
PERGAMON

Neural Networks 11 (1998) 793–813

Neural Networks

Invited article

Distributed ARTMAP: a neural network for fast distributed supervised learning[1]

Gail A. Carpenter*, Boriana L. Milenova, Benjamin W. Noeske

Center for Adaptive Systems and Department of Cognitive and Neural Systems, Boston University, Boston, MA 02215, USA

Received 19 December 1997; accepted 19 December 1997

Abstract

Distributed coding at the hidden layer of a multi-layer perceptron (MLP) endows the network with memory compression and noise tolerance capabilities. However, an MLP typically requires slow off-line learning to avoid catastrophic forgetting in an open input environment. An adaptive resonance theory (ART) model is designed to guarantee stable memories even with fast on-line learning. However, ART stability typically requires winner-take-all coding, which may cause category proliferation in a noisy input environment. Distributed ARTMAP (dARTMAP) seeks to combine the computational advantages of MLP and ART systems in a real-time neural network for supervised learning. An implementation algorithm here describes one class of dARTMAP networks. This system incorporates elements of the unsupervised dART model, as well as new features, including a content-addressable memory (CAM) rule for improved contrast control at the coding field. A dARTMAP system reduces to fuzzy ARTMAP when coding is winner-take-all. Simulations show that dARTMAP retains fuzzy ARTMAP accuracy while significantly improving memory compression.

Keywords: Distributed ARTMAP; Adaptive resonance; ART; ARTMAP; Distributed coding; Fast learning; Supervised learning; Neural network

1. Distributed coding by adaptive resonance systems

Adaptive resonance theory (ART) began with an analysis of human cognitive information processing (Grossberg, 1976, 1980). Fundamental computational design goals have therefore always included memory stability with fast or slow learning in an open and evolving input environment. As a real-time model of dynamic processes, an ART network is characterized by a system of ordinary differential equations, which are approximated by an algorithm for implementation purposes. In a general ART system, an input is presumed to generate a characteristic pattern of activation, or spatial code, that may be distributed across many nodes in a field representing a brain region such as the inferior temporal cortex (e.g., Miller et al., 1991).

While ART code representations may be distributed in theory, in practice nearly all ART networks feature winner-take-all (WTA) coding. These systems include ART 1 (Carpenter & Grossberg, 1987) and fuzzy ART (Carpenter et al., 1991b) for unsupervised learning, and ARTMAP (Carpenter et al., 1991a) and fuzzy ARTMAP (Carpenter et al., 1992) for supervised learning. The coding field of a supervised system is analogous to the hidden layer of a multi-layer perceptron (MLP) (Rosenblatt, 1958, 1962; Rumelhart et al., 1986; Werbos, 1974), where distributed activation helps the network achieve memory compression and generalization. However, an MLP employs slow learning, which limits adaptation for each input and so requires multiple presentations of the training set. With fast learning, where dynamic variables are allowed to converge to asymptote on each input presentation, MLP memories suffer catastrophic forgetting. However, features of a fast-learn system, such as its ability to encode significant rare cases and to learn quickly in the field, may be essential for a given application domain. Additional ART capabilities, including stable coding and scaling to accommodate large databases, are also essential for many applications, such as the Boeing parts design retrieval system (Caudell et al., 1994).

An overall aim of the distributed ART (dART) research program is to combine the computational advantages of ART and MLP systems. Desirable properties include code stability when learning is fast and on-line, memory

* Requests for reprints should be sent to Professor Gail A. Carpenter, Department of Cognitive and Neural Systems, 677 Beacon Street, Boston University, Boston, MA 02215, USA. Tel.: (617) 353-9483; fax: (617) 353-7755; e-mail: gail@cns.bu.edu

[1] Technical Report CAS/CNS TR-97-026, Boston, MA: Boston University.

0893–6080/98/$19.00
PII: S0893-6080(98)00019-7

compression when inputs are noisy and unconstrained, and real-time system dynamics.

1.1. Distributed learning

A key step in the derivation of the first family of dART models (Carpenter, 1996, 1997) was the specification of dynamic learning laws for stable distributed coding. These laws generalize the instar (Grossberg, 1972) and outstar (Grossberg, 1968, 1970) laws used, for example, in fuzzy ART. Instar and outstar learning feature a gating operation that permits weight change only when a coding node is active. This property is critical to ART stability. With a distributed code and fast learning, however, instar and outstar dynamics cause catastrophic forgetting. A system such as Gaussian ARTMAP (Williamson, 1996) includes many features of a distributed coding network, but retains the instar and outstar learning laws of earlier ART and ARTMAP models. The weight update rules in a Gaussian ARTMAP algorithm therefore approximate a real-time system only in the slow-learn limit. Other ARTMAP variations, such as ART-EMAP (Carpenter & Ross, 1995) and ARTMAP-IC (Carpenter & Markuzon, 1998), acquire some of the advantages of distributed coding, but sidestep the learning problem by permitting distributed activation during testing only.

The distributed instar (Carpenter, 1997) and distributed outstar (Carpenter, 1994) laws used in dART dynamically apportion learned changes according to the degree of activation of each coding node, with fast as well as slow learning. The update rules listed in the dARTMAP implementation algorithm represent exact, closed form solutions of the model differential equations. These solutions are valid across all time scales, with fast or slow learning. When coding is WTA, the distributed learning laws reduce to instar and outstar equations, and dART reduces to fuzzy ART. Similarly, with coding that is WTA during training but distributed during testing, the dARTMAP algorithm specified here reduces to ARTMAP-IC, and further reduces to fuzzy ARTMAP with coding that is WTA during both testing and training.

1.2. Distributed ARTMAP design choices

An ART module is embedded as the primary component of ARTMAP, and similarly an unsupervised dART module is embedded in a supervised dARTMAP network. In applications, ARTMAP requires few design choices: the number of coding nodes is determined by on-line performance, and the default network parameters work well in most settings. In contrast, a general dARTMAP system presents the user with a far greater array of choices, due to the new degrees of freedom afforded by distributed code possibilities. In practice, a number of the 'obvious' design choices have failed to produce good performance in simulation studies.

The present article presents one family of dARTMAP networks that have performed well in pilot studies. In particular, dARTMAP retains fuzzy ARTMAP test set accuracy while significantly reducing network size. A self-contained dARTMAP algorithm is designed both to expedite ready implementation and to foster the development of alternative designs adapted to the demands of new applications.

1.3. Outline

A number of computational devices that were not part of the more general distributed ART theory were found to be useful in dARTMAP simulations. These include a new rule characterizing the content-addressable memory stored at the coding field in response to a given input (Section 2.1), an internal control device that causes the system to alternate between distributed and WTA coding modes (Section 2.2), and credit assignment and instance counting (Section 2.3).

A geometric representation aids the visualization of distributed ARTMAP computational dynamics. Since the algorithm reduces to fuzzy ARTMAP when coding is WTA, the geometric characterization of dARTMAP builds upon the geometry of fuzzy ARTMAP, which represents weight vectors as category boxes in input space (Section 3.1). The relationship between these boxes and a system input determines the order in which categories are searched (Section 3.2), and box expansion represents weight changes during WTA learning (Section 3.3).

Distributed ARTMAP replaces the long-term memory weights of fuzzy ARTMAP with dynamic weights, which depend on short-term memory coding node activations, as well as long-term memory (Section 4.1). The corresponding geometric representation replaces each fuzzy ARTMAP category box with a nested family of boxes, one for each coding node activation value (Section 4.2). Some or all of these coding boxes may expand during dARTMAP learning, but the geometry shows how the system preserves dynamic range with fast as well as slow learning (Section 4.3). The rule in the dARTMAP algorithm that characterizes the signal transmitted to the coding field in response to a given input admits a geometric interpretation (Section 4.4), as does the rule characterizing the response of the content-addressable memory to the incoming signal (Section 4.5).

The dARTMAP algorithm includes the computational elements that were useful in simulation studies. For clarity, the training (Section 5.1) and testing (Section 5.2) portions of the algorithm are listed separately. In the version presented here, the dARTMAP algorithm is feedforward during testing.

A series of simulations indicate how the dARTMAP algorithm works. Distributed prediction in the basic algorithm reduces network size, but this system uses only binary connections from the coding field to the output field (Section 6.1). Performance can be improved by augmenting the trained dARTMAP system with a linear output map such as Adaline (Section 6.2). Other simulations analyze the role

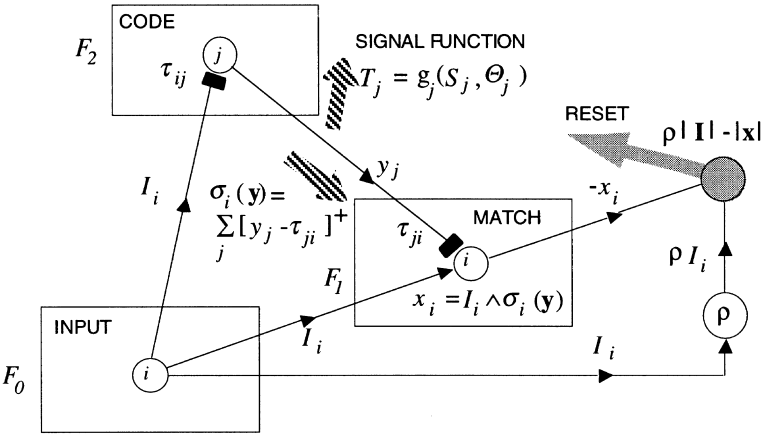

Fig. 1. Distributed ART network. A dART coding field F_2 receives signals directly from an input field F_0. The $F_0 \rightarrow F_2$ signal T_j is a function of a phasic component S_j, which depends on the current input, and a tonic component Θ_j, which is independent of the input. A CAM rule defines the transformation from signals T_j to the F_2 code $\mathbf{y}$, which may be arbitrarily distributed. Activity $\mathbf{x}$ at the field F_1 reflects a match between bottom-up input $\mathbf{I}$ and top-down input σ. The active code is reset when $\mathbf{x}$ fails to meet the vigilance matching criterion, determined by parameter ρ. Long-term memory is stored as $F_0 \rightarrow F_2$ thresholds τ_{ij}, which adapt according to a distributed instar learning law, and $F_2 \rightarrow F_1$ thresholds τ_{ji}, which adapt according to a distributed outstar learning law.

of dARTMAP learning that takes place in the distributed mode, as opposed to the WTA mode (Section 6.3). By varying the degree of pattern contrast in the content-addressable memory system, dARTMAP performance can be improved, without increasing network size (Section 6.4). A statistical analysis confirms the significance of simulation findings (Section 6.5).

Finally, a step-by-step presentation of the geometry of dARTMAP learning demonstrates the detailed mechanism of system dynamics (Section 7). Section 8 concludes with a discussion of possible dARTMAP variations and directions for future research.

2. CAM rules, coding modes, and credit assignment

The unsupervised distributed ART network (Carpenter, 1996, 1997) features a number of innovations that differentiate it from previous ART networks, including a new architecture configuration and distributed instar and outstar learning laws (Fig. 1). In order to stabilize fast learning with distributed codes, dART represents the unit of long-term memory (LTM) as a subtractive threshold rather than a traditional multiplicative weight. Despite their different architectures, a dART algorithm reduces to fuzzy ART when coding is WTA. While a dART module is the basic component of a supervised dARTMAP system, the algorithm specified in Section 5 also employs additional devices not included in the previous distributed ART description. These features, including a new rule defining coding field activation, alternation between WTA and distributed coding modes, and credit assignment, will now be described.

2.1. Increased gradient content-addressable memory rule

A neural network field of strongly competitive nodes can, once activated by an initial input, maintain a short-term memory (STM) activation pattern even after the input is removed. A new input then requires some active reset process before it can instate a different code, or content-addressable memory (CAM). A *CAM rule* specifies a function that characterizes the steady-state STM response to a given vector of inputs converging upon a field of neurons.

Traditional CAM rules include: McCulloch–Pitts activation, which makes STM proportional to input (McCulloch & Pitts, 1943); a power rule, which makes STM proportional to input raised to a power p; and a WTA rule, which concentrates all activation at the node receiving the largest net input. Other CAM rules include Gaussian activation functions, as used, for example, in radial basis function networks (Moody & Darken, 1989). A power rule reduces to a McCulloch–Pitts rule when $p = 1$ and converges to a WTA rule as $p \rightarrow \infty$. Moving p from 0 towards infinity produces a stored STM pattern that is a progressively contrast-enhanced transformation of the input vector. In many examples, however, a power rule is problematic because differences among input components are small. A CAM system may then require unreasonably large powers p to produce significant differences among STM activations.

The CAM rule used in the dARTMAP algorithm is designed to enhance input differences as represented in the distributed internal code without raising input components to high powers. It is therefore called the *increased gradient CAM rule*. Beyond its role in the present system, this rule is useful for defining the steady-state activation function in other neural networks. The increased gradient rule includes a power p for contrast control. The role of p is analogous to the role of variance in Gaussian activation functions (Hertz et al., 1991; Moody & Darken, 1989). A geometric representation of dARTMAP provides a natural interpretation of the increased gradient CAM rule (Section 4.5).

2.2. Distributed and winner-take-all coding modes

The increased gradient CAM rule solves a pattern separation problem that often arises in neural systems, where each element has a limited dynamic range. A second common problem is how to choose the size of a neural network. In an MLP, for example, deciding on the number of hidden units is a critical design choice. With WTA coding, ARTMAP determines network size by adding category nodes incrementally, to meet the demands of on-line predictive accuracy. Some types of MLP networks have also been designed to add hidden units incrementally. A cascade correlation architecture, for example, creates a hierarchy of single-unit hidden layers until the error criterion is met (Fahlman & Lebiere, 1990), but weights in all lower layers are frozen during learning associated with the top layer.

With distributed coding, a dARTMAP network could, in principle, operate with a field of coding nodes that are fixed a priori. In practice, this type of network did not produce satisfactory results in simulation studies, where fast learning tended to make the learned representations too uniform. To solve this problem, the dARTMAP algorithm alternates between distributed and WTA coding modes, as follows.

