when the number of textures is large. This latter difference may reflect RAM’s inability to form arbitrarily small decision regions in the input space, as illustrated in Figure 6, which could degrade its performance when the input space becomes crowded with more and more classes that need to be discriminated. Figure 12 shows that RAM commits more categories per texture than GAM on all seven variations.

7 Memory Requirements

Section 6 shows that RAM performs well on several classification databases, and on the whole outperforms GAM. However, RAM’s memory requirements are a concern because: (i) RAM commits more categories than GAM on all the benchmarks, and (ii) RAM uses more weights, or parameters, per category than GAM does.
Natural Texture Classification

Figure 12: Results of RAM and GAM, using small scale parameters, on the natural texture benchmark. Classification rate (top) and number of committed categories per texture (bottom) are plotted, after 2 training epochs for GAM and 35 training epochs for RAM, as a function of the number of textures.
7.1 Number of Committed Categories

One reason that RAM commits more categories than GAM may be that, unlike GAM, prior probabilities are not factored into RAM's category activations in equation (3). Thus, more RAM categories may be needed because each one encodes less information. To explore this possibility, an alternative version of RAM was tested in which the count variable is factored into the activation equation:

\[ y_j = n_j [g_j - \rho T_j]^+ \quad \text{(19)} \]

This alternative RAM network was evaluated on the spoken vowel benchmark with 10 training epochs. While the alternative RAM network commits fewer categories, its performance is nowhere near that of the normal RAM network (49.7% correct using 51.4 categories versus 59.9% correct using 57.4 categories).

Figure 13 provides some insight into why normal RAM outperforms alternative RAM. Figure 13 (top) shows histograms of the count variables, \( n_j \), accumulated over all 25 runs. The histograms from individual runs are similar to these cumulative histograms. Figure 13 shows that most \( n_j \) variables are clustered around a common value in normal RAM, whereas they are spread out in the alternative RAM. Therefore, because it does not factor \( n_j \) into its activation equation, RAM's categories learn similar prior probabilities and thereby efficiently partition the input space. Because the categories have similar, low probabilities, they comprise a sparse distributed code (Field, 1994).

What is the effect on its receptive fields of alternative RAM's skewed prior probabilities? Figure 13 (bottom) provides indirect evidence about this by showing histograms of the match threshold, \( T_j \). Because \( T_j \) tracks the average match value of its category, it is an indicator of the shape of the category's receptive field: a small \( T_j \) indicates a wide receptive field, and a large \( T_j \) indicates a narrow receptive field. Figure 13 (bottom) therefore shows that normal RAM produces a much greater proportion of wide receptive fields than alternative RAM, which may help explain why it generalizes better on the test data.

Another factor that may explain why RAM commits more categories than GAM is that it uses a different match tracking algorithm. Relaxing the class match criterion, by lowering \( \rho(z) \) below its current value of \( \rho(z) = 0.85 \), causes RAM to commit fewer categories. Our simulations have shown that lowering \( \rho(z) \) often significantly reduces the number of committed categories, while only modestly reducing classification performance. Alternatively, one can define a ceiling on the number of committed categories, \( N_{\text{max}} \), so that once this number is reached, \( \rho(z) \) is set to zero, thereby automatically preventing the commitment of more categories. Even better is to gradually reduce \( \rho(z) \) as the number of committed categories approaches the ceiling, via (for example):

\[ \rho(z) = \frac{N_{\text{max}} - N}{N_{\text{max}}} \quad \text{(20)} \]
Figure 13: Top: histograms of the count variables, $n_j$, after 10 training epochs of normal RAM (left) and alternative RAM (right), on the spoken vowel benchmark (accumulated over 25 runs). Bottom: histograms of the match thresholds, $T_j$, for normal RAM (left) and alternative RAM (right).
Figure 14: Average distribution of sorted class weights obtained by RAM in the spoken vowel benchmark after 10 training epochs.

