1999-10

Self-Organization of Topographic Mixture Networks Using Attentional Feedback

https://hdl.handle.net/2144/2244

Boston University
Self-organization of topographic mixture networks using attentional feedback

James R. Williamson

October, 1999

Technical Report CAS/CNS-99-027
SELF-ORGANIZATION OF TOPOGRAPHIC MIXTURE NETWORKS
USING ATTENTIONAL FEEDBACK

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

Submitted: October, 1999
Technical Report CAS/CNS TR-99-027

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

This paper proposes a biologically-motivated neural network model of supervised learning. The model possesses two novel learning mechanisms. The first is a network for learning topographic mixtures. The network's internal category nodes are the mixture components, which learn to encode smooth distributions in the input space by taking advantage of topography in the input feature maps. The second mechanism is an attentional biasing feedback circuit. When the network makes an incorrect output prediction, this feedback circuit modulates the learning rates of the category nodes, by amounts based on the sharpness of their tuning, in order to improve the network's prediction accuracy. The network is evaluated on several standard classification benchmarks and shown to perform well in comparison to other classifiers. Possible relationships are discussed between the network's learning properties and those of biological neural networks. Possible future extensions of the network are also discussed.

1 Introduction

Recent years have seen significant advancement in the understanding of statistical learning properties of artificial neural networks (ANNs). However, ANNs are usually unconstrained from an implementational point of view. Applying constraints which are motivated by biological implementability, while making a statistical analysis more difficult, may offer the compensatory advantage of producing models that have more straightforward links with biological systems. In addition, the constraints may provide certain practical advantages, as described below. It is with these motivations that the following three constraints were used to guide neural network development:

1. Local: activation and learning equations employ only simple computations, using only locally available information. This constraint allows for the possibility of efficient hardware implementation.

2. On-line: weights are updated after presentation of each input pattern. This allows the network to learn in real-time.

3. Constructive: the network automatically constructs, or self-organizes, a representation of sufficient complexity to learn an input–output mapping. This allows for an efficient usage of resources since only the memory required for solving the problem at hand is allocated.

In this paper a neural network is proposed which obeys these three constraints. The network learns smooth receptive fields which are sensitive to input statistics by taking advantage of a data format suggested by models of cortical development—namely, representing inputs as topographic maps. The network's learning dynamics are also self-corrective in the presence of prediction errors. This is done, not by using complicated feedback computations based on an error gradient, but rather by using a simple biasing of network activities (and hence, of learning rates), based on the sharpness of receptive field tuning in the hidden layer. These two novel learning
mechanisms—learning a mixture density model based on topographic input maps, and biasing activities of the mixture components using “attentional” feedback—are described in sections 1.1 and 1.2 respectively.

1.1 Learning topographic mixtures

Models of cortical learning and development typically share two essential computations: (i) center-surround processing within cortical layers, and (ii) correlational learning of connections between layers. These computations yield characteristic activity patterns within layers in which cells positioned near to each other are more positively correlated than cells positioned further away from each other. Therefore, while all cells in the same layer may respond to the same feature dimension, such as the orientation of a visual stimulus, nearby cells tend to respond to nearby regions of the feature space (i.e., similar orientations) whereas cells spaced further apart do not. For example, many models have been proposed of how cells develop their orientationally tuned receptive fields within smoothly varying maps of orientation preference (Durbin and Mitchison, 1990; Obermayer et al., 1990, 1992; Swindale, 1992; Sirosh and Miikkulainen, 1994; Olson and Grossberg, 1998).

Figure 1 depicts a simple example of a topographic map in a feature layer, which encodes orientation preference for visual input. Activation levels of the orientationally selective nodes are shown, given a vertical edge as input. The feature layer is analogous to a set of orientation columns in primary visual cortex. Now imagine a set of nodes in a higher category layer, which have normalized activities due to mutual, divisive inhibition (i.e., the net activation across the layer constant), and which receives inputs from the feature layer via adaptive connections that are updated with a type of correlational learning called instar learning (Grossberg, 1976, 1980; Kohonen, 1989). Instar learning causes the connection weights to track the presynaptic signals from the feature nodes when the category nodes are active. The result, depicted in Figure 1, is that the weights to a category node consist of a weighted average of several similar feature patterns, encoding the expected feature pattern conditioned on the category node’s activity. The shape of the resulting category receptive field is determined by both the intrinsic spread of a single feature pattern, and the spread of different feature patterns learned by the category node.

The category nodes also receive signals from an output layer, which in this case represents the class labels “vertical” and “horizontal”. Connections from these nodes are also updated via instar learning. Signals from the feature and output nodes are combined multiplicatively at the category nodes, so that the category nodes learn conjunctions of the feature and class representations. By updating reciprocal category-output connections, the network learns to make correct predictions of the class labels given feature input only. Thus, we see that a combination of center/surround processing within the feature layer and correlational learning between layers results in the learning of a mixture model of smooth distributions governing the input-output mapping. In this model, each category node represents a single component of the mixture distribution. Because this circuit relies on topography in the feature layer, the representation that it learns is called a topographic mixture model. A probabilistic interpretation of this learning model is given in section 2.3.
Figure 1: Topographic mixture model. The learned paths are shown from a topographic feature layer and from an output layer to an internal category layer, as well as reciprocal paths from the category layer to the output layer. In this example, the feature nodes encode orientation preference. The category nodes learn smooth receptive fields which partition this feature space based on its associations with the class outputs. The dashed line represents divisive, normalizing inhibition within the category layer. The shaded bars represent activity levels.
1.2 Attentionally biased learning

Mixture models allow supervised learning within an unsupervised framework (Ghahramani and Jordan, 1994; Williamson, 1997). The essential idea is to optimize the model’s likelihood for representing the joint input/output density of the training data, and then to use this mixture model to estimate the conditional distribution in the output space given test inputs. This distribution is then used to obtain output predictions. An advantage of the density estimation approach is flexibility. The density model can be used to obtain predictions in any direction. In other words, the “output” variables merely correspond to the variables that are missing and need to be estimated given the variables that are not missing. Another advantage is locality. The posterior probabilities of the mixture components (i.e., category activations) can be computed using local information. These probabilities are then used to update local model parameters. In a neural network architecture, this yields simpler, more localizable computations than are typically used in gradient-descent based learning rules such as backpropagation.

However, one possible problem with this approach is that, due to inherent limitations in the ability to estimate an accurate model of the input/output density (due to imperfect assumptions underlying the parametric model), maximizing the likelihood of the mixture model may not correspond to maximizing the accuracy of predictions on a test set.