Each dARTMAP input first activates a distributed code. If this code produces a correct prediction, learning proceeds in the *distributed coding mode*. If the prediction is incorrect, the network resets the active code via ARTMAP *match tracking feedback* (Carpenter et al., 1991a). In ARTMAP networks, the reset process triggers a search for a category node that can successfully code the current input. In dARTMAP, reset also places the system in a *WTA coding mode* for the duration of the search. The switch from a distributed mode to a WTA mode could be implemented in a competitive network by means of a nonspecific signal that increases the strength of intrafield inhibition (Ellias & Grossberg, 1975; Grossberg, 1973). Such an arousal signal might be interpreted as an increase in overall attentiveness in response to an error signal or alarm, the computational result being a sharpened focus on the most salient input features.

In WTA mode, dARTMAP can, like ARTMAP, add nodes incrementally as needed. When a coding node is added to the network, it becomes permanently associated with the output class that is active at the time. From then on, the network predicts this class whenever the same coding node is chosen in WTA mode. In distributed mode, STM activations across all nodes that project to a given output class provide evidence in favor of that outcome. Despite its computational advantages, the WTA possibility implies that dARTMAP coding is not fully distributed all the time, indicating one possible direction for future system modifications.

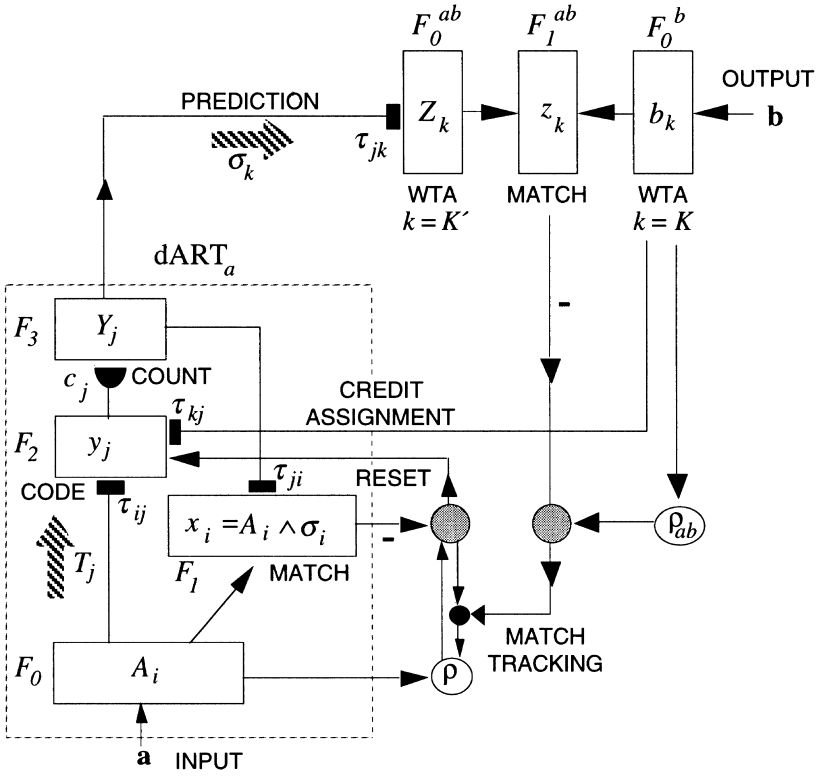

Fig. 2. Distributed ARTMAP network. A complement-coded input **A** activates a distributed F_2 code **y**, which in turn is filtered through counting weights c_j to produce the F_3 activation **Y**. The WTA field F_0^{ab} activates the node $k = K'$ that receives the largest input σ_k from F_3, representing the predicted output class. During training, activation at the field F_1^{ab} determines whether the predicted output class $k = K'$ matches the actual output class $k = K$, which is represented at the field F_0^b. Adaptation in paths from F_0^b to the coding field F_2 realizes credit assignment. A mismatch at F_1^{ab} causes a match tracking signal to raise ART$_a$ vigilance ρ just enough to reset the active code.

2.3. Credit assignment, instance counting, and match tracking

When a dARTMAP network makes a distributed prediction, some of the active coding nodes may be linked to an incorrect outcome. In a real-time network, a feedback loop for credit assignment would suppress activation in these nodes during training (Fig. 2). Credit assignment allows learning to enhance only those portions of an active code that are associated with the correct outcome. This procedure is similar to credit assignment algorithms widely used in other neural networks (e.g., Williamson, 1996) and genetic algorithms (e.g., Booker et al., 1989).

The current simulations were also found to benefit from design features used in the ARTMAP-IC network. These include instance counting of category exemplars and the MT − match tracking search rule. Instance counting biases output predictions according to previous coding node activations summed over training set inputs. The MT − search rule generally improves memory compression compared to the original ARTMAP match tracking algorithm (MT +). It also permits a system to encode inconsistent cases, where two identical training set inputs are associated with different outcomes. Inconsistent cases are common in medical databases, for example.

Aspects of the dARTMAP algorithm such as the increased gradient CAM rule, the combination of WTA with distributed coding during training, credit assignment, and instance counting are not necessarily fundamental principles intrinsic to the class of all dARTMAP networks. Rather, they are developed for the pragmatic purpose of defining one set of dARTMAP systems with the desired computational properties.

A real-time neural network can implement the computations of the dARTMAP algorithm (Fig. 2). Because the algorithm considers only the case where the output vector represents discrete classes, it does not require all the variables shown in the network diagram. A geometric representation of dARTMAP dynamics (Section 4) helps visualize and motivate computations of the algorithm (Section 5). Because dARTMAP reduces to fuzzy ARTMAP when coding is WTA, the geometry of dARTMAP generalizes the geometry of fuzzy ARTMAP, which will first be reviewed (Section 3). Where possible, dARTMAP retains fuzzy ARTMAP notation as well.

3. Fuzzy ARTMAP geometry

Both fuzzy ARTMAP and dARTMAP employ an input preprocessing device called *complement coding*. Complement coding creates a system input vector **A** equal to the concatenation of the original M-dimensional input **a**, where $0 \le a_i \le 1$, and its complement $\mathbf{a}^c$, where $(\mathbf{a}^c)_i \equiv (1 - a_i)$. The input **A** thus positively represents both 'present features' (**a**) and 'absent features' ($\mathbf{a}^c$). In addition, using

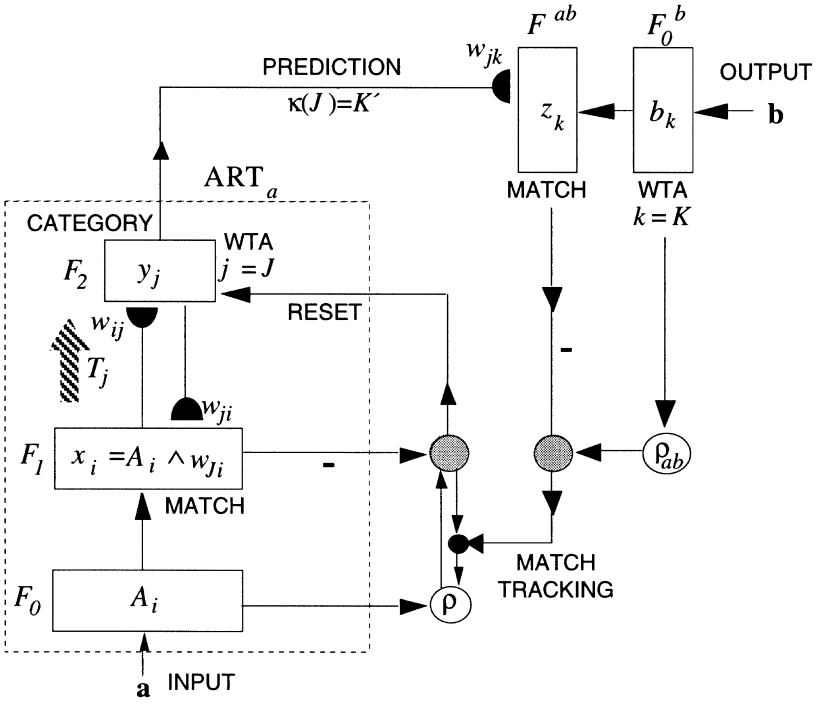

Fig. 3. Simplified fuzzy ARTMAP network. For an output class prediction task, the network does not require the full ART_b network of a general fuzzy ARTMAP system (Carpenter et al., 1992). A complement-coded input **A** makes a WTA category selection ($j = J$) at F_2, which predicts an output class $K' = \kappa(J)$. During training, if K' is not the same as the index K of the actual output class, then match tracking raises vigilance enough to trigger a search for a different F_2 node. The dARTMAP network in Fig. 2 has two additional fields: F_3, for instance counting; and F_0^{ab}, for translating distributed F_3 output into a WTA prediction.

the *city-block norm* defined by $|\mathbf{v}| \equiv \Sigma_i |v_i|$, complement coding serves to normalize inputs, since then

$$|\mathbf{A}| = \sum_{i=1}^{2M} A_i = \sum_{i=1}^{M} \big(a_i + (1 - a_i)\big) = M.$$

Complement coding allows weight vectors to be represented geometrically as boxes in the M-dimensional space of the vector **a**. The doubled system input vectors **A** produce the endpoints of a set of intervals that define the edges of each box, as described in the following section.

3.1. ARTMAP category boxes

During fuzzy ARTMAP learning, 2M-dimensional complement coded inputs **A** give rise to 2M-dimensional weight vectors $\mathbf{w}_j \equiv (w_{1j}, \ldots, w_{ij}, \ldots, w_{2M,j})$, one for each F_2 category node j (Fig. 3). Bottom-up weights equal top-down weights, so $\mathbf{w}_j$ may stand for both. For $i = 1, \ldots, M$, weight w_{ij} intuitively represents the degree to which the ith feature is consistently present in the inputs **a** coded by the jth category; and $w_{i+M,j}$ represents the degree to which the ith feature is consistently absent. When both w_{ij} and $w_{i+M,j}$ become small, the network treats the size of a_i as unpredictive with respect to the jth category.

The weight vector $\mathbf{w}_j$ is depicted geometrically as an M-dimensional *category box* R_j with edges defined by the intervals $\left[w_{ij},\ w^c_{i+M,j}\right]$. The box R_j is the set of points **q** for which $w_{ij} \le q_i \le (1 - w_{i+M,j})$ (Fig. 4a). The size $|R_j|$ is defined as the sum of the lengths of the box's M defining intervals. Thus,

$$|R_j| = \sum_{i=1}^{M} \big(\big(1 - w_{i+M,j}\big) - w_{ij}\big) = M - |\mathbf{w}_j|.$$

When a node is first activated, or *committed*, the active node ($j = J$) becomes permanently associated with the active output class ($k = K = \kappa(J)$). The network adds a committed node when it determines that previously active nodes cannot adequately represent the current input. The number of

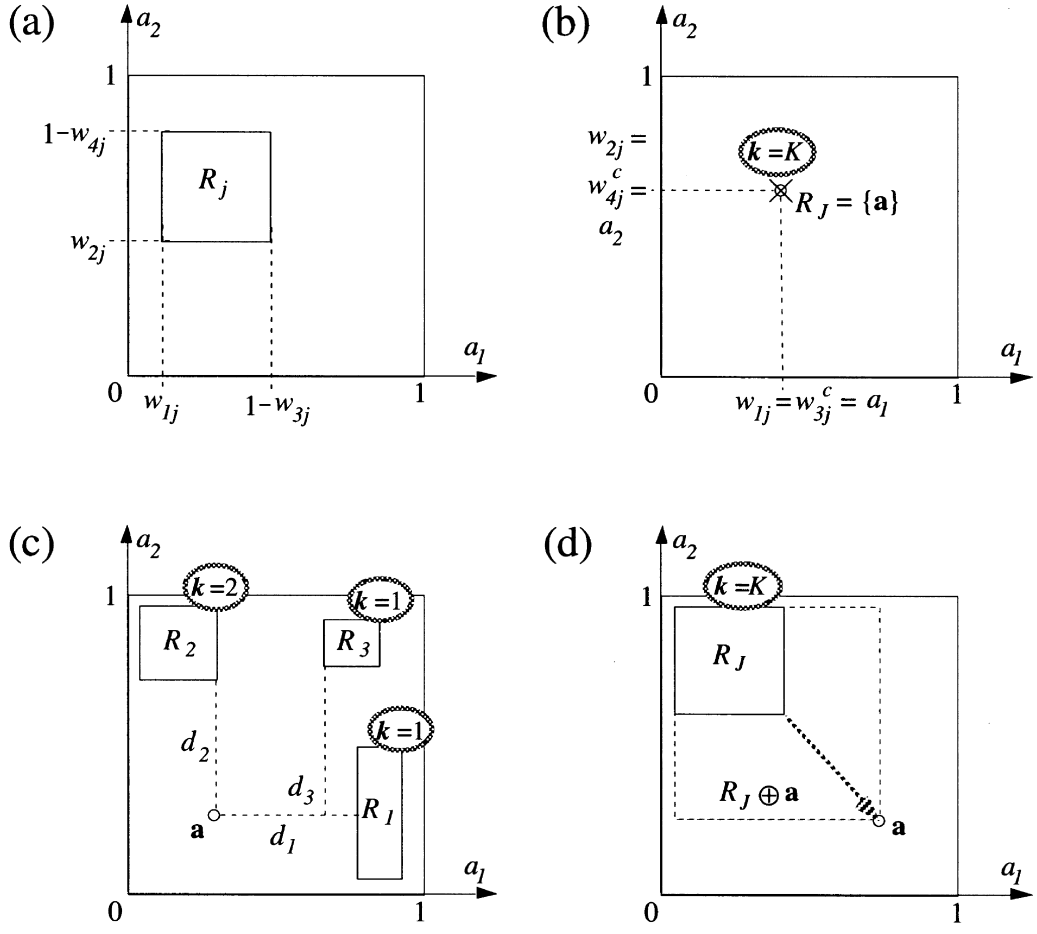

Fig. 4. Fuzzy ARTMAP geometry, in the conservative limit with fast learning and a choice-by-difference signal function. (a) A category box R_j represents the complement-coded weight vector $\mathbf{w}_j$. (b) When a node J is first committed, $w_{iJ} = w^c_{i+M,J} = a_i$ for $i = 1, \ldots, M$, so the category box R_J is the point box $\{\mathbf{a}\}$. Category J becomes permanently mapped to the current output class $\kappa(J) = K$. Point boxes are drawn as $\times$ and the current input **a** is drawn as $\bigcirc$. (c) If **a** is not contained in any box R_j, categories are searched in order of their boxes' distance to **a**. (d) Once a category J that makes the correct output prediction is found to meet the vigilance matching criterion, R_J expands just enough to include **a**.

committed nodes (C) grows incrementally during training. When a node j is *uncommitted*, $w_{ij} \equiv 1$. Then, when the node first becomes committed, R_j equals the *point box* $\{\mathbf{a}\}$, where $w_{ij} = w^{c}_{i+M,j} = a_i$ $(i = 1, \ \ldots, \ M)$ (Fig. 4b).