7.2 Number of Weights per Category

Even though RAM creates more categories than GAM, the difference is generally not very large. A potentially bigger problem is the number of weights, or parameters, that it uses per category. A GAM category uses only two parameters per input dimension (mean and standard deviation), and only a single connection to an output class. A RAM category, on the other hand, can use as many as \( L \) weights per input dimension (\( L = 7 \) in the large-scale case, and \( L = 10 \) in the small-scale case), and \( C \) class weights (where \( C \) is the number of output classes). Fortunately, RAM requires only a small subset of these weights to obtain good performance, as illustrated below.

**Number of Class Weights.** Although a category is initialized with \( C \) uniformly distributed class weights, this distribution quickly changes to one that is dominated by a single strong connection. For example, Figure 14 shows the average distribution of sorted class weights obtained in the spoken vowel benchmark. In general, pruning all but one or two class weights per node has little effect on performance.

**Number of Weight Field Weights.** Good performance is also maintained if the majority of weight field weights are pruned. Figure 15 illustrates this result by plotting classification performance on two benchmarks as a function of the number of non-pruned weights. Figure 15 (top) shows that the classification rate on the (large scale) spoken vowel benchmark remains very good with 4.5 weights per dimension, and reasonably good with only 3.0 weights per dimension.
Spoken Vowel Classification

Letter Image Classification

Figure 15: RAM's classification rate on the spoken vowel benchmark following 10 training epochs (top) and on the letter image benchmark following 100 training epochs (bottom). Classification rates are plotted as a function of the number of positive weights per dimension, that is, the average number of positive weights in each vector, $w_{ji}$, as the smallest weights are pruned.
Figure 15 (bottom) shows that similar results obtain in the (small scale) letter image recognition benchmark (after 100 training epochs). Due to pruning during training (see Section 3.5), RAM’s weight fields contain only about 5 non-pruned weights per dimension. Further pruning to 3 weights per dimension has virtually no effect on RAM’s performance.

Even with pruning, RAM’s memory requirements are still higher than those of GAM. This difference can be accepted as the price of biological plausibility. Therefore, it is fitting to compare RAM with another network, Fuzzy ARTMAP (FAM), which is also biologically plausible. Like GAM, FAM can be implemented algorithmically using only 2 parameters in each dimension, representing the minimum and maximum value (Carpenter, Grossberg, and Rosen, 1991a). However, a neurally plausible implementation of FAM involves 4 weights per dimension, 2 in bottom-up $F_1 \rightarrow F_2$ connections, and 2 in top-down $F_2 \rightarrow F_1$ connections (Carpenter, Grossberg, and Rosen, 1991b). FAM obtains 51.1% correct using 66 categories on the spoken vowel benchmark, and 91.9% correct using 1,035 categories on the letter image benchmark (Williamson, 1996). Therefore, with pruning, RAM performs much better than FAM on both benchmarks while using fewer weights.

8 Topographic RAM

Thus far, all of our simulations have used unimodal distributions in the $F_1$ fields. This is not necessary, and further simulations are needed to explore learning with multimodal $F_1$ distributions. RAM only requires that $F_1$ distributions contain local correlations, so that $F_2$ learns smooth receptive fields. Local correlations result from topographic structure, in which mutual excitation between nearby cells causes them to learn correlated receptive fields. Since the only requirement for categorization in $F_2$ is the presence of topography in $F_1$, then, similarly, the only requirement for categorization in a higher $F_3$ layer is the presence of topography in $F_2$.

Obtaining topography in $F_2$ requires center-surround interaction kernels as shown in Figure 16a, which induce correlations among nearby cells and decorrelation among cells spaced further apart. Center-surround kernels of this sort are the essential ingredients for topographic map formation, such as in models for self-organization of coordinated maps of ocular dominance and orientation preference (Durbin and Mitchison, 1990; Obermayer et al., 1992; Swindale, 1992; Sirosh and Miikkulainen, 1994, Olson and Grossberg, 1998). It is beyond the scope of this article to specify the implementational details for topographic map formation. It is instead assumed that center-surround kernels such as in Figure 16a induces topography in the $F_2$ layer, yielding a topographic RAM, or T-RAM, network.

