Biased ART: A neural architecture that shifts attention toward previously disregarded features following an incorrect prediction

Gail A. Carpenter, Sai Chaitanya Gaddam

Department of Cognitive and Neural Systems
Boston University
677 Beacon Street
Boston, Massachusetts 02215 USA

Neural Networks, in press.

Technical Report CAS/CNS TR-2009-003
Boston, MA: Boston University

Acknowledgements: This research was supported by the SyNAPSE program of the Defense Advanced Projects Research Agency (Hewlett-Packard Company, subcontract under DARPA prime contract HR0011-09-3-0001 and HRL Laboratories LLC, subcontract #801881-BS under DARPA prime contract HR0011-09-C-0001) and by the Science of Learning Centers program of the National Science Foundation (NSF SBE-0354378).

Biased ART: A neural architecture that shifts attention toward previously disregarded features following an incorrect prediction

Abstract

Memories in Adaptive Resonance Theory (ART) networks are based on matched patterns that focus attention on those portions of bottom-up inputs that match active top-down expectations. While this learning strategy has proved successful for both brain models and applications, computational examples show that attention to early critical features may later distort memory representations during online fast learning. For supervised learning, biased ARTMAP (bARTMAP) solves the problem of over-emphasis on early critical features by directing attention away from previously attended features after the system makes a predictive error. Small-scale, hand-computed analog and binary examples illustrate key model dynamics. Two-dimensional simulation examples demonstrate the evolution of bARTMAP memories as they are learned online. Benchmark simulations show that featural biasing also improves performance on large-scale examples. One example, which predicts movie genres and is based, in part, on the Netflix Prize database, was developed for this project. Both first principles and consistent performance improvements on all simulation studies suggest that featural biasing should be incorporated by default in all ARTMAP systems. Benchmark datasets and bARTMAP code are available from the CNS Technology Lab Website: http://techlab.bu.edu/bART/.

Keywords
Adaptive Resonance Theory, ART, ARTMAP, featural biasing, supervised learning, top-down / bottom-up interactions

Biased ART: A neural architecture that shifts attention toward previously disregarded features following an incorrect prediction

1. Introduction

During learning, Adaptive Resonance Theory (ART) models encode attended featural subsets called critical feature patterns. With winner-take-all coding, when a novel exemplar activates an established category only the features of the bottom-up input that are also in the top-down critical feature pattern remain active in working memory. The network hereby focuses attention on a subset of the input, ignoring other incoming features as not relevant to the currently active category. If the top-down / bottom-up pattern meets a matching criterion, the learned critical feature pattern sharpens, shedding features not represented in the current input.

The strategy of learning attended critical feature patterns, rather than basing memories on whole bottom-up inputs, has proved successful both in models of cognitive information processing and in applications of unsupervised ART and supervised ARTMAP systems. However, focusing on features that that were critical early in learning may lead a system later to pay too much attention to these features. Computational examples show that, for certain input sequences, such undue featural attention can distort system memories and reduce test accuracy. If training inputs are repeatedly presented, an ARTMAP system will correct these errors – but real-time learning may not afford such repeat opportunities before action is required.

Biased ARTMAP (bARTMAP) solves the problem of over-emphasis on early critical features by directing attention away from previously attended features after the system makes a predictive error. A variety of examples demonstrate that bARTMAP performance is consistently better than that of fuzzy ARTMAP. Small-scale, hand-computed analog and binary examples illustrate key model dynamics. Two-dimensional simulation examples demonstrate the evolution of bARTMAP memories as they are learned online. Benchmark simulations show that featural biasing also improves performance on large-scale examples. The Boston remote sensing image example (Carpenter, Martens, & Ogas, 2005) has been used in previous studies. A second example, which predicts movie genres and is based, in part, on the Netflix Prize database, was developed for this project. Both benchmark datasets and biased ARTMAP code are available from the CNS Technology Lab Website (http://techlab.bu.edu/bART).

For a given training input, biased ARTMAP tracks attended features that have led to predictive errors, and reduces activation of these features during search. Bias strength is controlled by a free parameter λ, with the network reducing to the unbiased system (fuzzy ARTMAP) when $\lambda=0$. For a given application, an optimal value of λ can be determined by validation, but setting λ equal to a default value of 10 produces near-optimal results on small-scale and large-scale computational examples. Improvements in test accuracy are accompanied by reduced overlap of the category boxes that geometrically represent network memories, with little or no increase in network size. All examples use the same default ARTMAP parameters, with winner-take-all coding, fast learning, and maximum generalization. In a fast-learning system, long-term memory variables reach their asymptotes on each input trial.

2. ART and ARTMAP

ART neural networks model real-time prediction, search, learning, and recognition. ART networks serve both as models of human cognitive information processing (Grossberg, 1999, 2003; Carpenter, 1997) and as neural systems for technology transfer (Caudell et al., 1994; Lisboa, 2001; Parsons & Carpenter, 2003).

Design principles derived from scientific analyses and design constraints imposed by targeted applications have jointly guided the development of many variants of the basic networks, including fuzzy ARTMAP (Carpenter et al., 1992), ARTMAP-IC (Carpenter & Markuzon, 1998), and Gaussian ARTMAP (Williamson, 1996). One distinguishing characteristic of different ARTMAP models is the nature of their internal code representations. Early ARTMAP systems, including fuzzy ARTMAP, employ winner-take-all coding, whereby each input activates a single category node during both training and testing. When a node is first activated during training, it is permanently mapped to its designated output class.

Starting with ART-EMAP (Carpenter & Ross, 1995), ARTMAP systems have used distributed coding during testing, which typically improves predictive accuracy while avoiding the design challenges inherent in the use of distributed code representations during training. In order to address these challenges, distributed ARTMAP (Carpenter, 1997; Carpenter, Milenova, & Noeske, 1998) introduced a new network configuration, new learning laws, and even a new unit of long-term memory, replacing traditional weights with adaptive thresholds (Carpenter, 1994).

Comparative analysis of the performance of ARTMAP systems on a variety of benchmark problems has led to the identification of a *default ARTMAP* network (Carpenter, 2003), which features simplicity of design and robust performance in many application domains. Default ARTMAP employs winner-take-all coding during training and distributed coding during testing within a distributed ARTMAP network configuration. With winner-take-all coding during testing, default ARTMAP reduces to the version of fuzzy ARTMAP that is used here as the basis of comparison with biased ARTMAP. However, the biased ARTMAP mechanism is a small modular addition to the ART orienting subsystem, and could be readily added to any other version of the network.

2.1. Complement coding: Learning both absent and present features

ART and ARTMAP employ a preprocessing step called *complement coding* (Figure 1), which models the nervous system's ubiquitous computational design known as *opponent processing* (Hurvich & Jameson, 1957). Balancing an entity against its opponent, as in agonist-antagonist muscle pairs, allows a system to act upon relative quantities, even as absolute magnitudes may vary unpredictably. In ART systems, complement coding (Carpenter, Grossberg, & Rosen, 1991) is analogous to retinal ON-cells and OFF-cells (Schiller, 1982). When the learning system is presented with a set of feature values $\mathbf{a} \equiv (a_1 \ldots a_i \ldots a_M)$, complement coding doubles the number of input components, presenting to the network both the original feature vector $\mathbf{a}$ and its complement $\mathbf{a}^c$.

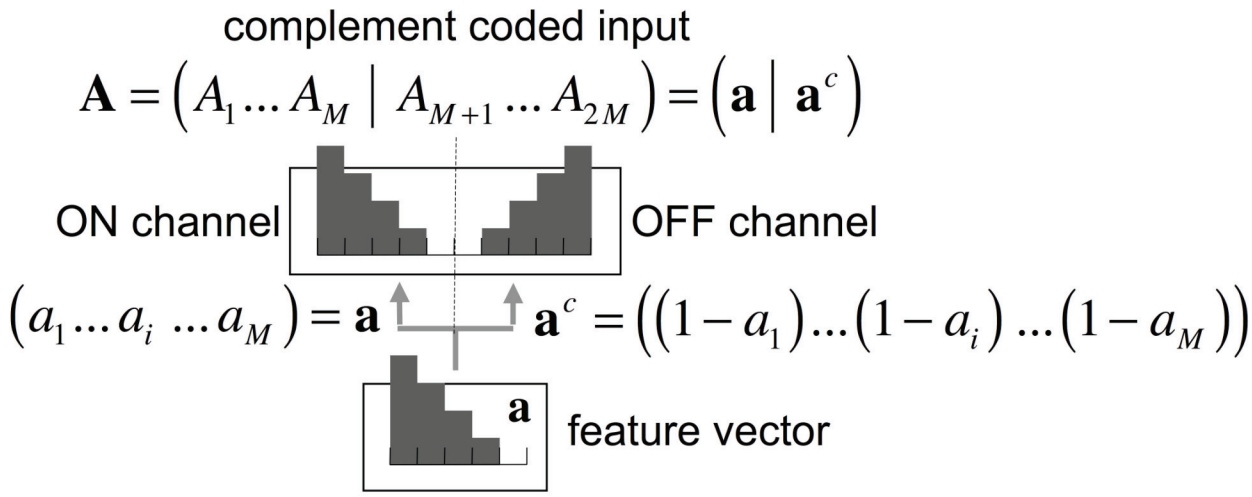

Figure 1. Complement coding transforms an *M*-dimensional feature vector **a** into a 2*M*-dimensional system input vector **A**. A complement-coded input represents both the degree to which a feature *i* is present $\left(a_i\right)$ and the degree to which that feature is absent $\left(1-a_i\right)$.