3.2. ARTMAP order of search

When a committed F_2 node becomes active and incorrectly predicts the output class, fuzzy ARTMAP triggers a search process called *match tracking*. Match tracking increases the *vigilance* matching parameter ρ just enough to reset the active category. Search ends when the chosen node predicts the correct output class $k = K$, provided that the vigilance matching criterion is also satisfied.

Fuzzy ARTMAP geometry serves to illustrate the order in which nodes are searched. Let T_j denote the signal sent to the jth node of the category field F_2. The function that determines T_j depends jointly on the current input $\mathbf{a}$ and on the learned weight vector $\mathbf{w}_j$. With WTA coding, F_2 nodes become active in order of the size of T_j, starting with the largest. The geometric version of a *choice-by-difference* signal function (Carpenter & Gjaja, 1994) sets:

$$T_j = M(2-\alpha) - d\big(R_j, \ \mathbf{a}\big) - \alpha|R_j|, \tag{1}$$

where $\alpha \in (0, 1)$. In Eq. (1), $d(R_j, \mathbf{a}) \equiv d_j$ denotes the city-block distance from $\mathbf{a}$ to R_j. That is, $d(R_j, \mathbf{a}) = |R_j \oplus \mathbf{a}| - |R_j|$, where $R_j \oplus \mathbf{a}$ is the smallest box enclosing both R_j and $\mathbf{a}$. When j is an uncommitted node, $T_j \equiv T^{u}$. At an uncommitted node j, $w_{ij} \equiv 1$, so formally $|R_j| = M - |w_j| = -M$ and $d(R_j, \mathbf{a}) = |\{\mathbf{a}\}| - |R_j| = M$. Thus, by Eq. (1), $T^{u} = M(2 - \alpha) - M + \alpha M = M$.

For boxes R_j that contain $\mathbf{a}$, $d(R_j, \mathbf{a}) = 0$, so $T_j = M(2 - \alpha) - \alpha|R_j|$. When $\mathbf{a}$ is contained in one or more boxes R_j and $\alpha = 0^{+}$, nodes first become active in order of the sizes of these R_j, starting with the smallest. When $\mathbf{a}$ is not contained in any box and $\alpha = 0^{+}$, $T_j \cong 2M - d(R_j, \mathbf{a})$ so nodes become active in order of the distances from $\mathbf{a}$ to R_j, starting with the nearest. In Fig. 4c, nodes would become active in the order $j = 2, 1, 3$.

Search continues until the chosen node makes the correct prediction and satisfies the vigilance matching criterion. If all committed nodes with $T_j \geq T^{u}$ are reset, the network chooses a previously uncommitted node, which learns the correct prediction.

3.3. ARTMAP winner-take-all learning

During fast learning with node $j = J$ active, R_J expands just enough to include $\mathbf{a}$; that is, the category box grows to $R_J \oplus \mathbf{a}$ (Fig. 4d). The total weight increase therefore equals the initial distance $d_J = d(R_J, \mathbf{a})$. Thus, as $\alpha \rightarrow 0^{+}$, selecting the closest box via the signal function T_j [Eq. (1)] is equivalent to selecting the node where weights will be minimally changed, or maximally conserved. The parameter choice $\alpha = 0^{+}$ is therefore called the *conservative limit* (Carpenter et al., 1991b). At each stage of learning, R_j is the smallest box that contains all the training set inputs $\mathbf{a}$ that have been coded by category j without reset.

In summary, with fast learning in the conservative limit, a fuzzy ARTMAP input $\mathbf{a}$ chooses boxes in turn, starting with the closest and, if $\mathbf{a}$ is in more than one box, starting with the smallest box that contains $\mathbf{a}$. A predictive error increases vigilance enough to cause reset. Search ends when the chosen box makes the correct prediction and satisfies the matching criterion. For a committed node, the box expands until it includes $\mathbf{a}$, and for an uncommitted node, a point box is established.

4. Distributed ARTMAP geometry

The geometric representation of dARTMAP builds upon the geometry of fuzzy ARTMAP. With distributed coding, steady-state activations y_j at F_2 nodes may take on any value between 0 and 1, in contrast to WTA coding, which produces only binary activations. Distributed ARTMAP therefore replaces the single fuzzy ARTMAP category box R_j with a nested family of coding boxes $R_j(y_j)$, with the smallest box, $R_j(1)$, corresponding to R_j. As with fuzzy ARTMAP, dARTMAP geometry illustrates the dynamics of code selection, search, and learning.

For a given system input $\mathbf{a}$, the vector transmitted to the coding field F_2 is determined by a chosen signal rule (Section 4.4). At F_2, the resulting distributed steady-state activation pattern $\mathbf{y}$ is determined by a chosen CAM rule (Section 4.5). If that code predicts the correct output class K, learning within the dART module ensues (Section 4.3). Distributed learning is depicted geometrically in terms of families of coding boxes (Section 4.2), which represent dARTMAP dynamic weights (Section 4.1). If the distributed code makes an incorrect prediction, the network reverts to a WTA mode. For the rest of the search, system dynamics then closely resemble those of fuzzy ARTMAP. Committed F_2 nodes are added only in WTA mode, with each newly committed node producing a geometric point box associated with a unique output class. During testing, coding is always distributed and the network operates as a feedforward system.

4.1. Distributed ARTMAP dynamic weights

The key step in the transformation from fuzzy ART to distributed ART replaces the traditional LTM path weights w_{ij}/w_{ji} with *dynamic weights*. Each dynamic weight is a function of a coding node activation y_j (STM), as well as a subtractive threshold τ_{ij}/τ_{ji} (LTM). The formal substitutions that convert a fuzzy ART algorithm into a dART algorithm are described as:

$$w_{ij} \rightarrow \big[y_j - \tau_{ij}\big]^{+} \tag{2}$$

in bottom-up paths and:

$$w_{ji} \rightarrow \big[y_j - \tau_{ji}\big]^{+} \tag{3}$$

in top-down paths (Figs. 1 and 3). In Eqs. (2) and (3), $[\xi]^+ \equiv \max\{\xi, 0\}$ denotes the rectification operator. The LTM thresholds τ_{ij} and τ_{ji}, initially 0, rise toward a maximum of 1 during learning.

With WTA coding and setting $y_j \equiv 1$ in Eqs. (2) and (3), distributed ART reduces to fuzzy ART. WTA coding limits learned changes to processes associated with the single active category node, thereby stabilizing memory by imposing an upper bound on the total change. With distributed coding, dynamic weights take over responsibility for bounding the total learned change. A dynamic weight $[y_j - \tau_{ij}]^+$ is positive only when coding node activation y_j exceeds the adaptive threshold τ_{ij}. Only then does the dART learning law permit the threshold to increase. This restriction imposes an upper bound on total threshold changes: the sum $\Sigma_{j=1}^{C} \Delta\tau_{ij}$ is bounded above by $\Sigma_{j=1}^{C} [y_j - \tau_{ij}]^+$, which in turn is bounded above by 1, since $|\mathbf{y}| = \Sigma_{j=1}^{C} y_j = 1$. On the other hand, the coding capacity of each threshold set $\{\tau_{i1}, \ldots, \tau_{ij}, \ldots, \tau_{iC}\}$ is limited only by the number of committed coding nodes, which may be arbitrarily large. Dynamic weights thereby allow the limited capacity of STM to impose an upper bound on adaptive changes in an LTM system of unlimited capacity, even when coding field activation is fully distributed and fast learning causes all variables to reach asymptote on every input presentation.

4.2. Distributed ARTMAP coding boxes and matching boxes

Consider now the geometry of dARTMAP dynamics in paths from the input field F_0 to the coding field F_2 (Fig. 2). An entire family of dynamic weights $[y_j - \tau_{ij}]^+$, one for each $y_j \in [0, 1]$, replaces each single fuzzy ARTMAP path weight w_{ij}. A corresponding family of nested *coding boxes* $R_j(y_j)$ thus replaces the single category box R_j. The box $R_j(y_j)$ equals the set of points $\mathbf{q}$ for which $[y_j - \tau_{ij}]^+ \leq q_i \leq (1 - [y_j - \tau_{i+M,j}]^+)$ (Fig. 5a).

In dARTMAP, initial code selection depends on the boxes $R_j(y_j)$ only for the case where $y_j = 1$: the network models reset as a process that breaks competitive feedback loops at F_2 by momentarily saturating all y_j activations. Therefore, since a CAM system maintains an F_2 activation pattern $\mathbf{y}$ until the next reset, the boxes $R_j(1)$ determine the sequence of stored codes. Just as the dynamic weight $[y_j - \tau_{ij}]^+$ is formally equivalent to the weight w_{ij} when $y_j \equiv 1$, the dARTMAP coding box $R_j(1)$ is formally equivalent to the fuzzy ARTMAP category box R_j. Once a code $\mathbf{y}$ has been established, the boxes $R_j(y_j)$ control the dynamics of search and learning.

In fuzzy ARTMAP, the dynamics of category search are determined by the degree of match between bottom-up signals from the input field F_0 and top-down signals from the category field F_2, calculated at the matching field F_1 (Fig. 3). When the Jth F_2 node is active, the top-down signal to the ith F_1 node equals w_{Ji}. Since $w_{ji} \equiv w_{ij}$, the category boxes R_j

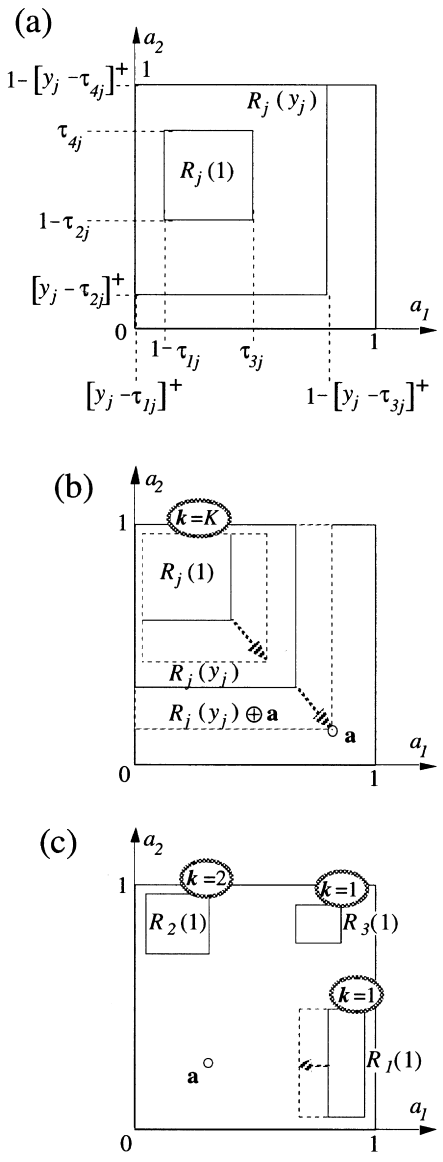

Fig. 5. Distributed ARTMAP geometry, in the conservative limit with fast learning, a choice-by-difference signal function, and the increased gradient CAM rule with $p = 1$. (a) A coding box $R_j(y_j)$ represents the complement-coded dynamic weight vector, with $[y_j - \tau_{ij}]^+$ replacing the fuzzy ARTMAP weight w_{ij} (Fig. 4a). (b) If a distributed code makes the correct output prediction, $\mathbf{y}$ is recalculated for credit assignment. Then, all boxes $R_j(y_j)$ expand just enough to include $\mathbf{a}$. The box $R_j(1)$ expands to meet $\mathbf{a}$ only if $y_j = 1$. (c) For the system depicted here, the distributed code $\mathbf{y} = (0.37, 0.42, 0.21)$ makes the correct prediction $k = 1$. After credit assignment, $\mathbf{y} = (0.64, 0.0, 0.36)$. Then $R_3(y_3)$ includes $\mathbf{a}$, so neither $R_3(y_3)$ nor $R_3(1)$ expands during learning. On the other hand, since the left edge of box $R_1(y_1)$ is a distance of 0.13 units from $\mathbf{a}$, box $R_1(y_1)$ does expand to meet $\mathbf{a}$, causing $R_1(1)$ also to grow as τ_{11} increases by 0.13 units.

(Fig. 4a) can represent the geometry of top-down matching as well as bottom-up category choice.

In dARTMAP, top-down signals to F_1 originate from the field F_3, where the F_2 code $\mathbf{y}$ is transformed into a new normalized coding vector $\mathbf{Y}$ by *instance counting*. That is, Y_j is proportional to $c_j y_j$, where the *counting weight* c_j reflects the sum of prior activations y_j during training

(Fig. 2). The total signal from F_3 to the ith F_1 node is the dynamic weight sum $\sigma_i(\mathbf{Y}) \equiv \Sigma_{j=1}^{C} [Y_j - \tau_{ji}]^+$.

Since the dARTMAP top-down signal to an F_1 node depends on the entire distributed code rather than a single dynamic weight, the coding boxes $R_j(y_j)$ cannot also represent the geometry of matching. Instead, the geometry of search is characterized by a family of *matching boxes* $R(\mathbf{Y})$, one for each vector $\mathbf{Y}$. The matching box $R(\mathbf{Y})$ equals the set of points $\mathbf{q}$ for which $\sigma_i(\mathbf{Y}) \leq q_i \leq (1 - \sigma_{i+M}(\mathbf{Y}))$ $(i = 1, \ldots, M)$. With the total signal $\sigma_i(\mathbf{Y})$ to the ith F_1 node replacing the w_{Ji}, dARTMAP matching and search are analogous to the corresponding processes in fuzzy ARTMAP. See Carpenter (1997) for a more complete description of the geometry of distributed search.