In fact, Williamson (1997) showed that an on-line classification network which learns a Gaussian mixture model of the input space in its category layer outperformed both an on-line and a batch-learning version of the classifier by using attentional feedback to bias the learning rates. That on-line classification network is a variant of predictive adaptive resonance theory (ARTMAP) networks called Gaussian ARTMAP, or GAM. When GAM makes an incorrect prediction during training, its receptive fields are biased by the raising of a “vigilance” threshold, the effect of which varies inversely with the width of each receptive field. This thresholding bias has two separate effects on relative activations among the categories, and hence, on learning rates. First, it differentially favors (in terms of relative activity levels) less selective generalist nodes over more selective specialist nodes. Second, due to the “iceberg effect” it narrows receptive fields, thereby making the activity pattern across categories less distributed by shutting some nodes off. For example, Figure 2a shows two receptive fields with different widths when a vigilance threshold is low. Raising the vigilance threshold favors the wider receptive field (the cross-over point moves to the left) while making both receptive fields narrower.

In this paper, a novel modulatory form of attentional biasing is introduced. Raising this modulatory form of vigilance (Figure 2b) favors the wider receptive field but does not make either receptive field narrower. A modulatory bias is more subtle and flexible than a thresholding bias, avoiding the danger of a network becoming completely silenced when the threshold is raised too high. The motivation for introducing this modulatory bias came from recent neurophysiological experiments which suggest that attention plays a modulatory role, and that neurons with broader tuning are proportionally boosted more by attentional modulation than neurons with sharper tuning (McAdams and Maunsell, 1999).

Historically, the use of a vigilance threshold in GAM was motivated by the ART concept of resetting, or shutting off, nodes when in some sense they do not adequately match the input
Figure 2: Effect of raising vigilance on two receptive fields. In this example, it is assumed that the bias weights for the two category nodes are proportional to their receptive field heights. This relationship typically holds, as illustrated in section 3.3. Two alternative methods for attentional biasing of receptive fields are illustrated. In the vigilance threshold case (top), receptive fields are defined by $x_{ji} = \sum_{k=1}^{L} f_{ik} w_{jik} - \rho^2 b_{ji}$. In the vigilance modulation case (bottom), they are defined by $x_{ji} = \frac{\sum_{k=1}^{L} f_{ik} w_{jik}}{1 + \rho^2 b_{ji}}$. See section 2 for an explanation of these variables. For clarity, receptive fields are rescaled in this plot to have the same height at different vigilance levels. A vigilance threshold (top) favors the wider receptive field and makes both receptive fields narrower. Vigilance modulation (bottom), on the other hand, favors the wider receptive field but does not make either receptive field narrower.
or output pattern. The vigilance modulation introduced in this paper is a modification of that concept. Simulations (not shown here) have shown that vigilance modulation is generally more effective than a vigilance threshold. The effectiveness of vigilance modulation suggests that, while it is useful to differentially bias nodes based on the sharpness of their tuning, it is generally not useful to carry this process to such an extreme that learning is completely turned off in some nodes.

### 1.3 Relationship to ARTMAP models

ARTMAP networks are designed to obey the three biologically-motivated constraints of being local, on-line, and constructive (Carpenter et al., 1991). The network proposed in this paper is the first ARTMAP variant to combine these attributes with the additional ones of using distributed learning and of learning smooth, statistically sensitive receptive fields. The two primary ARTMAP alternatives are fuzzy ARTMAP (FAM) and Gaussian ARTMAP (GAM) (Carpenter et al., 1992; Williamson, 1996, 1997). FAM is local, but does not learn smooth receptive fields and is not distributed (although a distributed variant has been developed; see Carpenter, 1997). GAM does learn smooth receptive fields with distributed learning, but is not local, in that its explicit Gaussian-defined receptive fields do not admit simple, biologically plausible activation and learning rules. The network proposed in this paper is called the Topographic Attentive Mapping, or TAM, network. This is because its novel contributions are the integral use of *topography* in the input feature maps, along with the use of *attentional* feedback to bias learning rates in the category layer, in order to learn an effective input—output mapping.

### 2 Equations

The TAM model variables are indexed as follows. The \( L \) feature nodes encoding a single input dimension are indexed by \( k \), the \( M \) input dimensions are indexed by \( i \), the \( N \) category nodes are indexed by \( j \), and the \( O \) output nodes are indexed by \( k \). Figure 3 illustrates a TAM network containing 2 input feature maps, 2 basis nodes, 1 category node, and 1 output node. Note that the category layer in the simple network architecture depicted in Figure 1 has been expanded to include both unidimensional basis nodes encoding a receptive field in each input dimension, and multidimensional category nodes encoding the conjunction of all the input dimensions and the output dimension. A vigilance node has also been added, which exerts divisive inhibition on the basis nodes.

### 2.1 Feedforward activations

Each category node receives input from \( M \) basis nodes, each encoding the match in a single input dimension. For the \( j^{th} \) category node, the match in the \( i^{th} \) basis node is computed by the inner product between the activity distribution in the \( i^{th} \) feature map, \( f_i \), and the distribution of the
Figure 3: TAM network, shown with two feature dimensions, one category node, and one output node. Dashed lines indicate inhibitory connections. Symbols indicate mathematical operation computed at each node.
node's weight field, \(w_{ji}\):

\[
x_{ji} = \frac{\sum_{k=1}^{L} f_{ik} w_{jih}}{1 + \rho^2 b_{ji}}.
\]

The numerator in equation (1) yields a smooth receptive field as illustrated in Figure 2. Changing the input value corresponds to shifting the activity pattern in the feature map, \(f_i\), which results in a change in the match between \(fi\) and \(w_{ji}\). The denominator in equation (1) describes the effect of attentional biasing on the node's activity. The vigilance term \(\rho\) is normally set to zero. Raising \(\rho\) produces divisive inhibition, with the amount of inhibition modulated by the inhibitory bias weight, \(b_{ji}\). The size of \(b_{ji}\) is positively correlated with the height (and negatively correlated with the width) of the receptive fields. This correlation is quantified on several benchmark simulations in section 3.3. Due to this correlation, tall narrow receptive fields are usually attenuated more than short wide receptive fields when \(\rho > 0\). Figure 2b illustrates the differential effects that raising vigilance has on two receptive fields with different widths.