Complement coding allows an ART system to encode within its critical feature patterns of memory features that are consistently *absent* on an equal basis with features that are consistently *present*. Features that are sometimes absent and sometimes present when a given category is learning becomes uninformative with respect to that category. Since its introduction, complement coding has been a standard element of ART and ARTMAP networks, where it plays multiple computational roles, including input normalization. However, this device is not particular to ART, and could, in principle, be used to preprocess the inputs to any type of system.

To implement complement coding, component activities a_i of a feature vector **a** are scaled so that $0 \le a_i \le 1$. For each feature *i*, the ON activity a_i determines the complementary OFF activity $\left(1-a_i\right)$. Both a_i and $\left(1-a_i\right)$ are represented in the 2*M*-dimensional system input vector $\mathbf{A} = \left(\mathbf{a} \mid \mathbf{a}^c\right)$ (Figure 1). Subsequent network computations operate in this 2*M*-dimensional input space. In particular, learned weight vectors $\mathbf{w}_J$ are 2*M*-dimensional.

2.2. ARTMAP search and match tracking

The ART matching process triggers either learning or a parallel memory search (Figure 2). When search ends, the learned memory may either remain the same or incorporate new information from matched portions of the current input. While this dynamic applies to arbitrarily distributed activation patterns at the coding field F_2, the code will here be described as a single active category node J in a winner-take-all system.

Before ARTMAP makes an output class prediction, the bottom-up input **A** is matched against the top-down learned expectation, or critical feature pattern, that is read out by the active node (Figure 2b). The matching criterion is set by a parameter ρ called *vigilance*. Low vigilance permits the learning of abstract prototype-like patterns, while high vigilance requires the learning of specific exemplar-like patterns. When a new input arrives, vigilance equals a baseline level $\overline{\rho}$. Baseline vigilance is set equal to zero by default in order to maximize generalization. Vigilance rises after the system has made a predictive error. The internal control process that determines how far ρ must rise in order to correct the error is called *match tracking* (Carpenter, Grossberg, & Reynolds, 1991). As vigilance rises, the network is required to pay more attention to how well top-down expectations match the current bottom-up input.

Match tracking (Figure 3) forces an ARTMAP system not only to reset its mistakes but to learn from them. With match tracking, fast learning, and winner-take-all coding, each ARTMAP network passes the Next Input Test, which requires that, if a training input were re-presented immediately after a learning trial, it would directly activate the correct output class, with no predictive errors or search. Match tracking simultaneously implements the design goals of maximizing generalization and minimizing predictive error without requiring the choice of a fixed matching criterion. ARTMAP memories thereby include both broad and specific pattern classes, with the latter typically formed as exceptions to the more general "rules" defined by the former. ARTMAP learning produces a wide variety of such mixtures, whose exact composition depends upon the order of training exemplar presentation.

Unless they have already learned on all their coding nodes, ARTMAP systems contain a reserve of nodes that have never been activated, with weights at their initial values. These *uncommitted* nodes compete with the previously active *committed* nodes, and an uncommitted node will be chosen over poorly matched committed nodes. An ARTMAP design constraint specifies that an active uncommitted node should not reset itself. Weights initially begin with $w_{iJ} = 1$. Thus, when the active node J is uncommitted, $\mathbf{x} = \mathbf{A} \wedge \mathbf{w}_J = \mathbf{A}$ at the match field. Then, $\rho|\mathbf{A}| - |\mathbf{x}| = \rho|\mathbf{A}| - |\mathbf{A}| = (\rho - 1)|\mathbf{A}|$. Thus $\rho|\mathbf{A}| - |\mathbf{x}| \le 0$ and an uncommitted node does not trigger a reset, provided that $\rho \le 1$. For biased ARTMAP, where **A** and/or **x** are replaced by their biased counterparts in the match/mismatch decision, the requirement that an uncommitted node does not trigger a reset provides a key system design constraint.

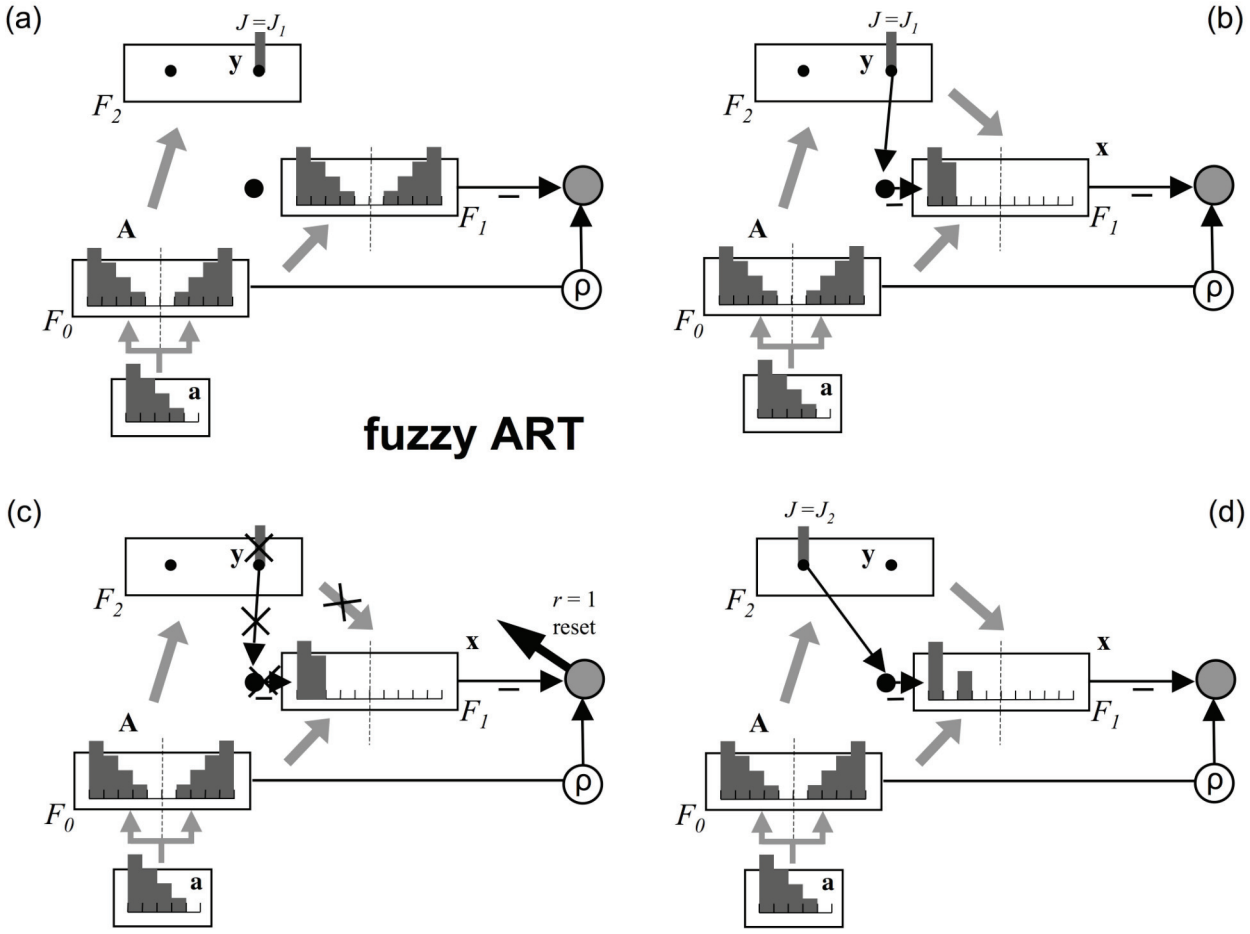

Figure 2. A fuzzy ART search cycle (Carpenter, Grossberg & Rosen, 1991), with winner-take-all coding in a distributed ART network configuration (Carpenter, 1997). The ART 1 search cycle (Carpenter & Grossberg, 1987) is the same, but allows only binary inputs and did not originally feature complement coding. When an F_2 node J is active, the field F_1 represents the matched activation pattern $\mathbf{x} = \mathbf{A} \wedge \mathbf{w}_J$, where $\wedge$ denotes the component-wise minimum, or fuzzy intersection, of the bottom-up input $\mathbf{A}$ and the top-down expectation $\mathbf{w}_J$. If the matched pattern fails to meet the matching criterion, then the active node is reset at F_2, and the system searches for another node that better represents the input. The match / mismatch decision is made in the ART orienting system. Each active feature in the input pattern $\mathbf{A}$ excites the orienting system with gain equal to the vigilance parameter ρ. Hence, with complement coding, the total excitatory input is $\rho|\mathbf{A}| = \rho\sum_{i=1}^{2M} A_i = \rho\sum_{i=1}^{M}\left(a_i + \left(1-a_i\right)\right) = \rho M$. Active cells in the matched pattern $\mathbf{x}$ inhibit the orienting system, leading to a total inhibitory input equal to $-|\mathbf{x}| = -\sum_{i=1}^{2M} x_i$. If $\rho|\mathbf{A}| - |\mathbf{x}| \le 0$, then the orienting system remains quiet, allowing resonance and learning to proceed. If $\rho|\mathbf{A}| - |\mathbf{x}| > 0$, then the reset signal r=1, initiating search for a better matching code.