4.3. Distributed ARTMAP learning

The coding boxes $R_j(y_j)$ provide a geometric representation of dARTMAP learning, with y_j equal to the jth component of the currently active F_2 code $\mathbf{y}$. During distributed learning, each box $R_j(y_j)$ that does not already contain the input $\mathbf{a}$ expands just enough to include $\mathbf{a}$ (Fig. 5b). Unless coding is WTA and j is the chosen node, y_j is less than 1, making $R_j(y_j)$ larger than $R_j(1)$. In this case, unless $R_j(1)$ initially contains $\mathbf{a}$, $R_j(1)$ still does not contain $\mathbf{a}$ after learning, even with all LTM variables reaching asymptote. The limited capacity of STM hereby permits the code-selecting boxes $R_j(1)$ to conserve their capacity during fast as well as slow learning.

Fig. 5c illustrates an example where, at the time $\mathbf{a}$ is presented, three coding nodes have previously been committed ($C = 3$). Nodes $j = 1$ and $j = 3$ map to the output class $k = 1$ and node $j = 2$ maps to the output class $k = 2$. The increased gradient CAM rule distributes activation y_j across all three nodes, with $1 > y_2 > y_1 > y_3 > 0$. Before the output prediction is made, y_j is multiplied by the counting weight c_j. The subsequent field pools the distributed activations to make an output class prediction $k = K'$. Suppose that $K' = K = 1$, i.e. that the system correctly predicts class $k = 1$. Feedback from the output layer F_0^b to the coding layer F_2 then implements credit assignment (Fig. 2), which restricts adaptation to paths to and from nodes j associated with the correct prediction K. Credit assignment would permit the boxes $R_1(y_1)$ and $R_3(y_3)$ to expand towards $\mathbf{a}$. In fact, $R_3(y_3)$ already contains $\mathbf{a}$ at the outset. Thus, only $R_1(y_1)$ expands to include $\mathbf{a}$, as τ_{11} increases, and $R_1(1)$ expands as well (Fig. 5c).

In a real-time dARTMAP network, thresholds in bottom-up paths from F_0 to F_2 adapt according to a *distributed instar learning law*:

$$\frac{\mathrm{d}}{\mathrm{d}t}\tau_{ij} = [y_j - \tau_{ij} - A_i]^+ = [[y_j - \tau_{ij}]^+ - A_i]^+ \tag{4}$$

(Carpenter, 1997). Eq. (4) states that the threshold τ_{ij} will grow until its dynamic weight $[y_j - \tau_{ij}]^+$ shrinks down to meet the $F_0 \rightarrow F_2$ input A_i. Geometrically, $R_j(y_j)$ expands toward $R_j(y_j) \oplus \mathbf{a}$ as the bottom-up thresholds $\tau_{1j}, \ldots, \tau_{2M,j}$ to the jth F_2 node increase.

Thresholds in top-down paths from F_3 to F_1 adapt according to a *distributed outstar learning law*:

$$\frac{\mathrm{d}}{\mathrm{d}t}\tau_{ji} = [Y_j - \tau_{ji}]^+ (\sigma_i(\mathbf{Y}) - x_i) = [Y_j - \tau_{ji}]^+ [\sigma_i(\mathbf{Y}) - A_i]^+ \tag{5}$$

(Carpenter, 1994). Eq. (5) states that the threshold τ_{ji} will grow until the $F_3 \rightarrow F_1$ input $\sigma_i(\mathbf{Y})$ shrinks down to meet the $F_0 \rightarrow F_1$ input A_i. The amount a given threshold will increase during learning depends upon the contribution of $[Y_j - \tau_{ji}]^+$ to the dynamic weight sum $\sigma_i(\mathbf{Y})$. In particular, τ_{ji} remains constant if $[Y_j - \tau_{ji}]^+ = 0$, i.e. if $Y_j \leq \tau_{ji}$. Geometrically, $R(\mathbf{Y})$ expands toward $R(\mathbf{Y}) \oplus \mathbf{a}$, as top-down thresholds τ_{ji} increase in parallel.

The piecewise-linear equations [Eqs. (4) and (5)] can be solved exactly: the adaptive threshold update equations in Step 8 of the dARTMAP algorithm (Section 5.1) represent closed form solutions of the differential equations, not approximations. These solutions are valid for all initial values and all learning rates, including the fast-learn limit.

4.4. Distributed ARTMAP signal rule

The signal T_j from the dARTMAP input field F_0 to the jth node of the coding field F_2 is a function $g_j(S_j, \Theta_j)$, where the *phasic component* S_j depends on the input $\mathbf{A}$, the *tonic component* Θ_j is independent of $\mathbf{A}$, $g_j(0, 0) = 0$, and $(\partial g_j/\partial S_j) > (\partial g_j/\partial \Theta_j) > 0$ for $S_j > 0$ and $\Theta_j > 0$. Components S_j and Θ_j also depend on adaptive thresholds τ_{ij} in paths from F_0 to the jth F_2 node and on target node activation y_j. Simulations in this paper use a *choice-by-difference signal rule*:

$$T_j(y_j) = S_j(y_j) + (1 - \alpha)\Theta_j(y_j), \tag{6}$$

with the signal rule parameter $\alpha \in (0, 1)$. In Eq. (6), the phasic component is defined by:

$$S_j(y_j) = \sum_{i=1}^{2M} A_i \wedge [y_j - \tau_{ij}]^+ \tag{7}$$

and the tonic component is defined by:

$$\Theta_j(y_j) = \sum_{i=1}^{2M} \tau_{ij} \wedge y_j, \tag{8}$$

where $a \wedge b \equiv \min\{a, b\}$.

At first, the definition of T_j appears to be circular: the signal function that determines the code $\mathbf{y}$ also seems to depend upon $\mathbf{y}$. Recall, however, that dARTMAP reset momentarily sets all $y_j = 1$. The content-addressable memory at F_2 is therefore determined by the values $T_j(1)$ at that time, and the ensuing stored code $\mathbf{y}$ then remains constant until the next reset. Between resets, signals $T_j(y_j)$ control

the internal computations underlying distributed search and learning. Since $|\mathbf{y}| = 1$, active F_2 nodes typically represent a concentrated subset of the field's total capacity, which can be arbitrarily large. Correspondingly, the signal $T_j(y_j)$ between resets is, on average, a small fraction of $T_j(1)$.

In geometric terms,

$$T_j(1) = (2-\alpha)M - d\big(R_j(1),\ \mathbf{a}\big) - \alpha|R_j(1)|, \tag{9}$$

as in Eq. (1). Thus, the signal rule favors nodes with small coding boxes ($|R_j(1)| \cong 0$) that are close to the input **a** ($d(R_j(1), \mathbf{a}) \cong 0$). In the conservative limit, where $\alpha = 0^+$, the system seeks primarily to minimize the distance from **a** to $R_j(1)$. In this case, the size of $R_j(1)$ is used only to break ties, as when **a** is contained in more than one box.

4.5. Distributed ARTMAP CAM rule

The CAM rule that specifies dARTMAP activation at the coding field approximates the steady-state response that a field of competitive nodes would make to the $F_0 \rightarrow F_2$ input vector $\mathbf{T}(1)$, evaluated at the time of reset. In general, the jth component of the F_2 code **y** is determined by a function $y_j = f_j(T_1, \ldots, T_C)$, with $\partial f_j/\partial T_j \geq 0$ and with $T_j = T_j(1)$ at the time of reset. Distributed ARTMAP simulations here use an *increased gradient* CAM rule to determine the F_2 activation vector **y**. The coding function f_j that defines this rule is specified formally in Step 2 of the algorithm (Section 5.1) using the terms $[(2-\alpha)M - T_j(1)]$. The increased gradient CAM rule is here given a geometric interpretation, in terms of coding boxes, by observing that:

$$(2-\alpha)M - T_j(1) = d\big(R_j(1),\ \mathbf{a}\big) + \alpha|R_j(1)| \tag{10}$$

[Eq. (9)].

In a real-time network, committed nodes would compete with uncommitted nodes for coding field activation. In order to simulate this situation in the dARTMAP algorithm, a committed node j is allowed to become active at reset only when $T_j(1)$ is at least as great as the signal T^{u} that would be sent to an uncommitted node, where all thresholds $\tau_{ij} \equiv 0$. The phasic signal to an uncommitted node j is $S_j(1) = |\mathbf{A}| = M$ [Eq. (7)] and the tonic signal is $\Theta_j(1) = 0$ [Eq. (8)], so the total choice-by-difference signal is $T_j(1)|_{\tau_{ij}\equiv 0} \equiv T^{\mathrm{u}} = M$ [Eq. (6)]. For a given input **a**, the *CAM index set* $\Lambda \equiv \{j = 1, \ldots, C\colon T_j(1) \geq T^{\mathrm{u}}\}$ denotes the F_2 nodes that may become active during distributed coding. With the choice-by-difference signal function [Eq. (6)], $T_j(1) \geq (1-\alpha)M = (1-\alpha)T^{\mathrm{u}}$. This indicates that smaller values of α predispose the network to activate larger numbers of committed nodes during distributed coding. In the conservative limit, where $\alpha = 0^+$, $T_j(1) \cong S_j(1) + \Theta_j(1) \geq T^{\mathrm{u}}$, so $\Lambda = \{1, \ldots, C\}$ and activation is distributed across all committed nodes.

In the conservative limit, the increased gradient CAM rule can be visualized geometrically as follows.

(**a**) If input **a** is not in any box $R_j(1)$, $(2-\alpha)M - T_j(1) \cong d(R_j(1), \mathbf{a})$ for all j. Then:

$$y_j \cong \begin{cases} \dfrac{1}{1 + \displaystyle\sum_{\substack{\lambda\in\Lambda \\ \lambda\neq j}} \left[\dfrac{d\big(R_j(1),\ \mathbf{a}\big)}{d\big(R_\lambda(1),\ \mathbf{a}\big)}\right]^p} & \text{if } j \in \Lambda \\ 0 & \text{if } j \notin \Lambda \end{cases} = \tag{11}$$

$$\begin{cases} 1 & \text{if } \Lambda = \{j\} \\ \dfrac{\displaystyle\prod_{\substack{\mu\in\Lambda \\ \mu\neq j}} d\big(R_\mu(1),\ \mathbf{a}\big)^p}{\displaystyle\sum_{\lambda\in\Lambda}\prod_{\substack{\mu\in\Lambda \\ \mu\neq\lambda}} d\big(R_\mu(1),\ \mathbf{a}\big)^p} & \text{if } |\Lambda| \geq 2 \text{ and } j \in \Lambda, \\ 0 & \text{if } j \notin \Lambda \end{cases}$$

where the power p is greater than 0 (Fig. 6a).

(**b**) If **a** is contained in at least one box $R_j(1)$, $(2-\alpha)M - T_j(1) = \alpha|R_j(1)|$ for these boxes. In this case, let $\Lambda' \equiv \Lambda \cap \{j\colon \mathbf{a} \in R_j(1)\}$.

(**i**) If $|R_j(1)| > 0$ for all $j \in \Lambda'$, then:

$$y_j \cong \begin{cases} \dfrac{1}{1 + \displaystyle\sum_{\substack{\lambda\in\Lambda' \\ \lambda\neq j}} \left[\dfrac{|R_j(1)|}{|R_\lambda(1)|}\right]^p} & \text{if } j \in \Lambda' \\ 0 & \text{if } j \notin \Lambda' \end{cases} = \tag{12}$$

$$\begin{cases} 1 & \text{if } \Lambda' = \{j\} \\ \dfrac{\displaystyle\prod_{\substack{\mu\in\Lambda' \\ \mu\neq j}} |R_\mu(1)|^p}{\displaystyle\sum_{\lambda\in\Lambda'}\prod_{\substack{\mu\in\Lambda' \\ \mu\neq\lambda}} |R_\mu(1)|^p} & \text{if } |\Lambda'| \geq 2 \text{ and } j \in \Lambda' \\ 0 & \text{if } j \notin \Lambda' \end{cases}$$

(Fig. 6b).

(**ii**) If $|R_j(1)| = 0$ for some $j \in \Lambda'$, $(2-\alpha)M - T_j(1) = 0$ for these boxes, which are just points. In this case, let $\Lambda'' \equiv \Lambda' \cap \{j\colon R_j(1) = \{\mathbf{a}\}\}$ (*point box case*). Then:

$$y_j = \begin{cases} \dfrac{1}{|\Lambda''|} & \text{if } j \in \Lambda'' \\ 0 & \text{if } j \notin \Lambda'' \end{cases} \tag{13}$$

where $|\Lambda''|$ is the number of elements in the set Λ'' (Fig. 6c).

Note that Λ'' indexes only point boxes of the current input **a**. The ARTMAP-IC match tracking search rule, MT $-$, permits the creation of two or more identical boxes $R_j(1) = \{\mathbf{a}\}$. This allows a network to encode inconsistent cases, where identical training vectors are associated with different output predictions.