Category nodes represent feature conjunctions across \(M\) perceptual dimensions. They are activated by a conjunction of bottom-up input from their \(M\) basis nodes:

\[
y_j = \prod_{i=1}^{M} x_{ji}.
\]

The network's output nodes (indexed by \(k\)) are then activated by the category nodes via weighted connections \(p_{jk}\) which represent the probability of output \(k\) given category \(j\):

\[
z_k = \sum_{j=1}^{N} y_j p_{jk}.
\]

The class prediction, \(K\), is the index of the maximally activated output node:

\[
K = \arg \max_k (z_k).
\]

### 2.2 Supervision and attention

Let \(K^*\) denote the index of the "correct" supervised output class. An output criterion (OC) determines whether the network's output prediction is similar enough to the supervised output to allow learning. If the OC is not met, attention is invoked: the vigilance level, \(\rho\), is incrementally raised from an initial value of \(\rho = 0\). This causes the predictions to change due to differential modulations, via equation (1), of activities in the basis nodes. Vigilance is raised until either the OC is satisfied or until the maximal vigilance level is reached:

\[
\text{If } z_K/z_{K^*} < \text{OC then repeat:}
\]

\[
(i) \quad \rho := \rho + \rho^{(\text{step})};
(ii) \quad \text{equations (1) - (4)};
\]

\[
\text{until either } z_K/z_{K^*} \geq \text{OC or } \rho \geq \rho^{(\text{max})}.
\]
Once equation (5) is satisfied, top-down feedback incorporates information as to the correct output. First, supervised feedback selects the correct output node:

\[ z^*_k = \delta[k - K^*], \tag{6} \]

where \( \delta[x] = 1 \) if \( x = 0 \); \( \delta[x] = 0 \) otherwise. Then, top-down output-category feedback factors in the conditional probabilities of the correct output (via the term \( \sum_{k=1}^{O} \hat{z}^*_k p_{jk} \)). In addition, normalization of the category activations causes each input to have the same net impact during learning, since the activations determine learning rates. The activations represent the category posterior probabilities given both the input and the correct output:

\[ y^*_j = \frac{\prod_{i=1}^{M} x_{ji} \cdot \sum_{k=1}^{O} \hat{z}^*_k p_{jk}}{\sum_{j'=1}^{N} \prod_{i=1}^{M} x_{j'j} \cdot \sum_{k=1}^{O} \hat{z}^*_k p_{j'k}}. \tag{7} \]

The output-category feedback is a feature-specific form of attention which serves a different role than the non-specific vigilance form of attention in equation (5). Feature-specific attention favors those categories that have strong associations with the correct external expectations. Vigilance-based attention, on the other hand, performs a memory search by favoring categories that have low bias weights, and hence wide receptive fields, regardless of their output associations.

### 2.3 Probabilistic interpretation of network activations

A probabilistic interpretation of the output activities, \( z^*_k \), and their influence on category activations in equation (7), is as follows. Supervised class labels are discrete, categorical data that can be modeled with a multinomial distribution. First, assume that the weights, \( p_{jk} \), encode this distribution:

\[ p_{jk} = P(k = K^*|j). \tag{8} \]

Then, the posterior probability of the \( j^{th} \) mixture component (i.e., activation of the \( j^{th} \) category) based solely on knowledge of the correct class, \( K^* \), is:

\[ P(j|K^*) = \frac{P(k = K^*|j)P(j)}{\sum_{j'=1}^{N} P(k = K^*|j')P(j')} = \frac{\sum_{k=1}^{O} \delta[k - K^*]p_{jk}}{\sum_{j'=1}^{N} \sum_{k=1}^{O} \delta[k - K^*]p_{j'k}} = \frac{\sum_{k=1}^{O} \hat{z}^*_k p_{jk}}{\sum_{j'=1}^{N} \sum_{k=1}^{O} \hat{z}^*_k p_{j'k}}. \tag{9} \]

Equation (9) shows that the addition of top-down feedback in equation (7) incorporates the conditional probabilities involving the output dimension. Note, however, that equation (9) assumes that the category prior probabilities, \( P(j) \), are uniformly distributed.

It has been found empirically that using learned estimates of category prior probabilities, \( \hat{P}(j) \), and incorporating these estimates into equation (7), degrades the network's accuracy as a classifier. A possible reason for this finding is as follows. To assume \( \hat{P}(j) = \hat{P}(j') \) for all \( j, j' \), as in equation (7), becomes a self-fulfilling prophecy. It causes the categories to learn a more balanced, efficient partitioning of the input/output space. If, on the other hand, learned estimates \( \hat{P}(j) \) are factored into equation (7), then the learning process tends to yield \( \hat{P}(j) \gg \hat{P}(j') \) for some \( j, j' \). Simulation results (not shown here) suggest that this yields a less effective utilization of resources.
A probabilistic interpretation of the input activities, \( \{ f_i \}_{i=1}^{M} \), and their influence on category activations, is as follows. For simplicity, assume \( \rho = 0 \) (\( \rho > 0 \) results in a biasing of the probabilities analyzed here). The discrete output data are encoded by discrete, nonoverlapping distributions among the output nodes, as in equation (6). The real-valued input data, on the other hand, are encoded by overlapping distributions among an ordered set of nodes, \( f_i \). Due to their overlapping distributions, these feature nodes represent values along a scale. Similarly, just as \( P_j \) represents the conditional distribution of discrete values among \( O \) output nodes, \( w_j^* \) represents the conditional density of smooth, overlapping distributions among \( L \) feature nodes encoding the \( i^{th} \) input dimension.

Therefore, with \( \rho = 0 \), the posterior probability of the \( j^{th} \) mixture component, based solely on knowledge of the feature input, is

\[
P(j|\{f_i\}_{i=1}^{M}) = \frac{P(\{f_i\}_{i=1}^{M}|j)P(j)}{\sum_{j'=1}^{N} P(\{f_i\}_{i=1}^{M}|j')P(j')} = \frac{\prod_{i=1}^{M} \sum_{j=1}^{L} f_{ih} w_{jih}}{\sum_{j'=1}^{N} \sum_{j=1}^{L} f_{ih} w_{jih}} = \frac{\prod_{i=1}^{M} x_{ji}}{\sum_{j'=1}^{N} \prod_{i=1}^{M} x_{ji}}. \tag{10}
\]

Equation (10) describes the posterior probability of component \( j \) based on a joint density model of the \( M \) input dimensions. The density distribution in each input dimension \( i \) is encoded by the weight vector \( w_{j}^* \), just as the distribution in the output dimension is encoded by \( P_j \). As in equation (9), the prior probabilities of the categories are assumed to be equal. It is easy to see from equations (9) and (10) that the activations \( y_{j}^* \) in equation (7) represent \( P(j|\{f_i\}_{i=1}^{M}, k = K^*) \).

### 2.4 Learning

Since category activations represent posterior probabilities conditioned on the current input/output, correlational learning rules allow the network to learn a mixture model of the input/output density. The learning procedure is, in effect, an on-line approximation of a statistical batch learning approach for optimizing mixture models, the expectation-maximization (EM) algorithm. See Williamson (1997) for a detailed explanation of this relationship.