Another important property of cortical representations is spatial structure. Sensory inputs comprise a spatial map, such as a retinotopic map in vision, in which position in the map encodes an important “feature” of the world, namely, spatial location in the visual field. At each location, visual input is analyzed into a set of local features within a hypercolumn. A RAM network has no concept of spatial structure, and therefore is analogous to a single hypercolumn. To encode spatial structure, a T-RAM network needs an $F_2$ map consisting of multiple contiguous $F_2$ fields, each receiving input from local, partially overlapping regions of the input maps. All of the
Figure 16: a) Center-surround kernels that can induce topography within a layer. b) Imposing topography allows for the development of a hierarchical RAM network. Reciprocal, top-down connections could be useful for priming or modulating lower-level representations. See text for details.
operations described in Section 2 should thus be localized to a limited region of this overall $F_2$ map. In particular, subtractive vigilance-based inhibition (equation (3)) and divisive normalizing inhibition (implicit in equation (8)) should only extend within a local neighborhood of T-RAM “hypercolumns”.

Figure 16b illustrates how topography and spatial structure make possible a hierarchical representation consisting of one or more fields at each level of the hierarchy. Multiple fields in one level could correspond to different topographic maps, or to different sections of the same map. Due to the limited spatial extent of inhibition, it is reasonable to approximate a single map with multiple independent fields. Figure 16b also shows the presence of reciprocal, top-down connections that can prime or modulate lower-level representations. Each $F_x$ field is linked, via symmetrical connections, to a $C_x$ field which sends predictive feedback which is appropriate given the type of features coded at the $F_x$ field. Each $F_x$ field may even be linked to several $C_x$ fields. In this case, behavioral context would determine which of those fields provides feedback to the $F_x$ field, triggering learning. If a $C_x$ field is embedded in its own hierarchy, it would be appropriate for it to be topographic as well. In this case, match at $C_x$ would depend on a different measurement than equation (6), such as the amount of overlap between the $F_x$--induced activity pattern and the supervised signal.

### 8.1 Perceptual Learning

As a general model for learning sensory/perceptual cortical hierarchies, T-RAM should be able to explain perceptual learning data in different cortical areas, and at different cortical levels. Behavioral relevance is critical for perceptual learning in visual cortex (Shiu and Pashler, 1992; Ahissar and Hochstein, 1993; Herzog and Fahle, 1994; Karni and Sagi, 1995), auditory cortex (Recanzone et al., 1993), and somatosensory cortex (Recanzone et al., 1992a–d). In particular, it has been found that identical visual stimuli in the context of different behavioral tasks lead to different perceptual learning effects (Ahissar and Hochstein, 1993; Karni and Sagi, 1995). In order to model perceptual learning effects, correspondences need to be specified between T-RAM’s architecture and attentional processes that encode behavioral relevance. We propose that “supervised” signals in $C_x$ fields correspond to such attentional processes. Learning at $F_x$ only takes place if it induces a match with the supervised signal in its $C_x$ field. See Sections 3.3–3.5 for details.

However, T-RAM currently provides no explanation for how the appropriate featural dimensions are selected. That is, it has no proposed mechanisms for selecting which $C_x$ fields will receive supervised signals. This is a subject for future research, which needs to take into account the following data. The capacity for selecting appropriate feature dimensions improves with age (Goldstone, 1998), and the selection of these dimensions appears to account for fast improvement in initial stages of a behavioral task, particularly when subjects are first exposed to easily discriminated stimuli (Mackintosh, 1974; Karni and Sagi, 1995). Ahissar and Hochstein (1997) showed that the degree of specificity of perceptual learning effects depends on the difficulty of training. For example, they found that increasing the orientational difficulty of the training task increases both the orientational and positional specificity of perceptual learning. Similarly, increasing the positional difficulty induces greater orientational specificity. Based on these results Ahissar and
Hochstein proposed that learning proceeds as a top-down countercurrent along the cortical hierarchy, with learning induced first in the highest levels and then proceeding to lower levels as required by the demands of the training task. Similarly, easy-to-difficult training schedules may help guide attention down the correct path of a T-RAM hierarchy so that it can focus on the appropriate \( C_x \) field(s).