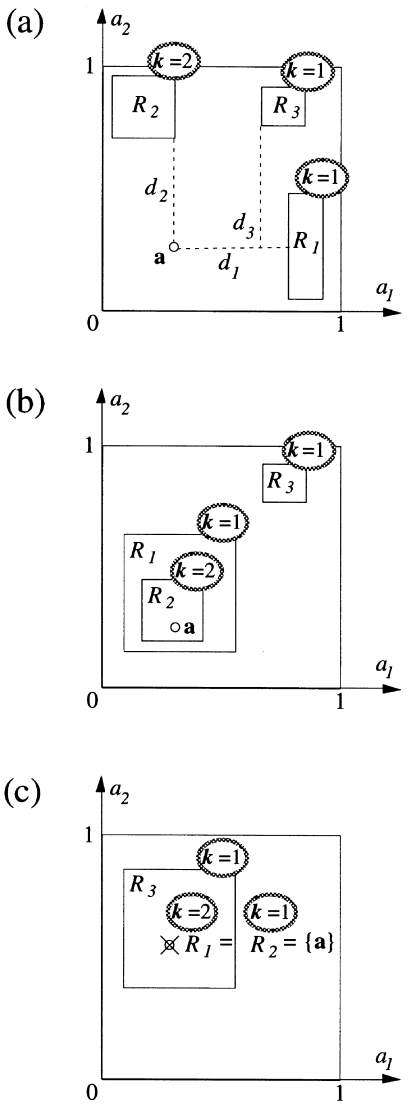

Fig. 6. Increased gradient CAM rule in the conservative limit with $p = 1$. (a) If **a** is not contained in any box $R_j(1)$, then **y** is a function of the distances from **a** to each box. In this example, $\Lambda = \{1, 2, 3\}$, and $d_1 = 0.49$, $d_2 = 0.44$ and $d_3 = 0.85$. Thus, $\mathbf{y} \cong (0.37, 0.42, 0.21)$, as in Fig. 5c. Coding converges toward WTA as **a** approaches one box $R_j(1)$, since then $d(R_j(1),\mathbf{a}) \rightarrow 0$ in Eq. (11). (b) If **a** is contained in one or more boxes $R_j(1)$, the corresponding activations dominate the stored code, in order of the box sizes. In this example, $\Lambda' = \{1, 2\}$, $|R_1(1)| = 0.96$ and $|R_2(1)| = 0.53$. Thus, $\mathbf{y} \cong (0.36, 0.64, 0)$. Coding converges toward WTA as one of these boxes shrinks toward the point box $\{\mathbf{a}\}$, since then $|R_j(1)| \rightarrow 0$ in Eq. (12). (c) If **a** is identical to one or more point boxes, the corresponding activations dominate the stored code. In this example, $\Lambda' = \{1, 2, 3\}$ and $\Lambda'' = \{1, 2\}$. Thus, $\mathbf{y} = (0.5, 0.5, 0)$, as in Eq. (13).

5. Distributed ARTMAP algorithm

In the general case, dARTMAP learns to predict an arbitrary outcome vector $\mathbf{b} = (b_1, \ldots, b_k, \ldots, b_L)$, given an input vector $\mathbf{a} = (a_1, \ldots, a_i, \ldots, a_M)$. The specific dARTMAP algorithm below considers the special case of classification problems, which set one component $b_K = 1$, placing the input **a** in the output class K. Disengaging the search process by setting $\rho \equiv 0$ converts the algorithm to a type of distributed competitive learning system. The full dARTMAP algorithm reduces to a version of ARTMAP-IC when F_2 coding remains distributed during testing, but is WTA during training. The algorithm further reduces to fuzzy ARTMAP when coding is WTA during both testing and training.

A complete dARTMAP system may be implemented as a real-time network with local computations (Fig. 2). The algorithm employs a reduced set of necessary variables (Table 1), eliminating computations that become redundant in the case of classification outputs. Table 2 lists system parameters, along with their ranges and the fixed values used in subsequent simulations. Table 3 summarizes the computational notation used in the algorithm. Each dART$_a$ input is complement coded, with $0 \leq a_i \leq 1$, so $\mathbf{I} = \mathbf{A} = (\mathbf{a}, \mathbf{a}^c)$.

5.1. Distributed ARTMAP training

During dARTMAP training, input pairs $\left(\mathbf{a}^{(1)}, \mathbf{b}^{(1)}\right)$, $\left(\mathbf{a}^{(2)}, \mathbf{b}^{(2)}\right)$, …, $\left(\mathbf{a}^{(n)}, \mathbf{b}^{(n)}\right)$, … are presented for equal time intervals. Prior to training, all LTM variables are set equal to 0 (Table 4).

Step 1—First iteration: $n = 1$

$$\text{Input vector—}A_i = \begin{cases} a_i^{(1)} & \text{if } 1 \leq i \leq M \\ 1 - a_i^{(1)} & \text{if } M+1 \leq i \leq 2M \end{cases}.$$

Output vector—K is the target output class, with $b_K^{(1)} = 1$.

Set $C = 1$, $y_1 = 1$, $Y_1 = 1$, $\sigma_i = 1$ $(i = 1, \ldots, 2M)$, and $\kappa(1) = K$.

Go to **Step 8—Resonance**

Step 2—Reset: New STM steady state at the coding fields F_2 and F_3

$F_0 \rightarrow F_2$ signal—For $j = 1, \ldots, C$:

$$\text{Phasic—}S_j = \sum_{i=1}^{2M} A_i \wedge \left(1 - \tau_{ij}\right).$$

Table 1
dARTMAP variables

STM		LTM		Signals	
x_i	F_1, matching	τ_{ij}	$F_0 \rightarrow F_2$	T_j, total (S_j, phasic	$F_0 \rightarrow F_2$ Θ_j, tonic)
y_j	F_2, coding	c_j	$F_2 \rightarrow F_3$	σ_i	$F_3 \rightarrow F_1$
Y_j	F_3, counting	τ_{ji}	$F_3 \rightarrow F_1$	σ_k	$F_3 \rightarrow F_0^{ab}$

$i = 1, \ldots, 2M$; $j = 1, \ldots, C$; $k = 1, \ldots, L$.

Table 2
Parameters

	Parameter	Range	Simulation value
F_0 input components	i	$i = 1, \ldots, 2M$	
Number of committed F_2 nodes	C		
F_2 coding nodes	j	$j = 1, \ldots, C$	
Output components	k	$k = 1, \ldots, L$	
$F_0 \rightarrow F_2$ signal to uncommitted nodes	T^u	$T_j(1)\vert_{\tau_{ij}\equiv 0}$	M
In distributed mode, the index set of F_2 nodes activated by the CAM rule	Λ	$\subseteq \{1, \ldots, C\}$	Committed nodes with $T_j(1) \geq T^u$
In WTA mode, the index of the single active node	J	$j = 1, \ldots, C$	
Correct output class	K	$k = 1, \ldots, L$	
Predicted output class	K'	$k = 1, \ldots, L$	
Association between the coding node j and the output class k	$\kappa(j) = k$	$k = 1, \ldots, L$	
Index set of F_2 nodes that are refractory	Δ	$\subseteq \{1, \ldots, C\}$	
Signal rule parameter	α	$(0, 1)$	$\alpha = 0.01$
CAM rule power	p	$(0, \infty]$	$p = 1$
Learning rate	β	$[0, 1]$	$\beta = 1$ (fast learning)
Match tracking	ϵ	$\vert\epsilon\vert$ small	$\epsilon = -0.001$ (MT $-$)
dART$_a$ baseline vigilance	$\bar{\rho}$	$[0, 1]$	$\bar{\rho} = 0$
dART$_a$ vigilance	ρ	$[\bar{\rho}, 1]$	

$$\text{Tonic—}\Theta_j = \sum_{i=1}^{2M} \tau_{ij}.$$

$$\text{Total—}T_j = \begin{cases} S_j + (1-\alpha)\Theta_j & \text{if } j \notin \Delta \\ 0 & \text{if } j \in \Delta \end{cases}.$$

The CAM rule index set $\Lambda \equiv \{j = 1, \ldots, C: T_j \geq T^u\}$

(a) If the network is in distributed mode: F_2 nodes are activated according to the increased gradient CAM rule.

F_2 activation—

(i) If $(2 - \alpha)M - T_j > 0$ for all $j \in \Lambda$, then:

$$y_j = f_j(T_1, \ldots, T_C) \equiv \begin{cases} \dfrac{1}{1 + \sum\limits_{\lambda\in\Lambda,\ \lambda\neq j} \left[\dfrac{(2-\alpha)M - T_j}{(2-\alpha)M - T_\lambda}\right]^p} & \text{if } j \in \Lambda \\ 0 & \text{if } j \notin \Lambda \end{cases}$$

(ii) Point box case: If $(2 - \alpha)M - T_j = 0$ for some $j \in \Lambda$, let $\Lambda'' = \{j \in \Lambda: (2 - \alpha)M - T_j = 0\}$. Then:

$$y_j = \begin{cases} \dfrac{1}{\vert\Lambda''\vert} & \text{if } j \in \Lambda'' \\ 0 & \text{if } j \notin \Lambda''. \end{cases}$$

$$F_3 \text{ activation—} Y_j = \frac{c_j y_j}{\sum\limits_{\lambda=1}^{C} c_\lambda y_\lambda} \quad (j = 1, \ldots, C)$$

$$F_3 \rightarrow F_1 \text{ signal—} \sigma_i = \sum_{j=1}^{C} \left[Y_j - \tau_{ji}\right]^+ \quad (i = 1, \ldots, 2M)$$

(b) If the network is in WTA mode: Only one F_2 node, with $j = J$, is activated.

(i) Committed node: If $\Lambda \neq \varnothing$, let J be the smallest index j such that $T_J = \max_{j\in\Lambda}\{T_j\}$.

Table 3
Notation

Vector norm	$\vert\mathbf{v}\vert \equiv \sum_i \vert v_i\vert$
Rectification	$[w]^+ \equiv \max\{w, 0\}$
Minimum	$a \wedge b \equiv \min\{a, b\}$
Complement	$a^c \equiv 1 - a$

Table 4
Initial values of LTM variables

$F_0 \rightarrow F_2$, threshold	$\tau_{ij} = 0$
$F_2 \rightarrow F_3$, count	$c_j = 0$
$F_3 \rightarrow F_1$, threshold	$\tau_{ji} = 0$

Uncommitted node: If $\Lambda = \varnothing$, let $J = C + 1$. Node J is then newly committed: increase C by 1, and let $\kappa(J) = K$.

(ii) F_2 and F_3 activation—

$$y_j = Y_j = \begin{cases} 1 & \text{if } j = J \\ 0 & \text{if } j \neq J \end{cases}$$

(iii) $F_3 \rightarrow F_1$ signal—

$\sigma_i = (1 - \tau_{Ji})$ $(i = 1, \ldots, 2M)$.

(iv) Add J to the refractory node index set Δ.

Step 3—Reset or prediction: Check the F_1 matching criterion

F_1 activation—$x_i = A_i \wedge \sigma_i$ $(i = 1, \ldots, 2M)$.

(a) F_1 **mismatch:** If $\frac{1}{M}\sum_{i=1}^{2M} x_i < \rho$, revert to WTA mode and go to **Step 2—Reset**.

(b) F_1 **match:** If $\frac{1}{M}\sum_{i=1}^{2M} x_i \geq \rho$, go to **Step 4—Prediction**.

Step 4—Prediction:

(a) If the network is in distributed mode, define the $F_3 \rightarrow F_0^{ab}$ signal by:

$$\sigma_k = \begin{cases} \sum_{\substack{j=1 \\ \kappa(j)=k}}^{C} Y_j & \text{if } \kappa(j) = k \text{ for some } j \\ 0 & \text{otherwise} \end{cases} \qquad (k = 1, \ldots, L)$$

Let K' be the smallest index k such that $\sigma_{K'} = \max\{\sigma_k\}$.

(b) If the network is in WTA mode, let $K' = \kappa(J)$.

Step 5—Match tracking or resonance: Check the output class prediction

(a) Incorrect prediction:
If $K' \neq K$, go to **Step 6—Match tracking**.

(b) Correct prediction:
If $K' = K$ and the network is in distributed mode, go to **Step 7—Credit assignment**.

If $K' = K$ and the network is in WTA mode, go to **Step 8—Resonance**.

Step 6—Match tracking: Raise vigilance ρ to the point of dART$_a$ reset.

Set $\rho = \frac{1}{M}\sum_{i=1}^{2M} x_i + \epsilon$.

Revert to WTA mode and go to **Step 2—Reset**.

Step 7—Credit assignment: Black out F_2 nodes that do not predict the correct output class K, via $F_0^b \rightarrow F_2$ credit assignment connections (Fig. 2). Renormalize F_2 and F_3 and recalculate the $F_3 \rightarrow F_1$ signal σ_i. For $i = 1, \ldots, 2M$ and $j = 1, \ldots, C$:

F_2 blackout—$\bar{y}_j = \begin{cases} y_j & \text{if } \kappa(j) = K \\ 0 & \text{if } \kappa(j) \neq K \end{cases}$.

F_2 activation—$y_j = \dfrac{\bar{y}_j}{\sum_{\lambda=1}^{C} \bar{y}_\lambda}$.

F_3 activation—$Y_j = \dfrac{c_j y_j}{\sum_{\lambda=1}^{C} c_\lambda y_\lambda}$.

$F_3 \rightarrow F_1$ signal—$\sigma_i = \sum_{j=1}^{C} \left[Y_j - \tau_{ji}\right]^+$.

Step 8—Resonance: For $i = 1, \ldots, 2M$ and $j = 1, \ldots, C$:

Save old values—$\tau_{ij}^{\text{old}} = \tau_{ij}$, $c_j^{\text{old}} = c_j$, $\tau_{ji}^{\text{old}} = \tau_{ji}$.

Increase $F_0 \rightarrow F_2$ threshold (distributed instar)—$\tau_{ij} = \tau_{ij}^{\text{old}} + \beta\left[y_j - \tau_{ij}^{\text{old}} - A_i\right]^+$.

Increase $F_2 \rightarrow F_3$ instance counting weights—$c_j = c_j^{\text{old}} + y_j$.

Increase $F_3 \rightarrow F_1$ threshold (distributed outstar)—

$$\tau_{ji} = \tau_{ji}^{\text{old}} + \beta \frac{\left[\sigma_i - A_i\right]^+}{\sigma_i}\left[Y_j - \tau_{ji}^{\text{old}}\right]^+$$

Reset node recovery—$\Delta = \varnothing$.

ART$_a$ vigilance recovery—$\rho = \bar{\rho}$.

Step 9—Next iteration: Increase n by 1.

New input—$A_i = \begin{cases} a_i^{(n)} & \text{if } 1 \leq i \leq M \\ 1 - a_i^{(n)} & \text{if } M + 1 \leq i \leq 2M \end{cases}$.

New output—K is the target output class, with $b_K^{(n)} = 1$.

Revert to distributed mode.

Go to **Step 2—Reset**.