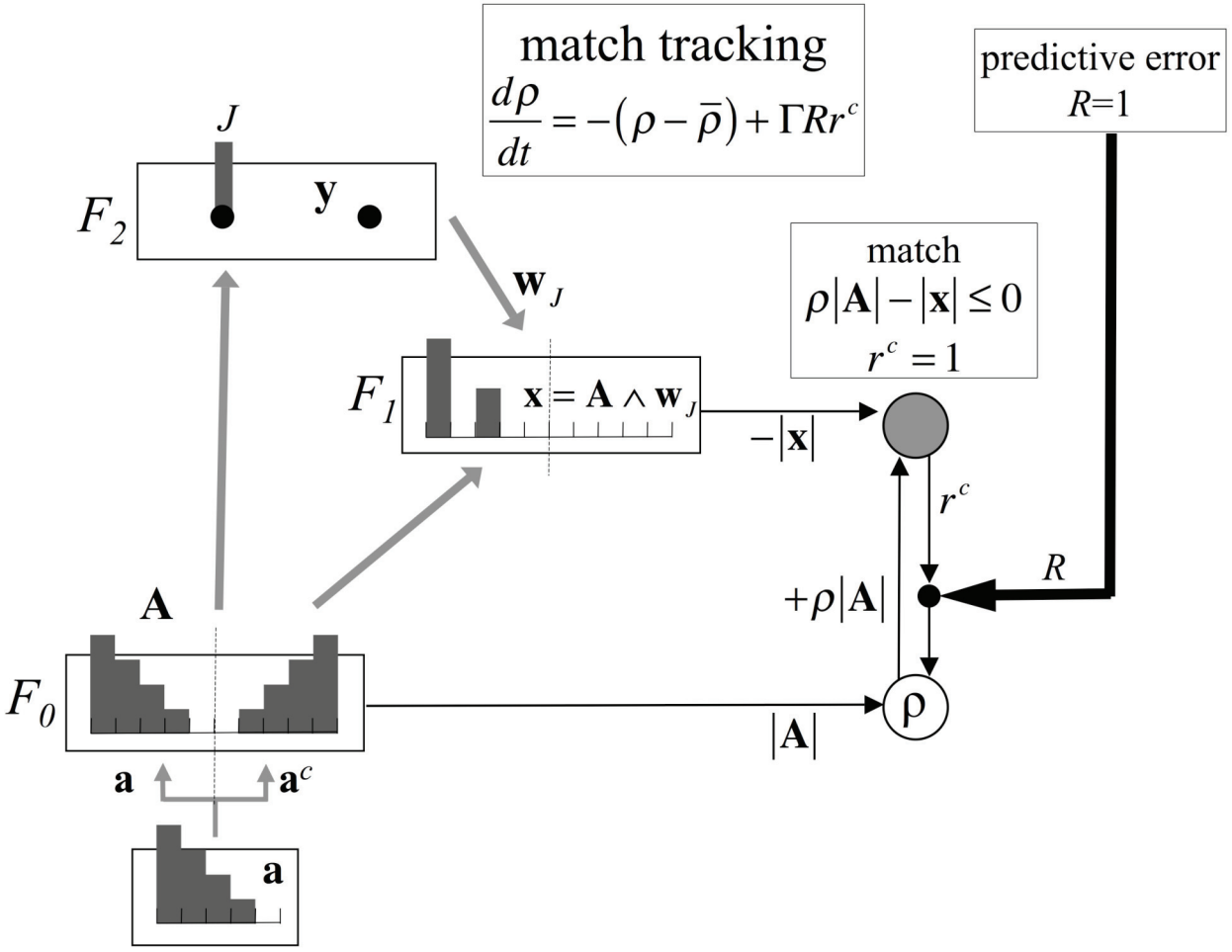

Figure 3. ARTMAP match tracking (Carpenter, Grossberg, & Reynolds, 1991). When an active node J meets the matching criterion $(\rho|\mathbf{A}| - |\mathbf{x}| \le 0)$, the reset signal $r = 0$ and the node makes a prediction. If the predicted output is incorrect, the feedback signal R=1. While $R = r^c = 1$, ρ increases rapidly. As soon as $\rho > \frac{|\mathbf{x}|}{|\mathbf{A}|}$, r switches to 1, which both halts the increase of ρ and resets the active F_2 node. From activation of one chosen node to the next within a search cycle, ρ decays to slightly below $\frac{|\mathbf{x}|}{|\mathbf{A}|}$ (MT–: Carpenter & Markuzon, 1998). On the time scale of learning ρ returns to its baseline level $\bar{\rho}$.

2.3. ART geometry

ART long-term memories are visualized in the *M*-dimensional feature space as hyper-rectangles called *category boxes*. A weight vector $\mathbf{w}_J$ is interpreted geometrically as a box R_J whose ON-channel corner $\mathbf{u}_J$ and opposite OFF-channel corner $\mathbf{v}_J$ are, in the format of the complement-coded input vector, defined by $\left(\mathbf{u}_J \mid \mathbf{v}_J^c\right) \equiv \mathbf{w}_J$ (Figure 4a). For fuzzy ART with the choice-by-difference $F_0 \to F_2$ signal function T_J (Carpenter & Gjaja, 1994), an input **a** activates the node *J* of the closest category box R_J according to the L_1 (city-block) metric. In case of a distance tie, as when **a** lies in more than one box, the node with the smallest R_J is chosen, where the box size $|R_J|$ is defined as the sum of the edge lengths $\sum_{i=1}^{M}\left(v_{iJ} - u_{iJ}\right)$. The chosen node *J* will reset if $|R_J \oplus \mathbf{a}| > M\left(1-\rho\right)$, where $R_J \oplus \mathbf{a}$ is the smallest box enclosing both R_J and **a**. Otherwise, R_J expands toward $R_J \oplus \mathbf{a}$ during learning. With fast learning, $R_J^{new} = R_J^{old} \oplus \mathbf{a}$.

3. Anomalous online learning by fuzzy ARTMAP

The six-point training example (Table 1) is designed to demonstrate how ARTMAP fast online learning may distort memories by undue attention to critical feature patterns. Figure 5 illustrates this anomaly with this small training set, which suggests a medical example. The six-point supervised learning problem presents two feature values (*age*, *temperature*), with each training input labeled as belonging to class *red* or class *blue*. In the training phase, ARTMAP learns to associate class labels with the six sequentially presented inputs (Figure 5a). The six training patterns result in the activation of three coding nodes, represented as category boxes R_1, R_2, R_3. The test phase shows that the *blue* class has overwhelmed the *red* class (Figure 5b), due to recoding that occurs in response to input #6. The training sequence was constructed so that categories *J*=1 and *J*=2 are defined almost entirely by values of the feature *temperature*. Subsequent inputs that activate these categories treat *temperature* as the critical feature, disregarding values of the other feature, *age*. The normally useful strategy of attending to critical features becomes non-adaptive after one of these nodes makes a predictive error. At that point, a system with featural biasing would direct more attention to the previously disregarded feature *age*. Although ARTMAP would correct its errors if the training set were re-presented, real-world learning might provide important examples only once.