Another complicating factor is the role of environmental feedback. Herzog and Fahle (1997) found that (i) environmental feedback as to the accuracy of responses improves discrimination performance on a vernier acuity task, but that (ii) providing block feedback (e.g., \( x\% \) correct after every \( n \) trials) yields the same improvement as trial-by-trial feedback. Based on (i), Herzog and Fahle (1998) concluded that unsupervised learning models are insufficient, and based on (ii) they concluded that standard supervised learning models are insufficient. They instead proposed a model in which top-down attentional feedback gates the flow of bottom-up information.

T-RAM also allows for the possibility that attentional priming from \( C_x \rightarrow F_x \) could influence the bottom-up processing from \( F_{x-1} \rightarrow F_x \). This priming could sharpen \( F_x \) distributions and thereby model data about restriction of receptive fields caused by focused attention (Moran and Desimone, 1985). However, this is not required for T-RAM’s explanation of Herzog and Fahle’s data, which is as follows. The task in question requires the subject to select the direction of vernier offset. Therefore, suppose there are two \( C_x \) nodes corresponding to each direction of vernier offset, whose activities are denoted by \( z_1 \) and \( z_2 \). The \( F_x \) field would generate an initial pattern in these nodes, but this pattern would lead to a behavioral response only if either \( z_1/(z_1+z_2) > \rho^{(z)} \) or \( z_2/(z_1+z_2) > \rho^{(z)} \). Otherwise, match tracking would occur, sharpening the \( F_x \) receptive fields and thereby refining their representation when learning takes place. Environmental feedback would influence the subject’s choice of \( \rho^{(z)} \). Increasing \( \rho^{(z)} \) would tend to reduce predictive errors but at the cost of cortical resources. Since \( \rho^{(z)} \) would be adjusted incrementally, and therefore essentially integrate the number of feedback errors, it is not surprising that block feedback and trial-by-trial feedback would be equally effective in improving performance.

T-RAM offers similar explanations for other perceptual learning data. All of these explanations rely on the same basic idea: raising vigilance sharpens receptive fields, and this sharpening is stored via learning. Figure 17 illustrates how, during learning, the presence of higher average vigilance levels at certain regions of a perceptual dimension leads to narrower receptive fields, which in turn lead to perceptual categorical effects: better discriminability in the boundaries between perceptual categories, and lower discriminability in the interiors of perceptual categories (Goldstone, 1998).

Note that, regardless of their width, the receptive fields in Figure 17 overlap each other by the same amount. This is because, in topographic map formation, the fixed extent of excitatory/inhibitory kernels (see Figure 16) tends to fix the level of activity correlation between neighboring cells. Therefore, the spacing between receptive field centers will tend to vary with the receptive field widths.

Many studies have found that perceptual learning leads to the narrowing of receptive fields and/or the expansion of cortical representation. Recanzone et al. (1993) found that training monkeys to discriminate sound frequencies led to both larger cortical representations for those frequencies and narrower tuning of the frequency-sensitive receptors. Similarly, Jenkins et al.
Average Vigilance Level During Learning

---

Receptive Fields

---

Perceptual Category Boundary

Perceptual Category Interior

---

Perceptual Dimension

---

Figure 17: T-RAM's explanation for categorical perception effects. The width of receptive fields is determined by the average vigilance level during learning. When vigilance is high, receptive fields become sharpened. Topographic map formation processes cause the spacing between receptive field centers to be proportional to receptive field width, yielding a greater cortical representation and greater perceptual discriminability at category boundaries.
(1990) found that training monkeys on a task requiring them to regulate the contact of one or two fingertips on a revolving, grooved, disk resulted in a several-fold increase in the cortical representation of the surface of digit tips, accompanied by a corresponding decrease in the size of receptive fields. Saarinen and Levi (1995) found, using an orientation masking paradigm, that improvement on a vernier acuity task was closely matched with narrowing of tuning for the orientation sensitive mechanisms underlying vernier offset detection.