5.2. Distributed ARTMAP testing

Neither search nor learning occurs during dARTMAP

testing, when the dARTMAP algorithm implicitly implements feedforward network activation. With $\bar{\rho}=0$, the F_1 matching criterion is always met, so the algorithm does not need to compute F_1 activation. Alternative testing algorithms are also possible. For example, by setting $\bar{\rho}>0$ and checking the matching criterion, a system might reject certain predictions as unreliable. Since predictions are distributed across classes, an output class decision threshold could also be selected to calibrate the false alarm rate.

Test Step 1—Test vector n:

$$\text{Input vector—}A_i=\begin{cases} a_i^{(n)} & \text{if } 1\le i\le M \\ 1-a_i^{(n)} & \text{if } M+1\le i\le 2M \end{cases}$$

Output class—K is the target output class, with $b_K^{(n)}=1$.

Test Step 2—Reset: New STM steady state at the coding fields F_2 and F_3

$F_0\rightarrow F_2$ signal—For $j=1,\ \ldots,\ C$:

$$\text{Phasic—}S_j=\sum_{i=1}^{2M}A_i\wedge\left(1-\tau_{ij}\right).$$

$$\text{Tonic—}\Theta_j=\sum\nolimits_{i=1}^{2M}\tau_{ij}.$$

$$\text{Total—}T_j=S_j+(1-\alpha)\Theta_j.$$

The CAM rule index set $\Lambda=\{j=1,\ \ldots,\ C\colon T_j\ge T^{\mathrm{u}}\}$.

F_2 activation: Increased gradient CAM rule—

(i) If $(2-\alpha)M-T_j>0$ for all $j\in\Lambda$, then:

$$y_j=f_j(T_1,\ \ldots,\ T_C)\equiv\begin{cases}\dfrac{1}{1+\displaystyle\sum_{\lambda\in\Lambda,\ \lambda\neq j}\left[\dfrac{(2-\alpha)M-T_j}{(2-\alpha)M-T_\lambda}\right]^p} & \text{if } j\in\Lambda \\ 0 & \text{if } j\notin\Lambda\end{cases}$$

(ii) Point box case: If $(2-\alpha)M-T_j=0$ for some $j\in\Lambda$, let $\Lambda''=\{j\in\Lambda\colon(2-\alpha)M-T_j=0\}$. Then:

$$y_j=\begin{cases}\dfrac{1}{|\Lambda''|} & \text{if } j\in\Lambda'' \\ 0 & \text{if } j\notin\Lambda''\end{cases}$$

$$F_3\text{ activation—}Y_j=\frac{c_jy_j}{\sum_{\lambda=1}^{C}c_\lambda y_\lambda}\ (j=1,\ \ldots,\ C).$$

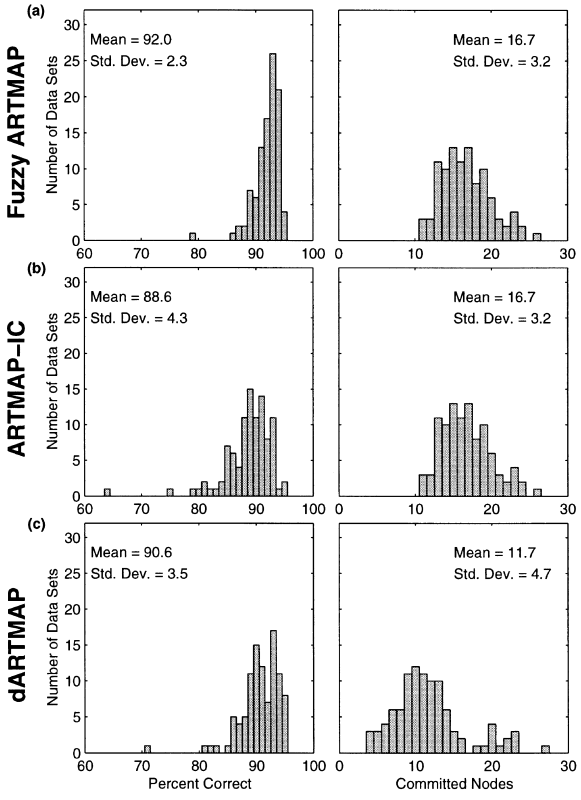

Fig. 7. Circle-in-the-square simulations. For (a) fuzzy ARTMAP, (b) ARTMAP-IC, and (c) dARTMAP, histograms show the number of data sets that produce a given test set predictive accuracy and a given number of committed nodes.

Test Step 3—Prediction:

$F_3\rightarrow F_0^{ab}$ signal—

$$\sigma_k=\begin{cases}\displaystyle\sum_{\substack{j=1\\ \kappa(j)=k}}^{C}Y_j & \text{if } \kappa(j)=k \text{ for some } j \\ 0 & \text{otherwise}\end{cases}\qquad(k=1,\ \ldots,\ L)$$

Let K' be the smallest index k such that $\sigma_{K'}=\max\{\sigma_k\}$.

Test Step 4—Evaluation:

The prediction is counted as correct if $K'=K$.

Alternatively, when $L=2$, the signal vector $\sigma=(\sigma_1,\ \sigma_2)$ may make the prediction $k=1$ when σ_1 exceeds a specified decision threshold γ; or σ_1 may be used to generate an ROC curve parameterized by γ.

6. Distributed ARTMAP simulations

The circle-in-the-square benchmark problem serves to illustrate computational properties of the dARTMAP

Table 5
Simulations with binary and Adaline output mappings

	Binary output map (% correct $\bar{x} \pm$ sd)	Adaline output map (% correct, $\bar{x} \pm$ sd)	No. of committed nodes ($\bar{x} \pm$ sd)
(a) Fuzzy ARTMAP	92.0 ± 2.3	92.5 ± 2.0	16.7 ± 3.2
(b) ARTMAP-IC	88.6 ± 4.3	93.7 ± 1.3	16.7 ± 3.2
(c) dARTMAP	90.6 ± 3.5	92.2 ± 2.4	11.7 ± 4.7

algorithm. This task requires a network to identify those points in a unit square that lie within a circle placed at the center of the square and occupying half the area. During training, 1000 randomly chosen input points $\mathbf{a} = (a_1, a_2)$ are each presented once, in an incremental learning paradigm. The output vector specifies whether a point is inside ($\mathbf{b} = (1, 0)$) or outside ($\mathbf{b} = (0, 1)$) the circle. During testing, the system makes an *in*/*out* prediction for each of 10 000 randomly chosen points in the unit square.

Fast learning ($\beta = 1$) implies that different training input presentation orders produce somewhat different results. In order to discount variations due to training set selection, each simulation is performed on 100 different input sets. Reported accuracy and network size results are averages across the 100 simulations. The same test set is used in all examples.

Each simulation in Sections 6.1–6.3 uses the parameter values listed in Table 2. In particular, the CAM rule power p is set equal to 1, thus fixing arbitrarily the degree of contrast enhancement. For this paradigm, average dARTMAP performance accuracy is slightly below that of fuzzy ARTMAP, but memory utilization is significantly less. Section 6.4 shows how validation set selection of p can boost dARTMAP performance beyond that of fuzzy ARTMAP, while further improving memory utilization. Section 6.5 includes a statistical analysis of results.

6.1. Distributed prediction

Fig. 7 shows two sets of histograms that illustrate the performance of fuzzy ARTMAP, ARTMAP-IC, and dARTMAP on the circle-in-the-square benchmark problem. The left-hand column shows how many of the 100 simulations produce a given percentage correct *in*/*out* test set prediction. The right-hand column shows the distribution of the numbers of committed nodes produced during training.

Fuzzy ARTMAP (Fig. 7a) creates an average of 16.7 recognition categories and achieves an average predictive accuracy of 92.0% on the test set. Except for the addition of instance counting, ARTMAP-IC (Fig. 7b) uses the same training regime as fuzzy ARTMAP, and so creates the same number of recognition categories. During testing, ARTMAP-IC uses the dARTMAP algorithm (Section 5.2), with an increased gradient CAM rule. Although instance counting and distributed prediction often improve fuzzy ARTMAP performance, in this case accuracy deteriorates somewhat, with the average correct prediction rate dropping to 88.6%. The dARTMAP training regime (Fig. 7c) brings performance back up somewhat, to an average of 90.6%, while reducing the average number of committed nodes by 30%, from 16.7 to 11.7. These simulations thus indicate how dARTMAP can improve the efficient use of network memory.

6.2. Post-processing with a linear map

When dARTMAP adds a new category node in WTA mode, that node becomes permanently connected to a single output class prediction. The algorithm thus does not take advantage of potentially distributed prediction at this stage, in contrast to the distributed representation of the ART_a stage. Similarly, the ARTMAP-IC output map remains binary and many-to-one after training.

One way to improve performance of a trained network is to calculate a fully connected linear mapping from ART_a to the output layer. Table 5 summarizes simulations that add such a mapping to paths from $F_3 \rightarrow F_0^{ab}$, computed via a least squares, or Adaline, method (Widrow & Hoff, 1960; Widrow & Sterns, 1985). A test set output then becomes a vector $\mathbf{YW}$, where $\mathbf{Y}$ is the distributed activation pattern generated by the test set input at F_3. The $C \times L$ matrix $\mathbf{W}$ equals $\sum_n \left(\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\right)^{-1}\mathbf{Y}^{\mathrm{T}}\mathbf{b}^{(n)}$, where $\mathbf{Y}$ is the distributed F_3 activation pattern in response to the training set input $\mathbf{a}^{(n)}$, summed across all training set pairs $(\mathbf{a}^{(n)}, \mathbf{b}^{(n)})$ as they are represented to the trained network in distributed mode. Analogous post-processing is a typical training stage of a radial basis function network (e.g., Hertz et al., 1991).

Table 6
Role of distributed learning

	% correct ($\bar{x} \pm$ sd)	No. of committed nodes ($\bar{x} \pm$ sd)
(a) Fuzzy ARTMAP	92.0 ± 2.3	16.7 ± 3.2
(b) dARTMAP without distributed learning	92.3 ± 2.8	17.6 ± 5.9
(c) dARTMAP with distributed learning	90.6 ± 2.3	11.7 ± 4.7

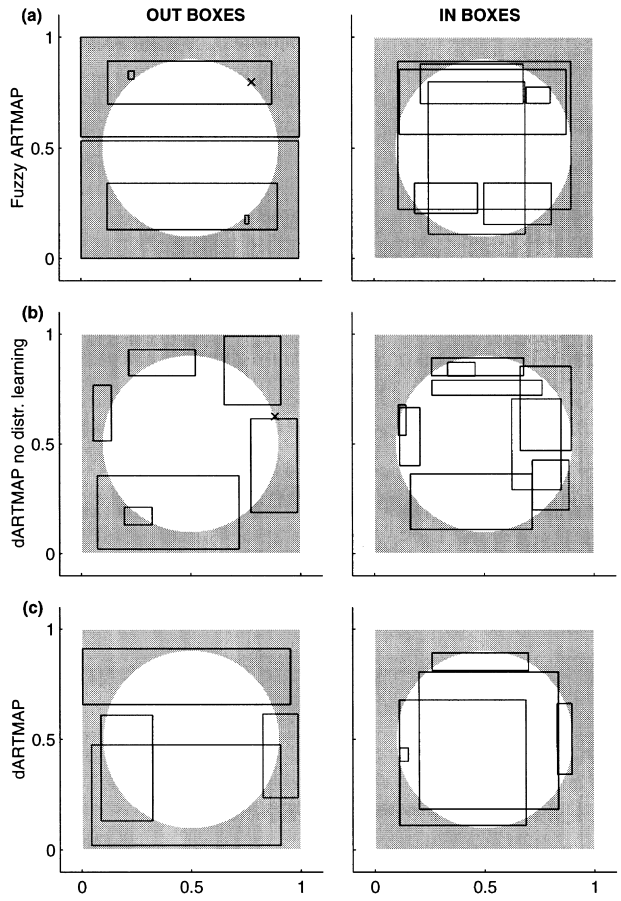

Fig. 8. Geometry of distributed learning. All systems learn the circle-in-the-square task from the same 1000-point training set. (a) Fuzzy ARTMAP uses 14 category nodes (seven *out*, seven *in*) to produce 93.3% predictive accuracy. (b) The dARTMAP alternative, without distributed learning, uses 16 coding nodes (seven *out*, nine *in*) to produce 93.2% accuracy. (c) The full dARTMAP network uses only nine coding nodes (four *out*, five *in*) to produce 92.5% predictive accuracy.

Since fuzzy ARTMAP coding remains WTA during testing as well as training, the linear mapping would not change the output map if the training set were presented for enough epochs to give 100% correct performance. For the networks trained here, in one epoch, the addition of a linear output mapping boosts performance from 92.0% to 92.5% for fuzzy ARTMAP, from 88.6% to 93.7% for ARTMAP-IC, and from 90.6% to 92.2% for dARTMAP (Table 5). The Adaline output map thus reduces error, most notably for ARTMAP-IC, without increasing the number of nodes in each network. Adaline also reduces the standard deviation in all cases, which implies that performance measures have become less dependent on input orderings. On the other hand, addition of a linear map does increase memory requirements, since the $F_3 \rightarrow F_0^{ab}$ map would then have $C \times L$ real-valued connections as opposed to the original network maps, which have only **C** binary connections. Section 6.5 includes an analysis of the statistical significance of the results shown in Table 5.

6.3. Learning vs. no learning while in distributed mode

For the first prediction for each input, dARTMAP is in distributed mode. If this prediction proves correct, then learning also takes place in the distributed mode. However, given that at least some learning also takes place in WTA mode, it is reasonable to ask to what extent learning in the distributed mode has any influence on network formation. It could be, for example, that all significant dARTMAP adaptation actually occurs in WTA mode. With this scenario, increased memory efficiency would then be ascribed to improved performance of the initial predictions, when the network is in distributed mode. Better initial predictions would diminish the number of searches, which would cause a reduction in the number of committed nodes.

To test whether most of the dARTMAP LTM changes occur in WTA mode, simulations were run with an alternative algorithm that makes initial predictions in distributed mode, but that suppresses distributed learning. For this dARTMAP alternative, if the first prediction is correct, the network updates instance counting, but otherwise experiences no learned changes. Learning is permitted only in WTA mode. Otherwise, the algorithm is identical to dARTMAP.