On-line learning obtains better statistical sampling if the learning rate begins high and then reduces with experience. Experience is represented by \( n_{j} \), which begins at zero when category \( j \) is first instantiated and converges toward 1 via a discrete difference equation in which the change in \( n_{j} \) is defined as

\[
\Delta n_{j} = \alpha y_{j}^*(1 - n_{j}). \tag{11}
\]

The learning rate for feature weights \( w_{jih} \) begins with a relatively large value but converges toward a small fixed value as \( n_{j} \rightarrow 1 \):

\[
w_{j}^{(\text{rate})} = \frac{\alpha}{\alpha\beta(M) + n_{j}}. \tag{12}
\]

The number of input dimensions, \( M \), has an effect on the net change in receptive fields during learning. Matches computed in equation (1) are multiplied together in equation (2) to produce an \( M \)-dimensional receptive field. Therefore, if the value of \( w_{j}^{(\text{rate})} \) were independent of \( M \), increasing \( M \) would result in greater total changes in the receptive fields during learning. In order to avoid this effect, and instead obtain the same effective learning rate regardless of the dimensionality...
of a data set, the function $\beta(M)$ was added to equation (12) and defined to make the effective “multidimensional learning rate” invariant of $M$:

$$\beta(M) = \frac{\lambda^{1/M}}{1 - \lambda^{1/M}},$$  \hspace{1cm} (13)$$

with the rate of learning inversely proportional to the free parameter $\lambda \in (0 : 1)$. The derivation of equation (13) is given in Appendix 1. With $\lambda = 1/3$, which was used in all the simulations in this paper, $\beta(M) \approx 0.91M - 0.45$.

The feature weight $w_{jih}$ tracks its input, $f_{ih}$, at a rate proportional to $y_j^*$. Over time, weight vector $w_{ji}$ learns the average of the spatially varying activity distributions that input to it, and thereby ends up encoding a wider distribution than exists in any single $f_i$ activity distribution:

$$\Delta w_{jih} = w_{j}^{(rate)} y_{j}^*(f_{ih} - w_{jih}).$$  \hspace{1cm} (14)$$
The learning rate for class weights $p_{jk}$ also begins with a relatively large value and converges toward a small fixed value as $n_j \rightarrow 1$:

$$p_j^{(rate)} = \frac{a}{\alpha + n_j}.$$  \hspace{1cm} (15)$$

The output weights $p_{jk}$ track the output activities, and thereby learn the conditional probability that output $k$ is correct given category $j$:

$$\Delta p_{jk} = p_{j}^{(rate)} y_{j}^*(z_k^* - p_{jk}).$$  \hspace{1cm} (16)$$
The inhibitory bias weights $b_{ji}$ track the activations of their basis nodes at a constant rate:

$$\Delta b_{ji} = b_{j}^{(rate)} y_{j}^*(x_{ji} - b_{ji}).$$  \hspace{1cm} (17)$$

Equation (17) causes $b_{ji}$ to learn the expected value of $x_{ji}$, category $j$'s match in the $i^{th}$ input dimension. Categories with narrow receptive fields will tend to have larger matches, and thus larger bias weights. This correlation is quantified on benchmark simulations in section 3.3. Figure 2b illustrates the differential effect that these weights have on narrow versus wide receptive fields. Categories that learn when $\rho$ is large will also tend to have small bias weights, due to the effect of $\rho$ on $x_{ji}$ in equation (1). This dynamic is illustrated in section 3.1.

2.5 Category instantiation

Various heuristics are possible for determining when to instantiate new categories. In our simulations the following procedure is used: training always begins with zero categories ($N = 0$) and a new category is instantiated ($N := N + 1$) every time vigilance reaches its maximal level, $\rho = \rho^{(max)}$. The reasoning behind this rule is that, by the time $\rho$ reaches $\rho^{(max)}$, the OC has not been satisfied for $0 \leq \rho \leq \rho^{(max)}$. This suggests that the existing receptive fields are poorly positioned to learn the current input-output mapping, and therefore a new receptive field is needed,
centered on the current input. As a result of this rule, the number of categories that is created depends on the difficulty of the classification task. New categories are initialized in a “tabula rasa” state, with \( n_j = b_j = 0 \) and with uniformly distributed weights, \( w_{ijk} = 1/L \) and \( p_{jk} = 1/O \). Following its instantiation, a new category’s activity is computed via equations (1)–(3). Learning then takes place for all categories via equations (6)–(7), (11)–(17).

2.6 Input preprocessing

Self-organizing feature maps (SOFMs) produce a data format with two properties: (i) nearby cells are correlated, i.e., they have overlapping receptive fields, and (ii) the distribution of receptive fields in the input space is sensitive to the input density. Figure 4 illustrates the ideal effect of a 1-D SOFM on 1-D data. The uneven data density (top) is uniformized across the space of nodes in the map (bottom).

Simulating SOFMs is beyond the scope of this paper. Rather, an approximation of their properties is obtained as follows. First, histogram equalization is performed on each dimension of the training data set, with the number of bins equal to the number of data items. If several data items have the same value, the median bin index corresponding to these items is used for all of them. The bin indices are then mapped into the range \([1/(2T) : 1 - 1/(2T)]\), where \( T \) is the number of items in the training set. This yields input values \( I_i \) in a new, warped feature space containing a uniform distribution over the training data, as shown in Figure 4. Fixed-width activity distributions are then obtained via the equation

\[
\frac{\exp[-0.5(LL_i - h + 0.5)^2]}{\sum_{h'=1}^{L} \exp[-0.5(LL_i - h' + 0.5)^2]},
\]

where \( L \) is the number of nodes in the 1-D topographic map. \( L \) determines the level of resolution of the map, and the width of the Gaussian activity distributions with respect to the input range of \([0 : 1]\) is \( \sigma = 1/L \). These fixed-width activity distributions correspond to variable-width distributions in the original input space. Therefore, they effect a variable bandwidth smoothing of the original input space. The width of an activity distribution corresponds to the narrowest possible width that a weight distribution can have, due to equation (12). Therefore, the size of \( L \) determines a minimum level of regularization for the network, with smaller \( L \) resulting in greater regularization. This can have the positive effect of preventing overlearning but the negative effect of limiting capacity and hence the ability to discriminate.

The \( I_i \) values for test data are obtained by linear interpolation between the two nearest training values. If a test value is smaller or larger than any training values, then its \( I_i \) value is obtained by linear interpolation between either the smallest training item and zero, or between the largest training item and one.
Figure 4: Warping of input space, as described in section 2.6, which approximates the effect of SOFM's. Top: Uneven density distribution of training data in the original input space is shown (dashed line), along with training items selected from this distribution (x's). Bottom: Uniform density distribution of data in the warped feature space is shown, as well as the evenly spaced training items. Fixed-width activity distributions in this warped space, as defined by equation (18), correspond to variable-width distributions in the original input space.
2.7 Parameters

The following set of parameters was used on all the simulations: \( OC = 0.8, \rho^{(step)} = 0.1, \rho^{(max)} = 100, \alpha = 10^{-7}, \lambda = 1/3, b_j^{(rate)} = 0.01. \) The simulation experiments in section 3.1 involve a high resolution 1-D representation, so a large value of \( L (L = 24) \) was used. The multidimensional classification benchmarks in section 3.2 do not presumably benefit from such high resolution in each dimension, so a smaller value of \( L (L = 12) \) was used.