T-RAM offers the following explanation for these perceptual learning data, which specifies a clear causal relationship between the sharpening of tuning properties and the expansion of cortical representation. Perceptual learning produces a sharpening of tuning properties by triggering learning with a higher-than-normal vigilance level, and this in turn causes an expansion of cortical area devoted to the relevant region of the input space, since map formation processes cause the receptive fields of nearby cells in the map to be “pulled” toward those of the sharpened cells.

The reverse process can also occur. For example, Hussain and Guenther (1998) found that, while training humans on an auditory discrimination task results in an improvement in discrimination performance, training them on an auditory categorization task results in a decrease in discrimination performance within the interiors of the perceptual categories. As Figure 17 indicates, the T-RAM explanation for these results is that discriminability within a category interior is reduced because training on the categorization task triggers learning with a lower-than-normal vigilance level, resulting in larger-than-normal receptive fields and a contraction of the cortical representation. Therefore, T-RAM predicts that increasing the difficulty of the categorization task by a sufficient amount would produce narrower receptive fields and a larger cortical representation, just as in the discrimination task.

Categorical perception is also found in high-level perceptual dimensions. For example, Beale and Keil (1995) found increased sensitivity to differences at the half-way point between familiar faces, but not between unfamiliar faces. Other data about face representation are also consistent with T-RAM. Busey and Tunnicliff (submitted) found that when two similar, studied faces are morphed together, the resulting “child” face is often perceived as being more familiar than either of its parents. However, this effect is not found when the child is produced by two dissimilar parents. Dailey et al. (1998) modeled these results with a Gaussian mixture model in which each studied face is represented with a Gaussian distribution in the input space. Their model matched Busey and Tunnicliff’s data only if distinctive faces were represented with Gaussians that were both wider and higher than those for nondistinctive faces.

T-RAM should produce representations consistent with these results. When T-RAM learns different categories for similar stimuli, it will tend to produce narrow receptive fields because the stimuli will be more likely to produce incorrect predictions and thus raise vigilance, and also because the learning rate will be reduced where there is receptive field overlap. When T-RAM categorizes distinctive stimuli, on the other hand, it will tend to produce wide receptive fields, since both vigilance and receptive field overlap will be low. Moreover, because T-RAM does not represent prior probabilities, its receptive fields for distinctive stimuli, in undersampled regions, will have similar heights to those for similar stimuli, in highly sampled regions. Therefore, T-RAM’s distinctive receptive fields will in effect be higher than those obtained by density estimation approaches.
8.2 Comparison to Laminart Model of Visual Cortex

T-RAM has many similarities to a detailed model of the laminar circuits in visual cortex called Laminart. Laminart provides a functional explanation of how the laminar circuits in V1 and V2 give rise to psychophysical data about perceptual groupings (Grossberg, Mingolla, and Ross, 1997; Grossberg, 1998; Grossberg and Williamson, 1998b; Ross, Grossberg, and Mingolla, 1998). In particular, Grossberg and Williamson (1998b) proposed a developmental model which explains how Laminart’s excitatory and inhibitory connections self-organize into an adult network capable of context-sensitive grouping. In comparing the models, it should be noted that a cortical column in Laminart, which consists of nodes in three different layers, is analogous to a single T-RAM category node. For brevity, these three layers are treated herein as a single entity, a level of detail which is sufficient for our present purposes.

In Laminart, learning of inhibitory connections among nearby columns stabilizes development and allows competition between alternative local boundary hypotheses. Inhibition takes similar forms and plays similar roles in Laminart and in T-RAM. Both models use a combination of subtractive and shunting (or divisive) inhibition within a recurrent network. In T-RAM, local subtractive inhibition implements the match rule, which decides if the input pattern is close enough to the category’s receptive field — given the current vigilance level — for the category to be active, and hence, to learn. Subsequently, shunting inhibition normalizes local activities, so that each category’s learning rate is proportional to its share of the total credit. Laminart also uses subtractive and shunting inhibition to determine final activity levels, which regulate excitatory instar learning and inhibitory outstar learning.