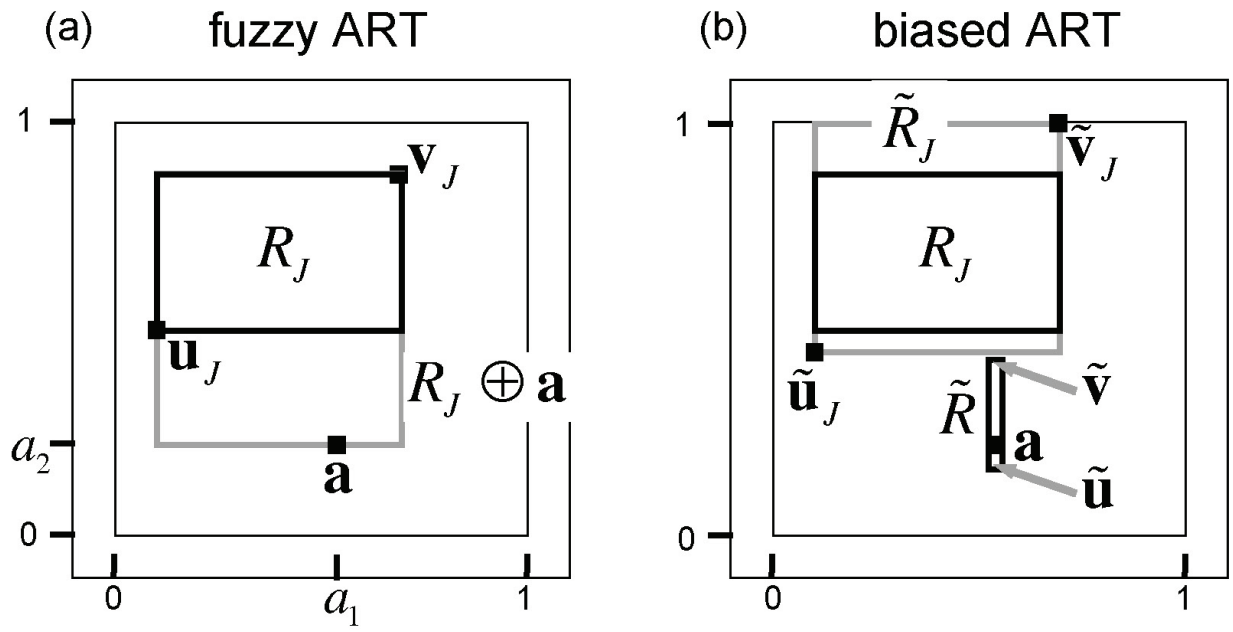

Figure 4. (a) Fuzzy ART geometry. The weight of a category node J is represented in complement-coding form as $\mathbf{w}_J = \left(\mathbf{u}_J \,\middle|\, \mathbf{v}_J^c\right)$, with the M-dimensional vectors $\mathbf{u}_J$ and $\mathbf{v}_J$ defining opposite corners of the category box R_J. When M=2, the size of R_J equals its width plus its height. During learning, R_J expands toward $R_J \oplus \mathbf{a}$, defined as the smallest box enclosing both R_J and $\mathbf{a}$. Node J will reset before learning if $\left|R_J \oplus \mathbf{a}\right| > M(1-\rho)$. (b) Biased ART geometry. The biased weight vector $\tilde{\mathbf{w}}_J = \left(\tilde{\mathbf{u}}_J \,\middle|\, \tilde{\mathbf{v}}_J^c\right)$ defines opposite corners of the biased category box $\tilde{R}_J$, and the biased input vector $\tilde{\mathbf{A}} = \left(\tilde{\mathbf{u}} \,\middle|\, \tilde{\mathbf{v}}^c\right)$ defines opposite corners of the biased input box $\tilde{R}$. The biased matched pattern $\tilde{\mathbf{x}} = \tilde{\mathbf{A}} \wedge \tilde{\mathbf{w}}_J = \left(\left(\tilde{\mathbf{u}} \wedge \tilde{\mathbf{u}}_J\right) \middle| \left(\tilde{\mathbf{v}} \vee \tilde{\mathbf{v}}_J\right)^c\right)$ defines opposite corners of the box $\left|\tilde{R}_J \oplus \tilde{R}\right|$, with $\vee$ denoting the component-wise maximum, or fuzzy union, of two vectors. Node J will reset before learning if $\left(\left|\tilde{R}_J \oplus \tilde{R}\right| - \left|\tilde{R}\right|\right) > \left(M - \left|\tilde{R}\right|\right)(1-\rho)$.

Table 1
Six-point training set.

Six-point example		
Input #	**Input a**	**Output class**
1	(0.0, 0.8)	*red*
2	(1.0, 0.8)	*red*
3	(0.0, 0.9)	*blue*
4	(0.8, 0.9)	*blue*
5	(1.0, 0.4)	*blue*
6	(0.5, 0.7)	*blue*

Figure 5 (following page). Six-point example: fuzzy ARTMAP. (a) For a small-scale prototype medical training dataset, subject records provide values of the features *age* and *temperature*. Inputs #1-2 belong to the class *red*, and inputs #3-6 belong to the class *blue*. (b) Class label predictions (*red* or *blue*) of fuzzy ARTMAP, trained online with winner-take-all coding and with each input presented once. (c) The coding node J=1 (box R_1) maps inputs #1 and #2 to the class *red*. (d) The coding node J=2 (box R_2) maps inputs #3 and #4 to the class *blue*. (e) The coding node J=3 (point box R_3) maps input #5 to the class *blue*. (f) Input #6 (class *blue*) first actives node J=1, which is reset following its incorrect prediction of the class *red*. ART search then activates node J=2, which meets the matching criterion and makes the correct prediction. However, learning then overwhelms most of the early *red* predictions based on inputs #1 and #2. Note that inputs #1-4 define categories J=1 and J=2 primarily in terms of the value of the critical feature *temperature*.

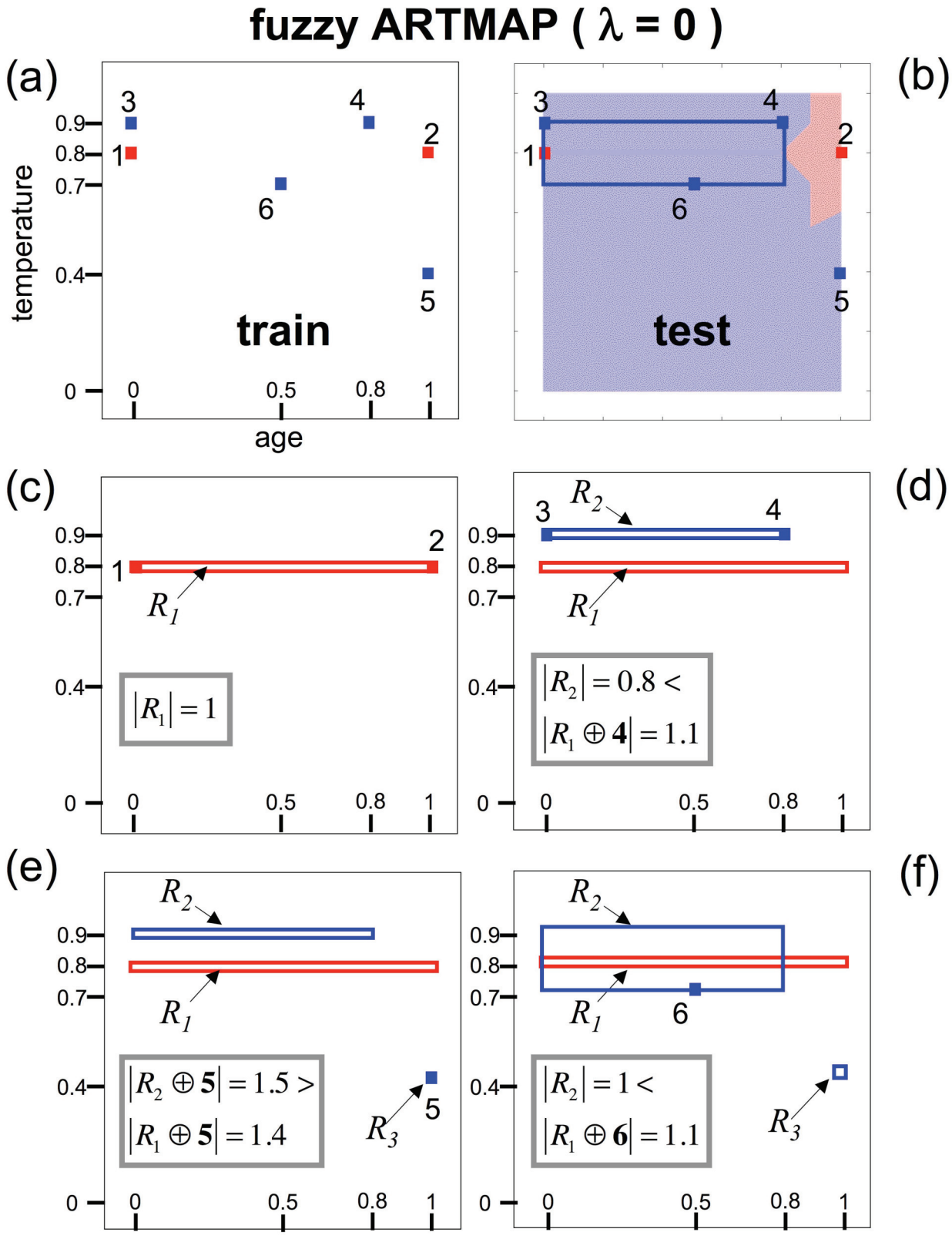
fuzzy ARTMAP (λ = 0)
(a)
temperature
age
train
(b)
test
(c)
R_1
$|R_1| = 1$
(d)
R_2
R_1
$|R_2| = 0.8 <$
$|R_1 \oplus \mathbf{4}| = 1.1$
(e)
R_2
R_1
$|R_2 \oplus \mathbf{5}| = 1.5 >$
$|R_1 \oplus \mathbf{5}| = 1.4$
R_3
(f)
R_2
R_1
$|R_2| = 1 <$
$|R_1 \oplus \mathbf{6}| = 1.1$
R_3

Figure 5

4. Biased ART

A medium-term memory in all ART models allows the network to shift attention among learned categories during search. The network developed here introduces a new medium-term memory that shifts attention among input features, as well as categories, during search. This device corrects the ARTMAP online search anomaly that was illustrated in Figure 5.

4.1. Biasing against previously active category nodes during attentive memory search

Biasing mechanisms are not new to ART network design: they have always played an essential role in the dynamics of search. Biased ART adds qualitatively to these mechanisms within the search cycle, as follows.