The behavior of the no-distributed-learning algorithm turns out to be much closer to that of fuzzy ARTMAP than to that of distributed ARTMAP. For dARTMAP (Table 6c), the average number of committed nodes is 11.7, as in Table 5c. Table 6b shows that the no-distributed-learning system produces far more nodes, an average of 17.6 with a large standard deviation. Performance measures of this network are similar to those of fuzzy ARTMAP, which produces an average of 16.7 nodes (Table 6a). In

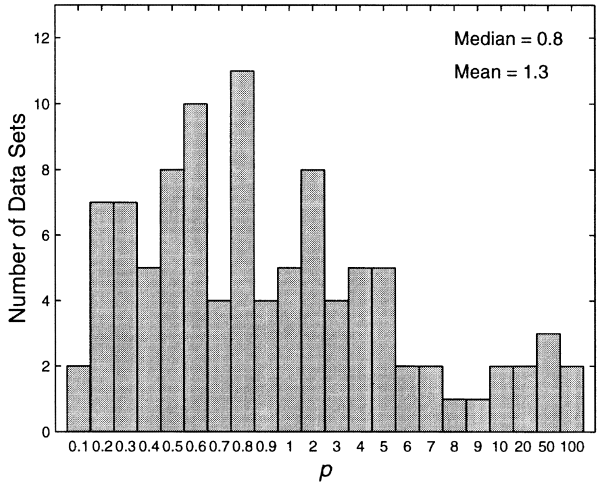

Fig. 9. Distribution of p values for 100 circle-in-the-square simulations. The CAM rule power was selected, via a validation set procedure, to maximize predictive accuracy while minimizing the number of coding nodes [Eq. (14)]. Low values of p produce highly distributed code representations, and codes become increasingly concentrated at a small number of nodes as p increases. Table 7 shows the performance improvements resulting from power selection.

Table 7
Power selection

	% correct ($\bar{x} \pm$ sd)	No. of committed nodes ($\bar{x} \pm$ sd)
(a) Fuzzy ARTMAP	92.0 ± 2.3	16.7 ± 3.2
(b) dARTMAP with power selection	92.1 ± 2.3	10.8 ± 3.4
(c) dARTMAP with power P = 1	90.6 ± 3.5	11.7 ± 4.7

particular, in the absence of distributed mode learning, all improvement in memory utilization is lost.

The geometry of coding boxes further reveals differences resulting from the learning strategies summarized in Table 6. An example that illustrates these differences trains all three systems on one of the 100 simulation input sets that contributed to the average statistics. The input set was chosen as having predictive accuracies and numbers of coding nodes that are close to each of the averages. The left-hand column of Fig. 8 shows the final boxes $R_j/R_j(1)$ that map to the prediction *out*, and the right-hand column shows the boxes that map to the prediction *in*. With fast learning, each fuzzy ARTMAP box R_j just contains all points **a** that select category j without reset. Thus, after training on 1000 randomly selected inputs, overlapping boxes tend to cover most of the unit square (Fig. 8a). The no-distributed-learning dARTMAP algorithm also learns in WTA mode, but LTM changes may occur only if the initial distributed prediction is incorrect. The resulting boxes $R_j(1)$ tend to cluster around the circle boundary (Fig. 8b), where most of the predictive errors occur. Despite these geometric differences, fuzzy ARTMAP and the no-distributed-learning system produce similar numbers of *out*/*in* boxes and nearly identical predictive accuracies for this input set. Like fuzzy ARTMAP, the full dARTMAP system produces boxes that cover most of the input space (Fig. 8c). However, distributed activation allows the network to utilize resources more efficiently, producing fewer coding nodes. Differences between Fig. 8b and c are due entirely to the absence or presence of learning in the distributed mode.

6.4. Contrast control with the increased gradient CAM rule

The basic dARTMAP algorithm admits a host of design alternatives. One such alternative varies the degree of contrast enhancement in the code by varying the power p in the increased gradient CAM rule. All previous simulations fixed $p = 1$. Different values of p alter the degree to which the stored F_2 code **y** contrast-enhances the signal T_j. As $p \to \infty$, the increased gradient rule converges to WTA coding, and codes become increasingly uniform as $p \to 0$. The simulations below show that selecting p using a validation subset of the training set improves dARTMAP performance beyond that of fuzzy ARTMAP, while further reducing the number of committed nodes.

ARTMAP and dARTMAP networks are designed to balance the twin goals of maximizing predictive accuracy and minimizing network size while carrying out fast, stable, on-line learning. In dARTMAP circle-in-the-square simulations, predictive accuracy is generally greater than 80%, with the number of coding nodes (C) less than 25 (Fig. 7). A criterion function that expresses the balance between maximal accuracy and minimal size can be expressed as:

$$criterion = \theta \frac{[percent - 80]^+}{20} + (1 - \theta) \frac{[25 - C]^+}{25} \qquad (14)$$

for $\theta \in [0, 1]$. Setting $\theta = 0$ would minimize the number of coding nodes while setting $\theta = 1$ would maximize predictive accuracy. The simulations summarized below set $\theta = 0.5$, which weighs the competing goals about equally.

For each 1000-point circle-in-the-square training set, a power p was selected by a validation procedure. For each of 22 candidate values between $p = 0.1$ and $p = 100$ (Fig. 9), 750 inputs were presented to a dARTMAP system using the given p value in the increased gradient CAM rule. The chosen power p was then the one that maximized the criterion function defined by Eq. (14) on the remaining 250 training set inputs. Before testing, dARTMAP was trained anew on all 1000 inputs using the chosen power.

Fig. 9 shows that selected p values vary across at least three orders of magnitude, depending upon the randomly chosen input sets. Despite the essential similarities among the simulations, certain input sets produce best results with highly distributed codes ($p = 0.1$), while others prefer codes that are nearly WTA ($p = 100$). Across 100 training sets, the median selected p value was 0.8 and $\exp(\text{mean}(\ln p)) = 1.3$. Therefore, the arbitrary value $p = 1$ used in previous simulations would actually be a reasonable a priori choice for this problem if a validation set selection process were not possible. Table 7 demonstrates that power selection improves dARTMAP performance even beyond that of fuzzy ARTMAP, while further decreasing the average number of coding nodes. In addition, the reduction in the standard deviation indicates that power selection produces more consistent performance with respect to the input orderings.

6.5. Statistical analysis of performance comparisons

Statistical analysis indicates the degree of significance of result differences for the systems fuzzy ARTMAP, ARTMAP-IC, and dARTMAP and their Adaline modifications, summarized in Table 5, as follows.

6.5.1. Procedure

Each system was trained on the same 100 data sets. System performance on a single, common test set was measured by two variables, percentage of correct responses and number of committed nodes.

6.5.2. Analysis

Repeated measures analysis of variance was the natural design to use, since the same subjects (100 data sets) were compared under different treatments (types of network algorithms) (Devore, 1995). The analysis performed included the following components:

- the percentage of correct responses of dARTMAP, fuzzy ARTMAP, and ARTMAP-IC (binary output map) were subjected to repeated measures analysis of variance with multiple dependent paired *t*-tests (employing the Bonferroni inequality) used as a post hoc procedure (Stevens, 1996);
- the same approach was used for the percent correct responses of the networks with the Adaline modification;
- paired *t*-tests were used to determine the effect of the Adaline modification for each network (i.e., dARTMAP vs. dARTMAP with Adaline, fuzzy ARTMAP vs. fuzzy ARTMAP with Adaline, and ARTMAP-IC vs. ARTMAP-IC with Adaline);
- a paired *t*-test was employed to compare the numbers of node results of dARTMAP vs. fuzzy ARTMAP, the latter being identical to ARTMAP-IC for this measure.

6.5.3. Percentage of correct responses

In the repeated measures analysis, both univariate and multivariate approaches were considered. Generally, the univariate approach is believed to be more powerful, provided that the sphericity assumption is not violated. On the other hand, the multivariate approach has certain advantages when the number of subjects is high (Stevens, 1996).

6.5.3.1. Binary output algorithms (distributed ARTMAP, fuzzy ARTMAP, and ARTMAP-IC). Greenhouse–Geisser $\epsilon = 0.95531$ (sphericity holds); univariate repeated measures test (with ϵ adjusted *df*): $F(1.91, 189.15) = 36.09$, $p < 0.001$; multivariate repeated measures test (Hotellings): $F(2, 98) = 29.72$, $p < 0.001$. This result indicates that the percentage of correct responses differs between algorithms to a statistically significant degree. To determine where the difference(s) exist(s), multiple dependent paired *t*-tests were employed. The performance of binary output dARTMAP was significantly worse than that of binary output fuzzy ARTMAP ($t = -3.79$, $df = 99$, $p < 0.001$). Analogously, binary output ARTMAP-IC was outperformed by both binary output dARTMAP ($t = -5.03$, $df = 99$, $p < 0.001$) and binary output fuzzy ARTMAP ($t = -7.75$, $df = 99$, $p < 0.001$).

6.5.3.2. Adaline output algorithms (distributed ARTMAP, fuzzy ARTMAP, and ARTMAP-IC). Greenhouse–Geisser $\epsilon = 0.83647$ (sphericity cannot be assumed); univariate repeated measures test (with ϵ adjusted *df*): $F(1.67, 165.62) = 23.92$, $p < 0.001$; multivariate repeated measures test (Hotellings): $F(2, 98) = 35.17$, $p < 0.001$. This result indicates that the percentage of correct responses differed within the group of networks with Adaline modification. dARTMAP with Adaline and fuzzy ARTMAP with Adaline did not differ significantly ($t = -1.1$, $df = 99$, $p = 0.27$). ARTMAP-IC with Adaline significantly outperformed both dARTMAP with Adaline ($t = 6.22$, $df = 99$, $p < 0.001$) and fuzzy ARTMAP with Adaline ($t = 6.92$, $df = 99$, $p < 0.001$).

6.5.3.3. Adaline effect. For each of the networks, the effect of the Adaline modification produced a significant improvement in the percentage of correct responses: dARTMAP vs. dARTMAP with Adaline ($t = -5.44$, $df = 99$, $p < 0.001$), fuzzy ARTMAP vs. fuzzy ARTMAP with Adaline ($t = -5.25$, $df = 99$, $p < 0.001$), and ARTMAP-IC vs. ARTMAP-IC with Adaline ($t = -12.38$, $df = 99$, $p < 0.001$).

6.5.4. Number of category nodes

A paired *t*-test determined that the numbers of category nodes for dARTMAP were significantly smaller than those for fuzzy ARTMAP/ARTMAP-IC ($t = -12.54$, $df = 99$, $p < 0.001$).

6.5.5. Summary

The results from the statistical analysis indicate that, for the binary output algorithms, fuzzy ARTMAP performs better than dARTMAP, which in turn performs better than ARTMAP-IC. However, dARTMAP requires significantly fewer nodes than fuzzy ARTMAP and ARTMAP-IC. The Adaline modification significantly boosted the performance of all networks, making dARTMAP results comparable to those of fuzzy ARTMAP, with ARTMAP-IC outperforming both. Once again, however, the memory efficiency of dARTMAP was significantly higher.

7. Circle-in-the-square geometry

Early in the learning process, both fuzzy ARTMAP and distributed ARTMAP typically establish a few categories that form the basis for subsequent coding. With a fast learning paradigm, the structure of this code foundation is highly dependent on the location of the first several input points. New committed nodes are added whenever the current network is unable to support the correct prediction. The circle-in-the-square example, with its two-dimensional inputs **a**, permits a graphical representation of this incremental learning process, as follows.

Fig. 10 shows the three coding boxes $R_j(1)$ learned by a dARTMAP network in response to an initial sequence of eight inputs, with fast learning in the conservative limit. These inputs were hand-picked to illustrate possible stages of distributed-mode learning. A more typical input sequence would produce a preponderance of early WTA learning. Subsequent input points were chosen at random (Fig. 11).

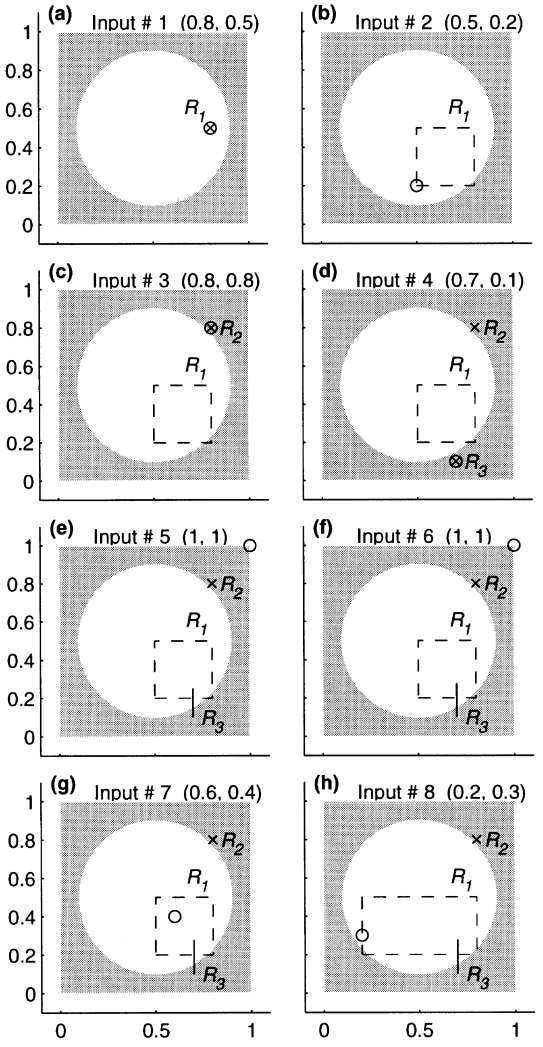

Fig. 10. Distributed ARTMAP circle-in-the-square simulations. Each square shows the coding box $R_j(1)$ after learning. During an initial series of eight input presentations, dARTMAP creates three coding boxes. Node $j = 1$ maps to the outcome *in* and nodes $j = 2$ and $j = 3$ map to the outcome *out*.