3 Simulations

3.1 Effect on learning of raising vigilance

Figure 2 illustrates the momentary effect on receptive fields of raising vigilance. But what is the long-term effect of raising vigilance on learning of receptive fields? To investigate this question the following simulation experiments were performed. For simplicity, only one input dimension and one output class were used. Learning with just one output class is functionally equivalent to unsupervised learning. The network was initialized with 10 category nodes. Each node’s receptive field was centered on a different part of the 1-D input space by allowing it to learn, independently of the other nodes, for one learning trial. The value of \( I_i \) for the first learning trial of each category \( j \) was \( I_i = (j - 1)/10. \) Therefore, the 10 receptive fields were centered at 0.0, 0.1, ..., 0.9 in the input space, which has a range of \([0:1] \). In addition, equation (18) was altered to allow wraparound so that receptive fields at the boundaries of the input range would look the same as those in the middle.

After this category initialization, learning took place in the normal way, using 1,000 randomly selected training inputs, \( I_i \in U[0 : 1] \). Figure 5a shows the receptive fields resulting from Condition 1, which is the default condition with \( \rho = 0. \) Figure 5b shows the result of Condition 2, in which \( \rho \) was elevated in one region of the input space: \( \rho = 50 \) whenever \( 1/3 \leq I_i \leq 2/3 \) and \( \rho = 0 \) otherwise. Where \( \rho \) was elevated, the representation became less distributed, with the receptive fields in that area becoming taller and narrower (all receptive fields are plotted using equation (7) with \( \rho = 0 \)).

When a category node learns with \( \rho \) elevated, its bias weight \( b_{ji} \) tracks a smaller match value due to equation (17), giving the node an even greater advantage the next time \( \rho \) is elevated. This leads to a snowballing effect in which the advantaged nodes crowd out the disadvantaged nodes where \( \rho \) is elevated.

A possible alternative approach for “paying greater attention” to a particular region of the input space is to monolithically increase the learning rates in that region. Figure 5c shows the result of Condition 3, in which all learning rates, \( w_j^{(rate)} \), were doubled when \( 1/3 \leq I_i \leq 2/3. \) As this plot shows, the opposite effect was obtained as in Condition 2. In Condition 3, the representation became more distributed in the region of the input space with the elevated learning rate.
Figure 5: Effects on learning of raising vigilance. Plotted are receptive fields learned after: a) training with $\rho = 0$; b) training with $\rho = 50$ if $1/3 \leq I_i \leq 2/3$ and $\rho = 0$ otherwise; c) training with $\rho = 0$ but with learning rate $w_{j}^{(roc)}$ doubled when $1/3 \leq I_i \leq 2/3$. d) A discrimination index ($DI$) is plotted for each of the representations in a–c. The $DI$ is the sum of the magnitudes of the receptive field slopes.
One measure of the discriminability of a point in the input space is a discriminability index, $DI$, which is the average of the magnitudes of the receptive field slopes at that point. Figure 5d plots the $DI$ for the three representations shown in Figure 5a-c. Condition 1 produces a roughly flat $DI$ across the input space, reflecting the uniform distribution of the inputs. Condition 2 produces a flat $DI$ everywhere except for the transitions from $p = 0$ to $p = 50$, where the $DI$ is elevated. Finally, Condition 3 produces a slightly lowered $DI$ within the region where the learning rate was doubled. Therefore, attentional biasing increases discriminability in regions where there are changes in the average vigilance level. This is done by, instead of increasing the learning rate of all nodes, as in Condition 3, increasing it just for those nodes with small bias weights while simultaneously decreasing their bias weights.

### 3.2 DELVE classification benchmarks

The TAM network was evaluated on several classification benchmarks in order to determine how well it performs with respect to other classifiers. All results were obtained using the same set of parameters (see section 2.7) following 200 training epochs, or iterations through the training set.

The DELVE benchmark collection (Rasmussen et al., 1996) provides an environment for assessing learning methods in a way that is both relevant to real-world problems, and that allows for statistically valid comparisons with other learning methods. TAM was evaluated on the seven classification benchmarks in the DELVE collection for which there are results from one or more alternative learning methods. From each classification dataset one or more classification task is defined, involving training sets with different amounts of data. Therefore, it is possible to determine not only how well a learning method handles a particular classification problem, but also how its performance scales with the size of the training set.

The datasets are of four different types: natural, cultivated, simulated, and artificial. Natural datasets were originally gathered for real-world applications; cultivated datasets came from a real-world source, but were never used to solve a real problem; simulated datasets were generated by a simulator, but are believed to resemble real data; artificial datasets were generated according to some mathematical formula and are not meant to resemble any real data.

TAM was compared to the following alternative learning methods. CART: a basic decision tree which creates decision boundaries parallel to the input axes (Breiman, et al., 1984). 1NN: one nearest neighbor based on Euclidean distance. KNN-Class: $k$-nearest neighbor algorithm, in which $k$ is chosen on the basis of leave-one-out cross-validation on the training set. Several mixtures-of-experts (ME’s) and hierarchical mixtures-of-experts (HME’s) variants, as follows (Waterhouse et al., 1996). ME-EL: a committee of ME’s trained by ensemble learning; ME-ESE: a committee of ME’s trained by early stopping; HME-EL: a committee of HME’s trained by ensemble learning; HME-ESE: a committee of HME’s trained by early stopping; HME-GROW: a committee of HME’s grown via early stopping.