Activity **x** at the ART field F_1 continuously computes the match between the field's bottom-up and top-down input patterns. A reset signal r shuts off the active F_2 node J when **x** fails to meet the matching criterion determined by the value of the vigilance parameter ρ. Reset alone does not, however, trigger a search for a different F_2 node: unless the prior activation has left an enduring trace within the F_0-to-F_2 category choice subsystem, the network will simply reactivate the same node as before. As modeled in ART 3 (Carpenter & Grossberg, 1990), biasing the bottom-up input to the coding field F_2 to favor previously inactive nodes implements search by allowing the network to activate a new node in response to a reset signal. The ART 3 search mechanism defines a medium-term memory (MTM) in the F_0-to-F_2 adaptive filter which biases the system against re-choosing a category node that had just produced a reset. A presynaptic interpretation of this bias is transmitter depletion, or habituation, as illustrated in Figure 6.

4.2. Biasing against previously active critical features during attentive memory search

Figure 7a shows how biased ART augments fuzzy ART (Figure 2b) in order to redirect attention away from critical features that have led to incorrect predictions. Biased ART and fuzzy ART are the same within the $F_0 - F_1 - F_2$ complex, where category choice and learning occur. The two networks are also the same during testing, when a system receives no feedback about predictive errors. After node $J=J_1$ produces an incorrect prediction, the bART biasing signal **e** grows in the components of the attended input features $i = 1$ and $i = 2$ (Figure 7b). Pattern **e** transforms the original input $\mathbf{A}$ into the biased input $\tilde{\mathbf{A}}$, and the matched pattern **x** into the biased matched pattern $\tilde{\mathbf{x}}$. Following a reset, which activates a new category node $J=J_2$ (Figure 7c), biased ARTMAP can now pay relatively more attention to input feature $i = 3$, which was not part of the critical feature pattern of the first chosen node $J=J_1$.

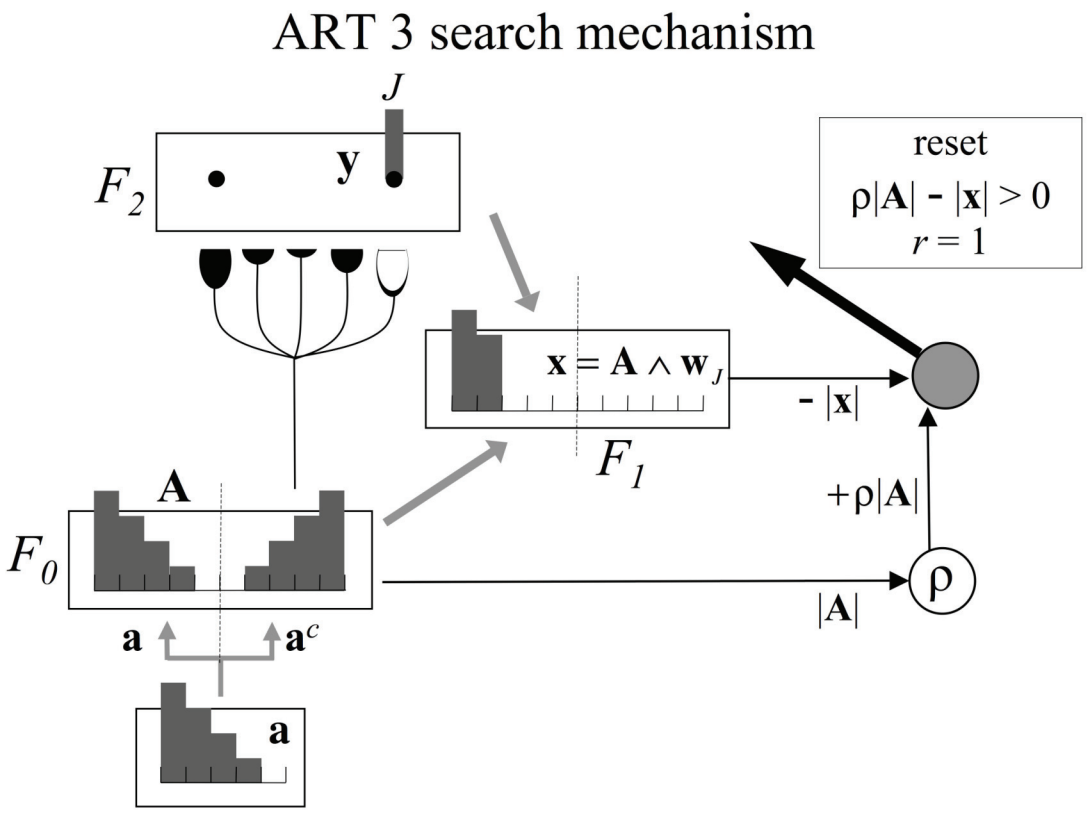

Figure 6. ART 3 search implements a medium-term memory within the F_0-to-F_2 pathways, which biases the system against choosing a category node that has just produced a reset.

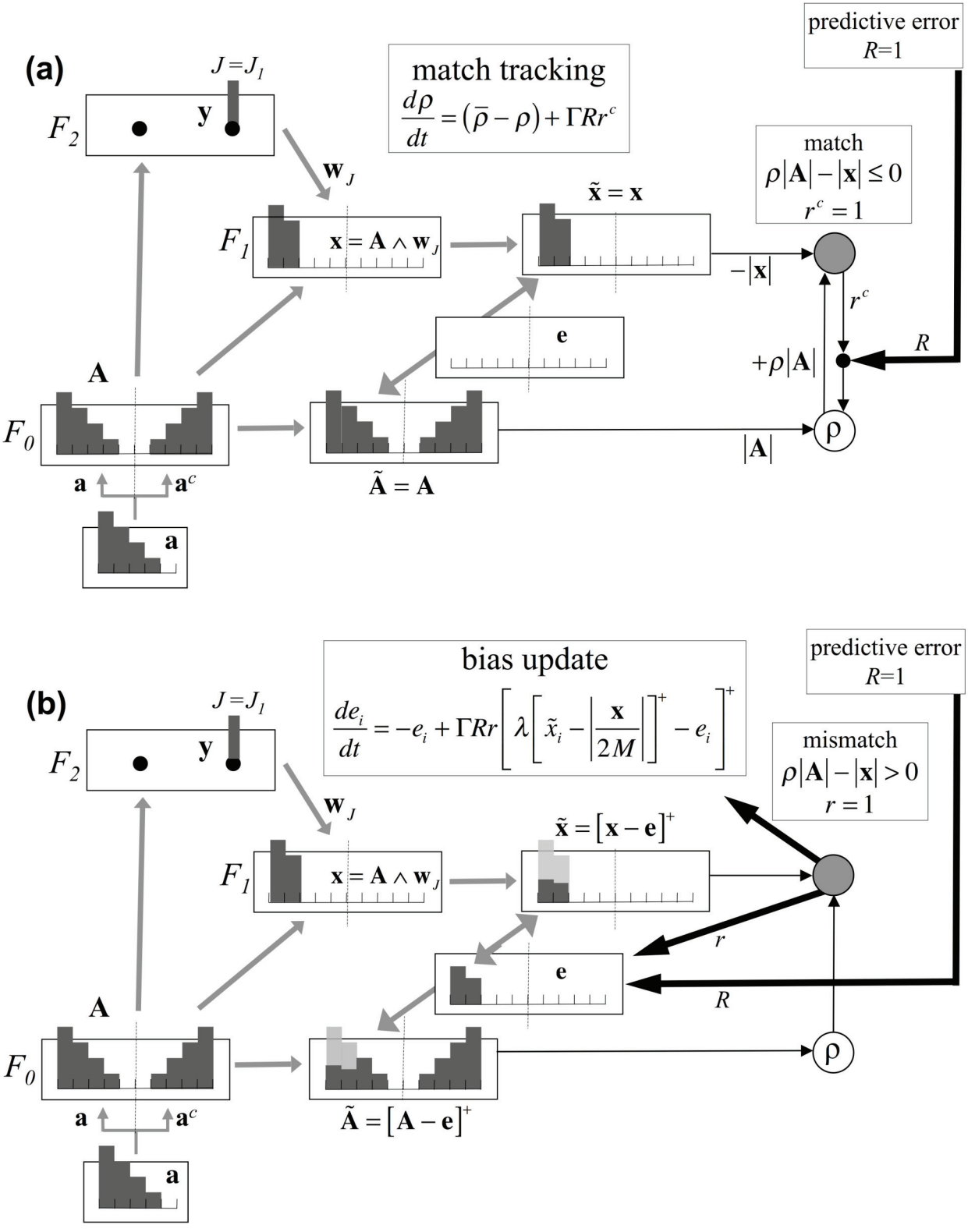
(a)
$J = J_1$
F_2
y
$\mathbf{w}_J$
match tracking
$\frac{d\rho}{dt} = (\bar{\rho} - \rho) + \Gamma R r^c$
predictive error
R=1
match
$\rho|\mathbf{A}| - |\mathbf{x}| \le 0$
$r^c = 1$
F_1
$\mathbf{x} = \mathbf{A} \wedge \mathbf{w}_J$
$\tilde{\mathbf{x}} = \mathbf{x}$
$-|\mathbf{x}|$
r^c
R
$+\rho|\mathbf{A}|$
e
A
F_0
a
$\mathbf{a}^c$
$\tilde{\mathbf{A}} = \mathbf{A}$
$|\mathbf{A}|$
ρ
(b)
bias update
$\frac{de_i}{dt} = -e_i + \Gamma R r \left[\lambda \left[\tilde{x}_i - \left| \frac{\mathbf{x}}{2M} \right| \right]^+ - e_i \right]^+$
predictive error
R=1
mismatch
$\rho|\mathbf{A}| - |\mathbf{x}| > 0$
$r = 1$
$\tilde{\mathbf{x}} = [\mathbf{x} - \mathbf{e}]^+$
r
$\tilde{\mathbf{A}} = [\mathbf{A} - \mathbf{e}]^+$