Each newly committed node begins as a point box (Fig. 10a, c, d), which may then grow during subsequent learning. Upon commitment, each point box becomes permanently mapped to an output prediction: the F_2 node $j = 1$ maps to $k = 1$ (*in*) and $j = 2$ and $j = 3$ map to $k = 2$ (*out*). If only one active node j makes the correct prediction, then the box $R_j(1)$ grows just enough to include the current input **a**. In Fig. 10b, $R_1(1)$ grows to include the input point (0.5, 0.2), and later grows further to include (0.2, 0.3) (Fig. 10h). In Fig. 10g, node $j = 1$ is also the only one that remains active after credit assignment, but the input point (0.6, 0.4) is, by then, already contained in $R_1(1)$, so no further expansion occurs.

If the network in distributed mode makes the correct prediction, then one or more of the boxes associated with the correct output class K may grow toward **a**. When the point $\mathbf{a} = (1, 1)$ is presented (Fig. 10e), the initial distances from **a** to the three boxes are: $d(R_1(1), \mathbf{a}) = 0.7$, $d(R_2(1), \mathbf{a}) = 0.4$, and $d(R_3(1), \mathbf{a}) = 1.2$. Thus, by the geometric version of the increased gradient CAM rule with $p = 1$ [Eq. (11)], F_2 steady-state activation is $\mathbf{y} \cong (12/40, 21/40, 7/40)$. As a result of the first four inputs, the counting weights begin as: $c_1 = 2$, $c_2 = 1$ and $c_3 = 1$. Thus, F_3 activation is $\mathbf{Y} \cong (24/52, 21/52, 7/52)$. The distributed prediction sets $\sigma_1 \cong 24/52 \cong 0.46$ and $\sigma_2 \cong 28/52 \cong 0.54$, so the output class prediction is $K' = 2$ (*out*), which is correct, despite the fact that the maximally active F_3 node ($j = 1$) would have predicted *in*. Before distributed learning can occur, credit assignment sets to 0 the activations of F_2 nodes that map to the incorrect outcome. After renormalization, then, $\mathbf{y} = \mathbf{Y} \cong (0.0, 0.75, 0.25)$.

How much a box $R_j(1)$ will expand during learning depends on the box $R_j(y_j)$ (Fig. 5b). Before learning in Fig. 10e, $R_2(1)$ is the point box $\{(0.8, 0.8)\}$, and $R_2(y_2)$ is the box with lower left-hand corner (0.55, 0.55) and upper right-hand corner (1, 1) (Fig. 5a). Thus, $R_2(y_2)$ initially contains the input $\mathbf{a} = (1, 1)$, so neither $R_2(y_2)$ nor $R_2(1)$ expands during learning. The boxes $R_3(y_3)$ and $R_3(1)$ do expand, however. Before learning, $R_3(1)$ is the point box $\{(0.7, 0.1)\}$ (Fig. 10d) and $R_3(y_3)$ is the set of points **q** in the unit square with $q_2 \leq 0.85$. The upper edge of box $R_3(y_3)$ is therefore a distance of 0.15 units away from the input $\mathbf{a} = (1, 1)$. During learning, this box expands to include **a**, as the threshold τ_{43} increases by 0.15 units, from 0.1 to 0.25. After learning, the box $R_3(1)$ has expanded to become the set of points **q** with $q_1 = 0.7$ and $0.1 \leq q_2 \leq 0.25$ (Fig. 10e). The counting weights are then $c_1 = 2$, $c_2 = 1.75$ and $c_3 = 1.25$.

Note that the pattern of learning in Fig. 10e would be altered for values of the CAM rule power different from $p = 1$. If p were large enough to make $y_2 > 0.8$ following credit assignment, then $R_2(y_2)$ would not initially contain the input $\mathbf{a} = (1, 1)$, so $R_2(y_2)$ and $R_2(1)$ would expand during learning. Further, if p were large enough to make $y_3 \leq 0.1$ following credit assignment, then $R_3(y_3)$ would initially contain **a** and so neither $R_3(y_3)$ nor $R_3(1)$ would expand during learning.

Additional learning may occur even if an input repeats the previous one. In both Fig. 10e and f, $\mathbf{a} = (1, 1)$. When this point is presented the second time, $d(R_1(1), \mathbf{a}) = 0.7$, $d(R_2(1), \mathbf{a}) = 0.4$, and $d(R_3(1), \mathbf{a}) = 1.05$, so $\mathbf{y} \cong (0.29, 0.51, 0.20)$ and $\mathbf{Y} \cong (0.34, 0.52, 0.14)$. The distributed output prediction sets $\sigma_1 \cong 0.34$ and $\sigma_2 \cong 0.66$. Note that, due to learning in the previous interval, the network prediction of $K' = 2$ (*out*) is now much stronger than it was when $\mathbf{a} = (1, 1)$ was first presented. After credit assignment, $\mathbf{y} \cong (0.0, 0.72, 0.28)$. During learning, $R_2(y_2)$ and $R_2(1)$ again do not grow. The box $R_3(y_3)$ begins as the set of points **q** in the unit square with $q_2 \leq 0.97$, so the upper edge is a distance of 0.03 units away from the input $\mathbf{a} = (1, 1)$. During learning, this box expands to include **a**, as the threshold τ_{43} increases by 0.03 units, from 0.25 to 0.28. After learning, the box $R_3(1)$ has expanded slightly, to become the set of points **q** with $q_1 = 0.7$ and $0.1 \leq q_2 \leq 0.28$ (Fig. 10f).

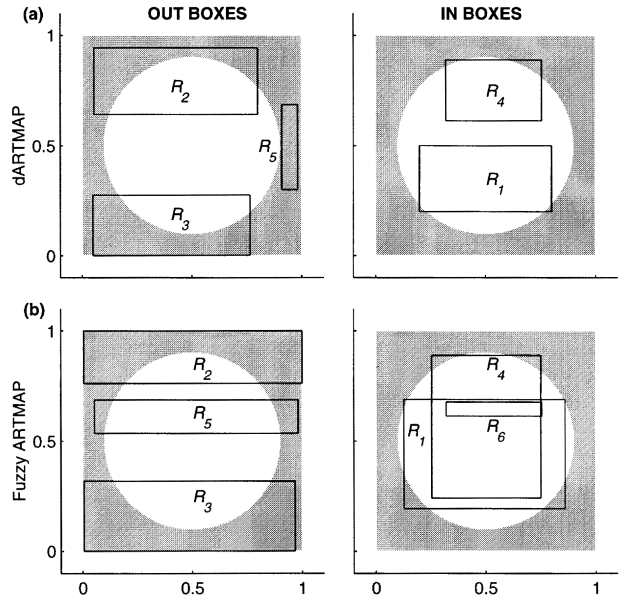

Fig. 11. (a) Following the sequence of points illustrated in Fig. 10, presenting 92 more inputs to the dARTMAP system creates a total of five coding boxes $R_j(1)$, three mapped to *out* and two mapped to *in*. (b) For the same 100 inputs, fuzzy ARTMAP creates six category boxes R_j, three mapped to *out* and three mapped to *in*.

Fig. 11a shows the dARTMAP boxes $R_j(1)$ that map to the predictions *out* and *in* after 100 input points have been presented, starting with the sequence illustrated in Fig. 10. Due to presentation of inputs 9–100, the *out* boxes $R_2(1)$ and $R_3(1)$ have expanded to the left and a third *out* box $R_5(1)$ has appeared at the right. The *in* box $R_1(1)$ has remained almost unchanged and a second *in* box $R_4(1)$ has appeared above it.

Early in the training process, dARTMAP and fuzzy ARTMAP usually commit similar numbers of nodes. In this example, dARTMAP has three *out* boxes and two *in* boxes, while fuzzy ARTMAP has three *out* boxes and three *in* boxes after 100 inputs (Fig. 11b). However, the two systems learn different codes for the same input sequence. Differences in the coding structure are traced to dARTMAP distributed activation and learning.

8. Distributed ARTMAP variations

In addition to contrast control of the F_2 code (Section 6.4), many other dARTMAP design variations are possible. Network parameters may be modified, for example. All reported simulations set the rate parameter $\beta = 1$, for fast learning, and the baseline vigilance parameter $\bar{\rho} = 0$, to minimize network size. Choosing $\beta < 1$ permits slow learning and choosing $\bar{\rho} > 0$ rejects poorly matched codes. Other variations include alternative rules defining the $F_0 \rightarrow F_2$ signal T_j, CAM steady-state activation at F_2, and $F_2 \rightarrow F_3$ instance counting. For example, a *Weber law rule* would define the $F_0 \rightarrow F_2$ signal by:

$$T_j(y_j) = \frac{S_j(y_j)}{\alpha + 2My_j - \Theta(y_j)}. \tag{15}$$

Alternative CAM rules could make F_2 activation y_j proportional to $T_j(1)$ or to a power of $T_j(1)$, for j in a defined index set Λ. Instance counting could be nonlinear, which is useful in preventing highly active nodes from overwhelming all other predictions. Alternatively, instance counting could be suppressed by setting all counting weights equal to 1. Also, pruning algorithms may reduce the size of a trained network.

The computational demands of targeted application domains, as well as limitations of the dARTMAP algorithm defined in Section 5, pose challenges that suggest design alternatives more radical than variations. One open problem is how to generalize the network to learn arbitrary mappings, not just those where the outputs are categorical. The alternating use of WTA and distributed coding leaves open the question of what capabilities can be realized in a network that is fully distributed all the time. Another question is whether this type of distributed learning requires credit assignment. Also, the present dARTMAP algorithm, which places a premium on code stability, may lock in codes prematurely. In particular, with fast learning and a noisy input environment, dARTMAP solves the category proliferation problem of fuzzy ARTMAP, but accuracy may suffer. Alternative learning laws or other changes in the training regime could make the system more flexible in response to later training inputs while preserving fundamentally desirable stability properties. The distributed ARTMAP algorithm presented here is thus but one member of a family of possible systems that seek to combine distributed coding with stable fast learning.

Acknowledgements

This research was supported in part by the National Science Foundation (NSF IRI-94-01659) and the Office of Naval Research (ONR N00014-95-1-0409 and ONR N00014-95-0657).

References

Booker L. B., Goldberg D. E., & Holland J. H. (1989). Classifier systems and genetic algorithms. *Artificial Intelligence*, *40*, 235–282.

Carpenter G. A. (1994). A distributed outstar network for spatial pattern learning. *Neural Networks*, *7*, 159–168.

Carpenter, G. A. (1996). Distributed ART networks for learning, recognition, and prediction. *Proceedings of the World Congress on Neural Networks (WCNN'96)* (pp. 333–344).

Carpenter G. A. (1997). Distributed learning, recognition, and prediction by ART and ARTMAP neural networks. *Neural Networks*, *10*, 1473–1494.

Carpenter, G. A., & Gjaja, M. N. (1994). Fuzzy ART choice functions. *Proceedings of the World Congress on Neural Networks (WCNN'94)*, IV (pp. 133–142).

Carpenter G. A., & Grossberg S. (1987). A massively parallel architecture

for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, *37*, 54–115.

Carpenter G. A., & Markuzon N. (1998). ARTMAP-IC and medical diagnosis: instance counting and inconsistent cases. *Neural Networks*, *11*.

Carpenter G. A., & Ross W. D. (1995). ART-EMAP: a neural network architecture for object recognition by evidence accumulation. *IEEE Transactions on Neural Networks*, *6*, 805–818.

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

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

Carpenter G. A., Grossberg S., Markuzon N., Reynolds J. H., & Rosen D. B. (1992). Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, *3*, 698–713.

Caudell T. P., Smith S. D. G., Escobedo R., & Anderson M. (1994). NIRS: large scale ART-1 neural architectures for engineering design retrieval. *Neural Networks*, *7*, 1339–1350.

Devore, J. L. (1995). *Probability and Statistics for Engineering and the Sciences*, 4th ed. Belmont: Duxbury Press.

Ellias S. A., & Grossberg S. (1975). Pattern formation, contrast control, and oscillations in the short term memory of shunting on-center off-surround networks. *Biological Cybernetics*, *20*, 69–98.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In D. S. Touretzky (Ed.), *Neural Information Processing Systems* 2 (pp. 524–532). San Mateo, CA: Morgan Kaufmann.

Grossberg S. (1968). A prediction theory for some nonlinear functional-differential equations. I: Learning of lists. *Journal of Mathematical Analysis and Applications*, *21*, 643–694.

Grossberg S. (1970). Some networks that can learn, remember, and reproduce any number of complicated space-time patterns. *Studies in Applied Mathematics*, *49*, 135–166.

Grossberg S. (1972). Neural expectation: cerebellar and retinal analogs of cells fired by learnable or unlearned pattern classes. *Kybernetik*, *10*, 49–57.

Grossberg S. (1973). Contour enhancement, short term memory, and constancies in reverberating neural networks. *Studies in Applied Mathematics*, *LII*, 213–257.

Grossberg S. (1976). Adaptive pattern classification and universal recoding. II: Feedback, expectation, olfaction, and illusions. *Biological Cybernetics*, *23*, 187–202.

Grossberg S. (1980). How does a brain build a cognitive code?. *Psychological Review*, *87*, 1–51.

Hertz, J., Krogh, A., & Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*. Reading, MA: Addison-Wesley.

McCulloch W. S., & Pitts W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *9*, 127–147.

Miller E. K., Li L., & Desimone R. (1991). A neural mechanism for working and recognition memory in inferior temporal cortex. *Science*, *254*, 1377–1379.

Moody J., & Darken C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, *1*, 281–294.

Rosenblatt F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*, 386–408.

Rosenblatt, F. (1962). *Principles of Neurodynamics*. Washington, DC: Spartan Books.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructures of Cognitions*, I (pp. 318–362). Cambridge, MA: MIT Press.

Stevens, J. (1996). *Applied Multivariate Statistics for the Social Sciences*, 3rd ed. Mahwah, NJ: Lawrence Erlbaum Associates.

Werbos, P. J. (1974). Beyond regression: new tools for prediction and analysis in the behavioral sciences. Unpublished PhD. thesis, Harvard University, Cambridge, MA.

Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits. *1960 IRE WESCON Convention Record* (pp. 96–104). New York: IRE.

Widrow, B., & Sterns, S. D. (1985). *Adaptive Signal Processing*. New York: Prentice-Hall.

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