Tables 1–6 summarize TAM’s performance, and provide statistical comparisons with the results of the alternative learning methods. Each column shows results obtained in a single classification task, with the number at the top of the column indicating the number of data items in the training set. In each row is listed the error rate for a given learning method. Next to the error rate is
Table 1:

<table>
<thead>
<tr>
<th>Classifier</th>
<th>100</th>
<th></th>
<th>200</th>
<th></th>
<th>400</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TAM</td>
<td>.125</td>
<td></td>
<td>.081</td>
<td></td>
<td>.062</td>
<td></td>
</tr>
<tr>
<td>TAM-C</td>
<td>.161</td>
<td>+</td>
<td>.090</td>
<td>+</td>
<td>.056</td>
<td></td>
</tr>
<tr>
<td>CART</td>
<td>.215</td>
<td>+</td>
<td>.140</td>
<td>+</td>
<td>.100</td>
<td>+</td>
</tr>
<tr>
<td>1NN</td>
<td>.459</td>
<td>+</td>
<td>.429</td>
<td>+</td>
<td>.393</td>
<td>+</td>
</tr>
<tr>
<td>KNN-CLASS</td>
<td>.339</td>
<td>+</td>
<td>.293</td>
<td>+</td>
<td>.263</td>
<td>+</td>
</tr>
<tr>
<td>ME-EL</td>
<td>.145</td>
<td>+</td>
<td>.100</td>
<td>+</td>
<td>.076</td>
<td></td>
</tr>
<tr>
<td>ME-ESE</td>
<td>.152</td>
<td>+</td>
<td>.097</td>
<td>+</td>
<td>.068</td>
<td></td>
</tr>
<tr>
<td>HME-EL</td>
<td>.144</td>
<td>+</td>
<td>.099</td>
<td>+</td>
<td>.077</td>
<td></td>
</tr>
<tr>
<td>HME-ESE</td>
<td>.154</td>
<td>+</td>
<td>.099</td>
<td>+</td>
<td>.070</td>
<td></td>
</tr>
<tr>
<td>HME-GROW</td>
<td>.153</td>
<td>+</td>
<td>.106</td>
<td>+</td>
<td>.072</td>
<td></td>
</tr>
</tbody>
</table>

an indication of whether the difference between that method’s performance and that of TAM is statistically significant ($p < 0.05$), with a “+” indicating that the alternative method obtained a significantly higher error rate, and a “−” indicating that it obtained a significantly lower error rate.

**Splice.** This natural problem involves, given a position in the middle of a window of 60 DNA sequence elements, predicting if the position is an “intron → exon” boundary, an “exon → intron” boundary, or neither type of boundary. Therefore, a classifier must learn to predict one of three class outputs based on a 60-dimensional input vector, in which each dimension takes on a categorical value, $A$, $G$, $T$, or $C$. Preprocessing involved assigning numerical values to these categorical data: $A = 0$, $G = 1$, $T = 2$, $C = 3$, and then mapping these values into feature map distributions as described in section 2.6. Table 1 shows the results. On the small and medium training sets, TAM performed significantly better than most of the alternative methods. On the largest training set, TAM performed significantly better than CART and the NN methods, and slightly better than the ME and HME variants, although these latter differences are not statistically significant.

TAM’s superior performance may result partly from the preprocessing, which, due to overlapping feature distributions for different categorical attributes, converts categorical data into a numerical scale. As discussed in section 2.3, categorical data can be represented discretely by making the distributions nonoverlapping. Accordingly, TAM was also evaluated using narrow, nonoverlapping feature map distributions whose widths with respect to the input range of $[0 : 1]$ are $\sigma = 1/(10L)$ rather than $\sigma = 1/L$:

$$f_{ik} = \frac{\exp[-5(LL_i - h + 0.5)^2]}{\sum_{h' = 1}^{L} \exp[-5(LL_i - h' + 0.5)^2]}$$

(19)

When TAM uses “categorical” feature maps due to preprocessing with equation (19) rather than
equation (18), it is referred to as “TAM-C”. Table 1 compares TAM-C’s performance to that of TAM. On the small and medium training sets, TAM-C did not perform as well. Apparently, TAM’s overlapping input distributions provided an additional benefit, even though the input data have a categorical interpretation. On the largest dataset, however, TAM-C performed slightly better than TAM, perhaps because its nonoverlapping input distributions precluded “crosstalk”.

**Titanic.** This natural problem involves learning to predict whether or not a person on board the Titanic survived based on their social class (first class, second class, third class, crewmember), age (adult or child), and sex. Preprocessing involved assigning numerical values to these categorical data: (first class = 0, second class = 1, third class = 2, crewmember = 3), (adult = 0, child = 1), (male = 0, female = 1), and then converting these data into feature distributions as described in section 2.6. Because the input data are categorical, TAM-C was also evaluated. As Table 2 shows, TAM and TAM-C obtained similar results except that TAM did better on the smallest training set. TAM obtained significantly better results than 1NN on all training set sizes. It obtained slightly better results than the other methods as well, although these differences are not significant.

**Image-seg.** This cultivated problem involves, given 16 continuous attributes derived from $3 \times 3$ pixel regions from outdoor images, predicting which region class (brickface, sky, foliage, cement, window, path, grass) it came from. Table 3 shows the results. Only a couple of the differences are statistically significant, however TAM performed slightly worse than the ME and HME variants on the medium and large training sets.

**Letter.** This simulated problem involves predicting which of the 26 upper case letters an image came from, given 16 simple statistical features derived from the image, with each feature taking on one of 16 integer values. Table 4 shows the results. TAM performed significantly better than CART, and significantly worse than virtually all the other methods, on all three training set sizes.
Table 3:

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Image-seg 70 error</th>
<th>Image-seg 140 error</th>
<th>Image-seg 280 error</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAM</td>
<td>.319</td>
<td>.341</td>
<td>.311</td>
</tr>
<tr>
<td>CART</td>
<td>.361</td>
<td>.348</td>
<td>.288</td>
</tr>
<tr>
<td>1NN</td>
<td>.338</td>
<td>.331</td>
<td>.322</td>
</tr>
<tr>
<td>KNN-CLASS</td>
<td>.353</td>
<td>.333</td>
<td>.322</td>
</tr>
<tr>
<td>ME-EL</td>
<td>.309</td>
<td>.317</td>
<td>.286</td>
</tr>
<tr>
<td>ME-ESF</td>
<td>.320</td>
<td>.337</td>
<td>.282</td>
</tr>
<tr>
<td>HME-EL</td>
<td>.311</td>
<td>.306</td>
<td>.283</td>
</tr>
<tr>
<td>HME-ESF</td>
<td>.319</td>
<td>.313</td>
<td>.283</td>
</tr>
<tr>
<td>HME-GROW</td>
<td>.317</td>
<td>.321</td>
<td>.285</td>
</tr>
</tbody>
</table>

Table 4:

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Letter 390 error</th>
<th>Letter 780 error</th>
<th>Letter 1560 error</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAM</td>
<td>.396</td>
<td>.308</td>
<td>.226</td>
</tr>
<tr>
<td>CART</td>
<td>.501</td>
<td>+ .410</td>
<td>+ .327</td>
</tr>
<tr>
<td>1NN</td>
<td>.351</td>
<td>- .262</td>
<td>- .186</td>
</tr>
<tr>
<td>KNN-CLASS</td>
<td>.351</td>
<td>- .262</td>
<td>- .186</td>
</tr>
<tr>
<td>ME-EL</td>
<td>.322</td>
<td>- .262</td>
<td>- .195</td>
</tr>
<tr>
<td>ME-ESF</td>
<td>.302</td>
<td>- .252</td>
<td>- .185</td>
</tr>
<tr>
<td>HME-EL</td>
<td>.314</td>
<td>- .246</td>
<td>- .195</td>
</tr>
<tr>
<td>HME-ESF</td>
<td>.312</td>
<td>- .240</td>
<td>- .190</td>
</tr>
<tr>
<td>HME-GROW</td>
<td>.345</td>
<td>- .276</td>
<td>- .218</td>
</tr>
</tbody>
</table>
Mushrooms. This artificial problem involves, given 21 nominally valued attributes describing the characteristics of mushrooms, predicting whether the mushroom is edible or poisonous. Pre-processing involved assigning integer values to the attributes (which range in number from 2 to 12 per dimension) by replacing each attribute with the number of its index in the documentation lists. Then, these data were converted into feature distributions as described in section 2.6. Because the input data are categorical, TAM-C was also evaluated. Table 5 shows the results. As in the Splice dataset, TAM performed slightly better than TAM-C on the smaller training sets, and slightly worse on the larger training sets, although the differences are not significant. TAM performed slightly worse than KNN-CLASS on all the training sets except the largest one.

Ringnorm. This artificial problem involves, given two output classes drawn from different 20-dimensional multivariate normal distributions, predicting which class each datum belongs to. Class 1 has mean zero and covariance 4 times the identity. Class 2 has mean \( (a, a, \ldots, a) \) and unit covariance, where \( a = 2/\sqrt{20} \). Breiman et al. (1984) report a theoretical expected error rate of 0.013. Table 6 shows that TAM performed significantly better than CART and the NN methods.

Twonorm. This artificial problem is similar to the ringnorm problem except that both classes have unit variance, Class 1 has mean \( (a, a, \ldots, a) \), and Class 2 has mean \( (-a, -a, \ldots, -a) \) where \( a = 2/\sqrt{20} \). Breiman et al. (1984) report a theoretical expected error rate of 0.023. Table 6 shows that TAM performed significantly better than CART and 1NN but worse than KNN-CLASS.
3.3 Statistics of Learning

In this section, several statistics relating to the above benchmark simulations are described in order to gain a better understanding of the representations that TAM learned. For clarity, results are only shown from the largest training set of the first five benchmarks.

**Speed of convergence.** The results shown in Tables 1-6 were obtained following 200 training epochs. However, the error rate converged much earlier than 200 epochs on most datasets. Figure 6 (top) illustrates this by plotting the average error rates obtained after each training epoch. The error rates nearly converged following only 32 training epochs.

**Number of category nodes.** Another important question is the amount of memory that TAM requires. The memory requirements are the number of category nodes that are created times the number of weights per category. Figure 6 (bottom) illustrates the average number of categories per output class that were created. For all except the titanic dataset, TAM equilibrated with a relatively small number of categories per output class. On these datasets TAM also had low training error rates, ranging from 0.0 on the mushrooms dataset up to 0.03 on the image-seg dataset. However, on the titanic dataset the number of categories kept growing despite the fact that the training and test error rates barely changed following the first few epochs. The problem appears to be that the training error rate remained fixed at a relatively high level of 0.2. Therefore, with the current heuristics governing category instantiation (see section 2.5) there may be a tendency for category proliferation when training set error remains high. This is a topic for future research.

**Weight pruning.** The number of feature weights per category node is $LM$, where $L$ is the number of weights in each dimension and $M$ is the number of dimensions. Weight pruning can significantly reduce the number of weights per dimension, often with little or no effect on the error rate. When a category is first instantiated, it has uniformly distributed feature weights, $w_{jih} = 1/L$. With learning these weights typically converge into a unimodal, Gaussian-like distribution. When this happens, the smaller weights become insignificant, and can be removed. A similar dynamic occurs for the class weights, $p_{ji}$, although pruning of these weights is not explored here. Figure 7 illustrates the effect on error rates of pruning the smallest feature weights following 200 training epochs. After the smallest weights were set to zero, the remaining weights in each vector $w_{ji}$ were renormalized to sum to one. The error rates are plotted as a function of the average number of weights remaining for each 1-D basis node. For all except the splice dataset, error rates were virtually unaffected after 7 out of the 12 weights were pruned.

**Receptive fields and bias weights.** Earlier in the paper, it was claimed that match tracking favors categories with wide receptive fields over those with narrow receptive fields because the latter tend to have larger bias weights. A way to quantify this relationship is to compute the correlations between the "narrowness" of receptive fields and the size of their corresponding bias weights. Narrowness of receptive fields is approximated by a simple statistic, the size of
Figure 6: Top: average error rates are plotted as a function of the number of training epochs for five DELVE datasets. Bottom: average number of categories per output class.
Figure 7: Effect of pruning the smallest feature weights. Error rate is plotted as a function of the average number of remaining feature weights per input dimension.
the maximum feature weight, \( \max_h(w_{jih}) \). Correlations were computed across all nodes \( j \) and dimensions \( i \), between the maximum feature weights, \( \max_h(w_{jih}) \), and the bias weights, \( b_{ji} \). The average correlations for each dataset were as follows. Splice: 0.92; titanic: 0.90; image-seg: 0.97; letter: 0.93; mushrooms: 0.99. Figure 8 shows the scatter plots relating these two variables from a single simulation of the first four datasets. The majority of cases, which lie on the main diagonal, correspond to bias weights which were updated with, for the most part, \( \rho = 0 \). The cases that lie below the diagonal correspond to bias weights which were often updated with \( \rho > 0 \), as was illustrated in section 3.1. Note that, in particular, a large proportion of bias weights fall below the main diagonal on the titanic dataset, on which the training error rate was especially high.

### 3.4 Classifying natural textures

One of the motivations for developing the TAM model was to obtain a classifier that was computationally similar to the GAM model, but which, unlike GAM, obeyed the locality constraint. TAM's primary differences from GAM are that it (i) synthesizes Gaussian-like receptive fields with a set of feature weights, instead of explicitly defining them with means and variances; (ii) uses vigilance modulation instead of a vigilance threshold (see Figure 2); (iii) employs distributed connections, \( p_{ji} \), with graded strengths between the categories and the output nodes, rather than binary-valued connections.

In order to evaluate the effect of these changes, TAM was compared to GAM on natural texture classification benchmarks, on which GAM obtained good results (Grossberg and Williamson, 1998). The dataset was produced by a biologically-motivated image processing system which extracted, from each 8x8 pixel region of an 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 items), and tested on a fourth image (256 items). GAM was evaluated in this way on different numbers, \( 6, 12, 18, \ldots, 42 \), of 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 9 summarizes GAM's results, which were obtained after two training epochs. On the whole, GAM's performance did not improve with further training. As reported in Grossberg and Williamson (1998), 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.