Figure 7

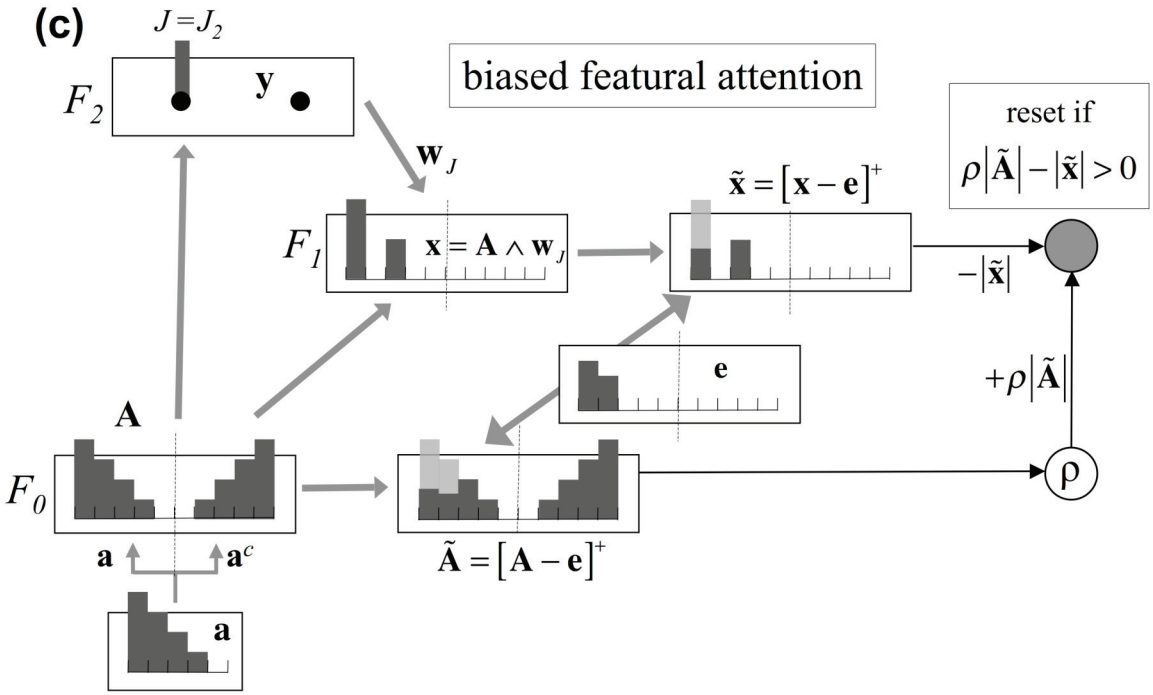

Figure 7. Biased ARTMAP search. After the system makes a predictive error, the pattern **e** directs attention away from input features that were recently in critical feature patterns of active F_2 nodes. In response to a predictive error when $J=J_1$, both fuzzy ARTMAP (Figure 2b) and biased ARTMAP raise ρ to just above 0.35. When $J=J_2$, fuzzy ARTMAP resets because $\frac{|\mathbf{x}|}{|\mathbf{A}|}=\frac{1.5}{5}=0.3 < \rho$. Biased ARTMAP also resets, with $\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|}=\frac{0.881}{3.950}=0.223 < \rho$. If ρ had been between 0.223 and 0.3, only biased ARTMAP would have reset. For this illustration, where λ=3, computational details appear in Section 6.1, following the biased ARTMAP system specification (Section 5). The rectification operator $[\mathbf{z}]^+$ truncates negative components of a vector **z** at 0, i.e., $[\mathbf{z}]_i^+ \equiv \max\{z_i, 0\} \geq 0$.

4.3. Biased ART corrects the ART featural attention anomaly

Figure 8 illustrates how redirected featural attention during search produces a new memory structure for the six-point example. Each component of the biasing vector **e** equals zero at the start of an input presentation. Thus, until the system makes a predictive error, the biased input $\widetilde{\mathbf{A}}$ equals the unbiased input $\mathbf{A}$ and the biased matched pattern $\widetilde{\mathbf{x}}$ equals the unbiased matched pattern **x**. Following a predictive error, the biasing signal e_i grows for attended features *i*, where components x_i of the matched pattern are large. After **A** chooses a new category node, the patterns $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{x}}$ are selectively biased against these previously attended features.

Until **A** produces a predictive error, search and learning are the same in biased ARTMAP as in fuzzy ARTMAP. Even after a predictive error makes $|\mathbf{e}| > 0$, the two networks may still choose and learn on the same category nodes, as for input #4 in the six-point example (Figure 5d). Thus biased ARTMAP learning on inputs #1-5 (Figure 8c-e) is the same as for fuzzy ARTMAP (Figure 5c-e).

When input #6 activates node J=1, bARTMAP produces a bias against the feature *temperature*, whose value alone defines this category. The network then rejects node J=2, which is also defined almost entirely by the value of *temperature*. The system can then search further, resulting in activation of the more successful category J=3 (Figure 8f). Modification of the weight vector $\mathbf{w}_3$ results in non-overlapping category representations and better test set predictions (Figure 8b). As here, across a variety of computed examples biased ARTMAP characteristically reduces test set errors and decreases category box overlap, without increasing memory size.

Figure 8 (following page). Six-point example: biased ARTMAP. In response to the initial predictive error (*red*) made when input #6 chooses node J=1, biased ARTMAP shifts attention away from feature $i = 2$ (*temperature*) toward the feature $i = 1$ (*age*), resulting in internal category representations that do not distort the early learning of the class *red*. When ($0<\lambda\leq 0.62$), biased ARTMAP produces the same memories as fuzzy ARTMAP ($\lambda=0$) (Figure 5). When $\lambda>18$, biased ARTMAP's input #4 resets node J=2, producing a new *blue* point box. Computational details appear in Section 6.2.

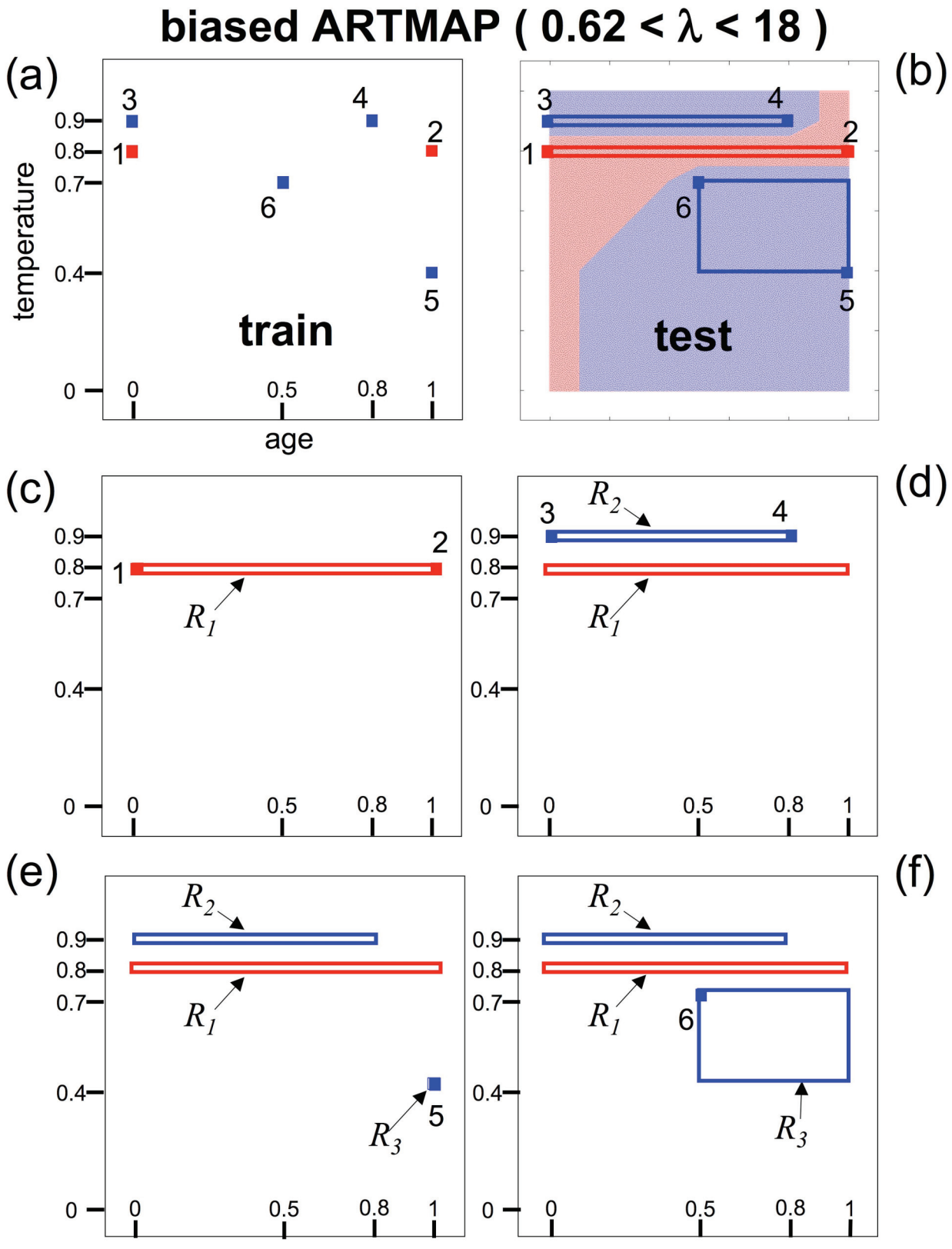
biased ARTMAP (0.62 < λ < 18)
(a)
(b)
(c)
(d)
(e)
(f)
temperature
age
train
test
0.9
0.8
0.7
0.4
0
0.5
1
R_1
R_2
R_3