The most striking difference between the models is that, in Laminart, long-range, horizontal connections are learned between cortical columns that have similar orientation tuning (Grossberg and Williamson, 1998b). These connections allow the network to group boundaries, integrating local boundary hypotheses over large extents of the visual field. Selective, long-range, horizontal connections are ubiquitous throughout cerebral cortex, not just in visual cortex (Lund et al., 1993). Therefore, an area for future research is the learning of horizontal connections between different fields in the same level of a T-RAM hierarchy. These connections would exploit the spatial structure of T-RAM’s representations, which would be exhibited by activity correlations between nearby fields.

Another difference between the two networks is the role of excitatory top-down connections between levels. Reciprocal top-down connections are the rule, rather than the exception, in all of cortex (Felleman and Van Essen, 1991). Grossberg (1998) proposed two roles for top-down feedback: (i) to prime or reinforce local representations based on more global information, and (ii) to stabilize learning using an ART matching rule.

Reason (i) is consistent with the T-RAM network. For example, learned top-down connections from $F_2$ to $F_1$, as in Figure 16b, can generate expectations of the distributions at $F_1$ fields based on the more global information that is available to $F_2$. This flexibility in the direction of processing is one of the major advantages of using mixture models to represent the joint I/O density (Ghahramani and Jordan, 1994). Similarly to horizontal connections within the same level, feedback signals between levels would help the lower level integrate information over larger extents of
the input space (e.g., the visual field).

8.3 Instar Match and Outstar Match

This brings us to the second reason given by Grossberg (1998) for top-down connections: to stabilize learning by matching top-down and bottom-up signals. This is not consistent with T-RAM. To explain why this is, it is helpful to distinguish between two different types of match computations: *instar match* and *outstar match*. The $F_2$ match computation in equation (3) is an instar match: it is local and analytic, evaluating independently the match between a single $F_2$ receptive field and the pattern at $F_1$. The only interdependence between different $F_2$ categories in equation (3) is an indirect one, stemming from their collective effect on the vigilance parameter, $\rho$, via their connections to $C_2$ in equation (4).

The match at $C_2$ in equation (6), on the other hand, is an outstar match: it is nonlocal (from $F_2$’s point of view) and synthetic, based on a comparison between the $C_2$ activity pattern generated by all the active $F_2$ nodes, and the supervised signal at $C_2$. Note that both the instar and the outstar match computations evaluate the representation at $F_2$. Instar match evaluates if a single $F_2$ category is suitable for representing the input, and outstar match evaluates if the entire $F_2$ representation is suitable in terms of the output prediction it generates.

Traditional ART and ARTMAP networks use only outstar match (Grossberg, 1980; Carpenter and Grossberg, 1987; Carpenter, Grossberg, and Rosen, 1991b). Specifically, $F_2$ sends a top-down signal containing its expectation of the $F_1$ pattern. The network resonates only if this top-down signal is sufficiently close to the existing pattern at $F_1$. With a winner-take-all representation in $F_2$, outstar match becomes functionally identical to instar match because each $F_2$ node’s match is evaluated in isolation after it wins the competition. However, with a distributed representation in $F_2$, outstar match can only be evaluated holistically due to the superposition of the feedback templates from all the active $F_2$ nodes. This is problematic due to the possibility of cases in which all the $F_2$ receptive fields match the $F_1$ pattern poorly, but the cumulative $F_2 \rightarrow F_1$ feedback signal matches the $F_1$ pattern very well. See Carpenter (1997) for a different approach to overcoming this problem of distributed representation in ART and ARTMAP networks.