Figure 9 also shows TAM's results on these datasets. Due to the large size of the training sets, TAM was trained for only 50 epochs for computational tractability. TAM obtained lower error rates than GAM when the number of textures was small and essentially equivalent error rates when the number of textures was larger (Figure 9, top). TAM required more training epochs to reach the same performance level as GAM, and also required more categories per class (Figure 9, bottom).
Figure 8: Scatter plots, obtained following 200 training epochs, relating two variables. These are $\max_h(w_{ijh})$ on the abscissa and $b_{ji}$ on the ordinate.
Figure 9: Top: error rates for TAM and GAM as a function of the number of textures in training/testing dataset. Bottom: numbers of categories created by TAM and GAM.
4 Discussion

4.1 Relations to biological neural networks

As discussed in section 1, TAM’s two novel learning mechanisms are motivated by data about biological neural networks. The first mechanism is a circuit for learning topographic mixtures, which is motivated by the apparent ubiquity of topographic maps in cortex. The second is an attentional biasing feedback circuit, which is motivated by findings that attention plays a modulatory role, and particularly that it proportionally boosts broadly tuned cells more than it does narrowly tuned cells (McAdams and Maunsell, 1999).

Another possible link with biology is with the finding that cortical development involves a period of synaptic proliferation followed by refinement and clustering of projections (Calloway and Katz, 1990; Kandel and O’Dell, 1992; Antonini and Stryker, 1993). If a pruning rule is used during training, so that weights are removed if they fall below a threshold (see section 3.3), then TAM also produces a pattern of synaptic proliferation followed by refinement. This is because the weight fields are instantiated with uniform distributions and pruning only occurs when the weight distributions become “peaked”, so that many weights become small and insignificant.

TAM may also serve as a useful model of perceptual learning. A previous version of the network was used to explain data about the learning of vernier discrimination (Williamson, 1999). Specifically, the model was used to explain (i) how various types of external feedback can affect the course of learning, based on the effect of that feedback on the OC (Herzog and Fahle, 1997); and (ii) why improvements in vernier discriminability during learning are reflected in sharper tuning in the orientational channels underlying discrimination, as revealed by visual masking experiments (Saarinen and Levi, 1995; Mussap and Levi, 1997).

4.2 Future research directions

Hierarchical network. In the current TAM network, there is no topography in the category layer. If topography were added to this layer—that is, if center-surround interactions were added so that the amount of overlap in the learned receptive fields was related to the distance between the category nodes in a map—then a hierarchical network structure could naturally emerge. A category layer at one level of the hierarchy would serve as one of the feature maps for the next level of the hierarchy. Meanwhile, each level of the hierarchy could have reciprocal connections to an output layer encoding the appropriate class labels for that level of the hierarchy.

Learned connections between basis nodes and category nodes. In the network diagram in Figure 3, there is one set of connections which does not contain adaptive weights: the connections between the unidimensional basis nodes and the multidimensional category nodes. Adding adaptive weights to these connections would allow for two interesting outcomes. The first is weight sharing, in which one basis node, and hence one 1-D weight field, could be shared by several category nodes. This would reduce memory requirements and possibly increase regularization. The second is subspace learning, in which a category node would retain connections from only a subset.
of basis nodes, thereby remaining invariant to one or more input dimensions. This could be a method for ignoring uninformative dimensions.

**Fast match tracking.** The primary technical disadvantage of the current TAM network is that the match tracking process in equation (5) can be very computationally intensive, due to its iterative nature. With the current set of parameters, equations (1)–(4) can be repeated as many as 1,000 times for a single input. The GAM network avoids these costly computations by employing a "one shot" match tracking algorithm in which the number of iterations can never exceed the number of output classes. A goal of future research is therefore to develop an approximation of equation (5) which greatly speeds up training without sacrificing classification accuracy.
References


Appendix 1

Consider the effect of $M$ on $\Delta w_{ji}$ in equation (14) if $w_j^{(rate)}$ is constant, i.e., if $w_j^{(rate)} = \gamma$. For simplicity, assume $y_i^j = 1$ and $\rho = 0$. Then, the change in the weight vector $w_{ji}$ is

$$\Delta w_{ji} = \gamma (f_i - w_{ji}).$$

(20)

What is the effect of $M$ on the net change in the multidimensional receptive field?

First, we decompose $w_{ji}$ into its previous value plus the most recent change: $w_{ji} = w_{ji}^{(old)} + \Delta w_{ji}^{(old)}$. Then, via equations (1) and (2), we can see that the changes in each dimension are multiplied across all $M$ dimensions:

$$y_i = \prod_{i=1}^{M} x_{ji} = \prod_{i=1}^{M} f_i^T \cdot \left( w_{ji}^{(old)} + \Delta w_{ji}^{(old)} \right)$$

$$= \prod_{i=1}^{M} f_i^T \cdot \left( w_{ji}^{(old)} + \gamma (f_i^{(old)} - w_{ji}^{(old)}) \right)$$

$$= \prod_{i=1}^{M} f_i^T \cdot \left( \gamma f_i^{(old)} + (1 - \gamma) w_{ji}^{(old)} \right).$$

(21)

From the right-hand term in equation (21) we can see that the fraction of the previous receptive field which is retained depends on $M$. In order to preserve the same amount of the previous receptive field regardless of the value of $M$, we would therefore like to have

$$(1 - \gamma)^M = \lambda,$$

(22)

where $\lambda \in (0 : 1)$ is a constant learning rate independent of $M$.

Obviously, equation (22) can only be true for arbitrary $M$ if we make $\gamma$ a function of $M$. This is why equation (12) contains $\beta(M)$, which is a function of $M$. Our goal here is to determine what form $\beta(M)$ should take. Different experience levels, $n_j$, result in different functions $\beta(M)$. Therefore, we need to assume a fixed $n_j$. Based on the idea that fixing learning rates with respect to $M$ is most important on a node's first learning trials, when the learning rate is highest, we fix $n_j$ to the value it takes after the first learning trial of the first instantiated node. This is $n_j = \alpha$, yielding

$$\gamma = w_j^{(rate)} = \frac{1}{\beta(M) + 1}.$$  

(23)

Plugging this into equation (22) yields

$$\left(1 - \frac{1}{\beta(M) + 1}\right)^M = \left(\frac{\beta(M)}{\beta(M) + 1}\right)^M = \lambda,$$

(24)

or

$$\beta(M) = \frac{\lambda^{1/M}}{1 - \lambda^{1/M}}.$$

(25)