Figure 8

4.4. Biased ART geometry

Figure 4b illustrates biased ART geometry. As for fuzzy ART, an input **a** activates the node *J* of the closest category box R_J and, unless reset, R_J expands toward $R_J \oplus \mathbf{a}$ during learning. The two systems differ in the reset decision, with $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{x}}$ replacing **A** and **x** in biased ART. Formally, the biased matched pattern $\tilde{\mathbf{x}} = \tilde{\mathbf{A}} \wedge \tilde{\mathbf{w}}_J$, though $\tilde{\mathbf{x}}$ is computed as $[\mathbf{x} - \mathbf{e}]^+$, and the biased ART network does not actually represent $\tilde{\mathbf{w}}_J$ (Figure 7). Having been chosen, a biased ART node *J* resets if $\rho|\tilde{\mathbf{A}}| - |\tilde{\mathbf{x}}| > 0$. Note that when the chosen node *J* is uncommitted, $\tilde{\mathbf{x}} = \tilde{\mathbf{A}}$, so $\rho|\tilde{\mathbf{A}}| - |\tilde{\mathbf{A}}| \le 0$ and *J* is not reset.

The dynamics and geometry of biased ART category choice and learning are the same as for fuzzy ART. During search, by analogy with fuzzy ART geometry, a box $\tilde{R}$ represents the biased input $\tilde{\mathbf{A}} \equiv (\tilde{\mathbf{u}} \mid \tilde{\mathbf{v}}^c)$ and a box $\tilde{R}_J$ represents the biased weight vector $\tilde{\mathbf{w}}_J \equiv (\tilde{\mathbf{u}}_J \mid \tilde{\mathbf{v}}_J^c)$. Rectification of $\tilde{\mathbf{A}} = [\mathbf{A} - \mathbf{e}]^+$ and $\tilde{\mathbf{w}}_J = [\mathbf{w}_J - \mathbf{e}]^+$ place $\tilde{R}$ and $\tilde{R}_J$ within the unit box. Thus in Figure 4b $\tilde{R}_J$ does not extend as far above R_J as $\tilde{R}$ extends above **a**.

$\tilde{R}_J \oplus \tilde{R}$ is defined as the smallest box enclosing both $\tilde{R}_J$ and $\tilde{R}$, and $|\tilde{R}_J \oplus \tilde{R}| = M - |\tilde{\mathbf{x}}|$ and $|\tilde{R}| = M - |\tilde{A}|$. In biased ARTMAP, node *J* resets if $\rho|\tilde{\mathbf{A}}| - |\tilde{\mathbf{x}}| > 0$. Thus, geometrically, node *J* resets if $\left(|\tilde{R}_J \oplus \tilde{R}| - |\tilde{R}|\right) > \left(M - |\tilde{R}|\right)(1 - \rho)$. Before a given input makes any predictive errors, **e**=**0** and $\tilde{R}$ is the point box **a**. Biased ART geometry then reduces to fuzzy ART geometry, with reset when $|R_J \oplus \mathbf{a}| > M(1 - \rho)$.

5. Biased ARTMAP model

Computation of the biasing vector **e** is embedded in fuzzy ARTMAP as follows.

5.1. Variables

Bias vector	$\mathbf{e} \equiv \left(e_1 \ldots e_i \ldots e_{2M}\right)$
Biased input vector	$\tilde{\mathbf{A}} \equiv \left[\mathbf{A} - \mathbf{e}\right]^+ \equiv \left(\tilde{\mathbf{u}} \mid \tilde{\mathbf{v}}^c\right)$
Biased weight vector	$\tilde{\mathbf{w}}_J \equiv \left[\mathbf{w}_J - \mathbf{e}\right]^+ \equiv \left(\tilde{\mathbf{u}}_J \middle\vert \tilde{\mathbf{v}}_J^c\right)$
Biased matched vector	$\tilde{\mathbf{x}} \equiv \left[\mathbf{x} - \mathbf{e}\right]^+ = \tilde{\mathbf{w}}_J \wedge \tilde{\mathbf{A}}$
Vigilance	ρ
Mismatch reset	$r = \begin{cases} 1 & \text{if } \rho\left\vert\tilde{\mathbf{A}}\right\vert - \left\vert\tilde{\mathbf{x}}\right\vert > 0 \\ 0 & \text{otherwise} \end{cases}$
Predictive error	$R = \begin{cases} 1 & \text{when the active node } J \text{ makes a predictive error} \\ 0 & \text{otherwise} \end{cases}$

5.2. Parameters

The bias parameter λ is the only free parameter of biased ARTMAP. In all simulations, ARTMAP parameters assume the default values listed below.

$\lambda \geq 0$	Bias parameter Biased ARTMAP reduces to fuzzy ARTMAP when $\lambda = 0$. By default, $\lambda = 10$.
$\Gamma \gg 1$	Fast integration of MTM variables ρ and e_i after a predictive error
$\alpha = 0^+$	ARTMAP choice parameter By default, $\alpha = 10^{-8}$.
$\beta \in [0,1]$	ARTMAP learning rate parameter By default, β=1 (fast learning).
$\bar{\rho} \in [0,1]$	ARTMAP baseline vigilance parameter By default, $\bar{\rho} = 0$.
$\varepsilon = 0^-$	ARTMAP match tracking parameter (MT–) By default, $\varepsilon = -10^{-5}$.
$w_{iJ} = 1$	Weight initial values

5.3. New training input

When a new input **a** is first presented,

$$\mathbf{e} = \mathbf{0}$$
$$\tilde{\mathbf{A}} = \mathbf{A} \equiv \left(\mathbf{a} \mid \mathbf{a}^c\right)$$
$$\rho = \bar{\rho}$$
$$r = R = 0$$

5.4. ARTMAP category choice

Among F_2 category nodes j that have not been reset, the chosen node J maximizes the choice-by-difference $F_0 \to F_2$ signal function $T_j = \left|\mathbf{A} \wedge \mathbf{w}_j\right| + (1-\alpha)\left(M - \left|\mathbf{w}_j\right|\right)$. For uncommitted nodes, $T_j = \alpha M$. Winner-take-all coding assumes slight tie-breaking variations, so that the system chooses one committed or uncommitted node at a time.

Matched pattern: $\mathbf{x} = \mathbf{A} \wedge \mathbf{w}_J$

5.5. Match or mismatch?

Biased matched pattern: $\tilde{\mathbf{x}} = \left[\mathbf{x} - \mathbf{e}\right]^+$

If node J fails to meet the matching criterion, i.e., if $\frac{\left|\tilde{\mathbf{x}}\right|}{\left|\tilde{\mathbf{A}}\right|} < \rho$, then node *J* is shut off for the duration of the search cycle (mismatch reset).

A then chooses another node *J* (GO TO: *5.4. ARTMAP category choice*). In this case, *J* does not make an output class prediction, so R remains equal to 0 and neither ρ nor e_i increases.

If node J meets the matching criterion, it makes an output prediction.

If the prediction is correct, or if *J* is an uncommitted node, learning ensues (GO TO: *5.9. ARTMAP learning*).

If the prediction is incorrect, vigilance increases enough to reset *J* (GO TO: *5.6. Match tracking*).

5.6. Match tracking

When the active node J meets the matching criterion but makes a predictive error, $r = 0$ and $R = 1$. Vigilance ρ then quickly increases according to the ARTMAP match tracking equation:

$$\frac{d\rho}{dt} = -\left(\rho - \bar{\rho}\right) + \Gamma R r^c \ .$$

Vigilance stops increasing as soon as $r = 1$, at which point ρ is infinitesimally larger than the match value $\frac{\left|\tilde{\mathbf{x}}\right|}{\left|\tilde{\mathbf{A}}\right|}$.