9 Concluding Remarks

This article presents a simple and straightforward approach for incorporating predictive feedback into self-organizing networks. The result is a biologically plausible supervised learning network, RAM, which compares favorably to alternative classifiers on several benchmarks. The simplicity of RAM’s learning rules owes to the fact that it directly encodes into its receptive fields the smoothness of activity distributions in topographic maps. RAM thereby obtains a regularized representation that generalizes well.

One example of smoothness in cortical maps is the property of orientation preference in primary visual cortex, which shifts gradually as cell recordings are moved parallel to the cortical surface (Hubel and Weisel, 1962, 1963, 1968). Analogously, orientation preference in RAM would be coded
in an $F_1$ field with a smooth distribution over a set of nodes, each of which successively represents a different, nearby orientation. This distribution would convey both a precise orientation value (via coarse coding), and a precise level of orientational variance, or uncertainty, due to its width. $F_2$ categories would directly learn this information via simple, correlational learning rules.

An attractive feature of the proposed topographic RAM, or T-RAM, network, is that it is a general purpose model for learning at any level of a cortical hierarchy. The same learning rules can be used to self organize smooth maps of orientation preference at the lowest level of the hierarchy (e.g., primary visual cortex), or smooth maps of object categories at the highest level of the hierarchy (e.g., IT cortex). T-RAM’s learning rules would produce, in each level of the hierarchy, sparse distributed codes in a topographic map that categorize input from lower levels using smooth receptive fields, which are refined based on predictive feedback.

In fact, receptive fields in IT cortex exhibit many similarities with those in primary visual cortex. As in primary visual cortex, receptive fields in nearby columns of IT cortex are highly correlated with each other (Wang et al., 1996). Therefore, even though IT cells respond to complex feature combinations, they exhibit generalization gradients similar to those of primary cortical cells. For example, the tuning curves obtained from object-selective IT cells as a function of object rotation are similar to the tuning curves obtained from striate neurons as a function of line rotation (Logothetis and Pauls, 1995).

IT tuning curves also closely parallel the generalization gradients found in human priming experiments (Logothetis and Sheinberg, 1996). Perceptual priming seems to be related to repetition suppression, in which many IT neurons exhibit a reduced response, but some an enhanced response, to the repeated presentation of the same stimulus (Wiggs and Martin, 1998). Human neuroimaging studies have shown an association between perceptual priming and decreased neural activity (Schacter and Buckner, 1998), and therefore support a link between perceptual priming and repetition suppression. One explanation of these phenomena is that repeated experience with a novel object leads to a sharpening of its representation in cortex (Desimone, 1996). This is consistent with T-RAM’s sharpening of receptive fields, and hence of representational activity patterns.

IT neurons typically respond to views of behaviorally relevant objects, such as faces. They can also become tuned to novel stimuli. After several months of training with different views of novel wire-like and amoeboid objects, IT neurons responded preferentially to particular views of the objects (Logothetis and Pauls, 1995). Some neurons responded to a whole stimulus, and some just to simple features, such as an angle between two wire segments. The frequency of encountering neurons selective to a particular object type corresponded to the animal’s familiarity with the object class. Rapid learning in IT cortex has also been demonstrated. When initially exposed to binarized faces, face-sensitive cells gave little response, but after the animal was given a few seconds of viewing gray-scale versions of the same faces, the cells responded equally to the binarized images (Tovee et al., 1996).

Modeling these learning effects in IT neurons is an important area for future research with T-RAM networks. This is particularly true because of strong experimental support for view-based models of object recognition (summarized in Edelman, 1998; Logothetis, 1998; Treisman and Kanwisher, 1998). View-based computational models obtain invariance to 3-D rotation by
using smooth receptive fields to interpolate between multiple categories that represent different memorized 2-D views (Poggio and Edelman, 1990; Edelman and Poggio, 1992; Bulthoff et al., 1995; Edelman, 1995; Vetter et al., 1995). These models typically use prespecified sets of radial basis functions that learn via biologically implausible computations. T-RAM, on the other hand, suggests how hierarchical circuits with appropriate size and complexity can develop using simple, biologically plausible mechanisms.
References