MT–: On the time scale of search, ρ decays by ε before the next coding node is chosen.

5.7. Bias update

When $R = r = 1$ (after a predictive error has produced a match tracking reset), bias vector components e_i $(i = 1 \ldots 2M)$ quickly increase according to the bias update equation:

$$\frac{de_i}{dt} = -e_i + \Gamma R r \left[\lambda \left[\tilde{x}_i - \frac{|\mathbf{x}|}{2M} \right]^+ - e_i \right]^+ ,$$

where $\tilde{\mathbf{x}} \equiv [\mathbf{x} - \mathbf{e}]^+$ is continuously updated as the bias variables e_i increase. Note that $\frac{|\mathbf{x}|}{2M}$ is the average value of components of the matched pattern **x**. Attended features are hereby defined as those of above-average activation, approximating the dynamics of a competitive network.

While $R = r = 1$,

$$\frac{1}{\Gamma}\frac{de_i}{dt} \cong \left[\lambda \left[[x_i - e_i]^+ - \frac{|\mathbf{x}|}{2M} \right]^+ - e_i \right]^+ .$$

Γ is assumed to be large enough so that e_i reaches equilibrium before node J shuts off, which will switch the predictive error signal R back to 0.

Like ρ, e_i values decay only slightly on the rapid time scale of search.

5.8. Bias model solution

Starting with $e_i = e_i^{old}$, $e_i \to e_i^{new}$ for $0 \le \lambda \le \infty$.

Case 1 $e_i^{new} = e_i^{old}$ if $\lambda \left[[x_i - e_i^{old}]^+ - \frac{|\mathbf{x}|}{2M} \right] \le 0$

Case 2 $e_i^{new} = e_i^{old}$ if $e_i^{old} \ge \lambda \left[[x_i - e_i^{old}]^+ - \frac{|\mathbf{x}|}{2M} \right] > 0$

Case 3 $e_i^{new} = \frac{\left[x_i - \frac{|\mathbf{x}|}{2M} \right]}{1 + \lambda^{-1}}$ if $x_i > e_i^{old}$ and $\lambda \left[[x_i - e_i^{old}] - \frac{|\mathbf{x}|}{2M} \right] > e_i^{old}$.

Biased input: $\tilde{\mathbf{A}} = [\mathbf{A} - \mathbf{e}]^+$

A chooses another node J (GO TO: *5.4. ARTMAP category choice*).

5.9. ARTMAP learning

Update weights: $\mathbf{w}_J^{new} = (1-\beta)\mathbf{w}_J^{old} + \beta\left(\mathbf{w}_J^{old} \wedge \mathbf{A}\right)$

With fast learning, $\mathbf{w}_J^{new} = \mathbf{w}_J^{old} \wedge \mathbf{A}$.

The time scale of learning is assumed to be much greater than the medium-term memory time scale. Thus vigilance ρ and the bias variables e_i decay to their initial values as $\mathbf{w}_J \rightarrow \mathbf{w}_J^{new}$. Biasing during search affects the choice of the final node *J*, but does not otherwise affect learning.

Choose the next input **a** (GO TO: *5.3. New training input*).

5.10. ARTMAP testing

Testing is the same as training, with $\rho \equiv 0$ (no reset) and no learning. Since the system receives no feedback about which outputs are correct, *R*=0 during testing, so all bias terms e_i remain equal to 0.

6. Biased ARTMAP illustrations

When the choice parameter α and the match tracking parameter ε are small, fuzzy ARTMAP dynamics may be calculated by hand, with each step visualized geometrically (Figure 5). Similarly, biased ARTMAP dynamics may be hand-calculated, as shown here for the examples illustrated in Figure 7 and Figure 8.

6.1. Calculations for the Figure 7: Biased ARTMAP search

6.1.1. Figure 7a: Match tracking

$\mathbf{A} = (1, 0.75, 0.5, 0.25, 0 \mid 0, 0.25, 0.5, 0.75, 1), \; M = 5$

Match tracking reset at $J = J_1$: $\mathbf{w}_J = (1, 0.75, 0, 0, 0 \mid 0, 0, 0, 0, 0)$

$\mathbf{x} = \mathbf{A} \wedge \mathbf{w}_J = (1, 0.75, 0, 0, 0 \mid 0, 0, 0, 0, 0), \; \frac{|\mathbf{x}|}{2M} = \frac{1.75}{10} = 0.175$

$\rho \rightarrow \frac{|\mathbf{x}|}{|\mathbf{A}|} = \frac{1.75}{5} = 0.35$

6.1.2. Figure 7b: Bias update

$$e_1 \to \frac{x_1 - \frac{|\mathbf{x}|}{2M}}{1+\lambda^{-1}} = \frac{x_1 - 0.175}{1+\lambda^{-1}} = \frac{0.825}{1+\lambda^{-1}}, \quad e_2 \to \frac{x_2 - 0.175}{1+\lambda^{-1}} = \frac{0.575}{1+\lambda^{-1}}$$

$\lambda = 3$: $e_1 = 0.619$, $e_2 = 0.431$

$$\tilde{A}_1 = \left[A_1 - e_1\right]^+ = \left[1 - e_1\right]^+ = 0.381$$

$$\tilde{A}_2 = \left[0.75 - e_2\right]^+ = 0.319$$

6.1.3. Figure 7c: Biased featural attention

$J = J_2$: $\mathbf{w}_J = (1, 0, 0.5, 0, 0 \mid 0, 0, 0, 0, 0) = \mathbf{A} \wedge \mathbf{w}_J = \mathbf{x}$

$$\tilde{x}_1 = \left[x_1 - e_1\right]^+ = \left[1 - e_1\right]^+ = 0.381$$

$$\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|} = \frac{1.5 - e_1}{5 - e_1 - e_2} = \frac{0.881}{3.950} = 0.223$$

6.2. Calculations for Figure 8: Six-point example

The biased ARTMAP learning illustrated in Figure 8d requires that $\lambda \le \bar{\bar{\lambda}} = \frac{109}{6} = 18.17$, and Figure 8f requires that $\lambda > \bar{\lambda} = \frac{2}{3.225} = 0.620$, as follows.

$\lambda \le \bar{\bar{\lambda}}$: Input point #4 (0.8, 0.9) [*blue*] first chooses node J=1, which incorrectly predicts *red*, with $\mathbf{A} = (0.8, 0.9 \mid 0.2, 0.1)$, $\mathbf{w}_1 = (0, 0.8 \mid 0, 0.2)$, and $\mathbf{x} = (0, 0.8 \mid 0, 0.1)$.

Match tracking raises ρ to $\frac{|\mathbf{x}|}{|\mathbf{A}|} = \frac{0.9}{2} = 0.45$ and e_2 to $\frac{x_2 - \frac{|\mathbf{x}|}{2M}}{1+\lambda^{-1}} = \frac{0.8 - \frac{0.9}{4}}{1+\lambda^{-1}} = \frac{0.575}{1+\lambda^{-1}}$.

Input #4 then chooses node J=2, with $\mathbf{w}_2 = (0, 0.9 \mid 1, 0.1)$, $\mathbf{x} = (0, 0.9 \mid 0.2, 0.1)$, $\tilde{\mathbf{A}} = (0.8, 0.9 - e_2 \mid 0.2, 0.1)$, and $\tilde{\mathbf{x}} = (0, 0.9 - e_2 \mid 0.2, 0.1)$. This node will produce another mismatch reset unless $\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|} \ge \rho$, i.e., $e_2 \le \frac{6}{11}$ or $\lambda \le \frac{109}{6} = 18.17 \equiv \bar{\bar{\lambda}}$.

$\bar{\lambda} < \lambda$: Input point #6 (0.5, 0.7) [*blue*] first chooses node *J*=1, which incorrectly predicts *red*. Match tracking raises ρ to 0.45 and e_2 to $\frac{0.475}{1+\lambda^{-1}}$.

A subsequent mismatch reset at $J = 2$ requires that $e_2 > 0.182$, i.e., $\lambda > \frac{2}{3.225} = 0.620 \equiv \bar{\lambda}$.

No reset occurs when point #6 chooses $J = 3$: With $\rho = 0.45$ and $\bar{\lambda} < \lambda \le \bar{\bar{\lambda}}$, the minimum value of $\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|}$ is 0.5, which is realized when $e_2 = 0.4$ $(\lambda = 5.33)$.

7. Biased ARTMAP examples analytically computed

7.1. Biasing in favor of previously inattended features

The six-point prototype example (Figure 8) illustrates how biased ARTMAP may generate extra resets by biasing *against* previously attended features. This example might produce the hypothesis that biasing, like higher baseline vigilance, can only add more committed nodes to memory. However, calculations for a similar example with three training exemplars (Table 2) show how bARTMAP mechanisms may generate fewer resets, by biasing *in favor of* features that had not been previously attended within the search (Figure 9).

Table 2
Three-point training set.

Three-point example		
Input #	**Input a**	**Output class**
1	(1.0, 0.1)	*red*
2	(0.0, 1.0)	*blue*
3	(1.0, 1.0)	*blue*

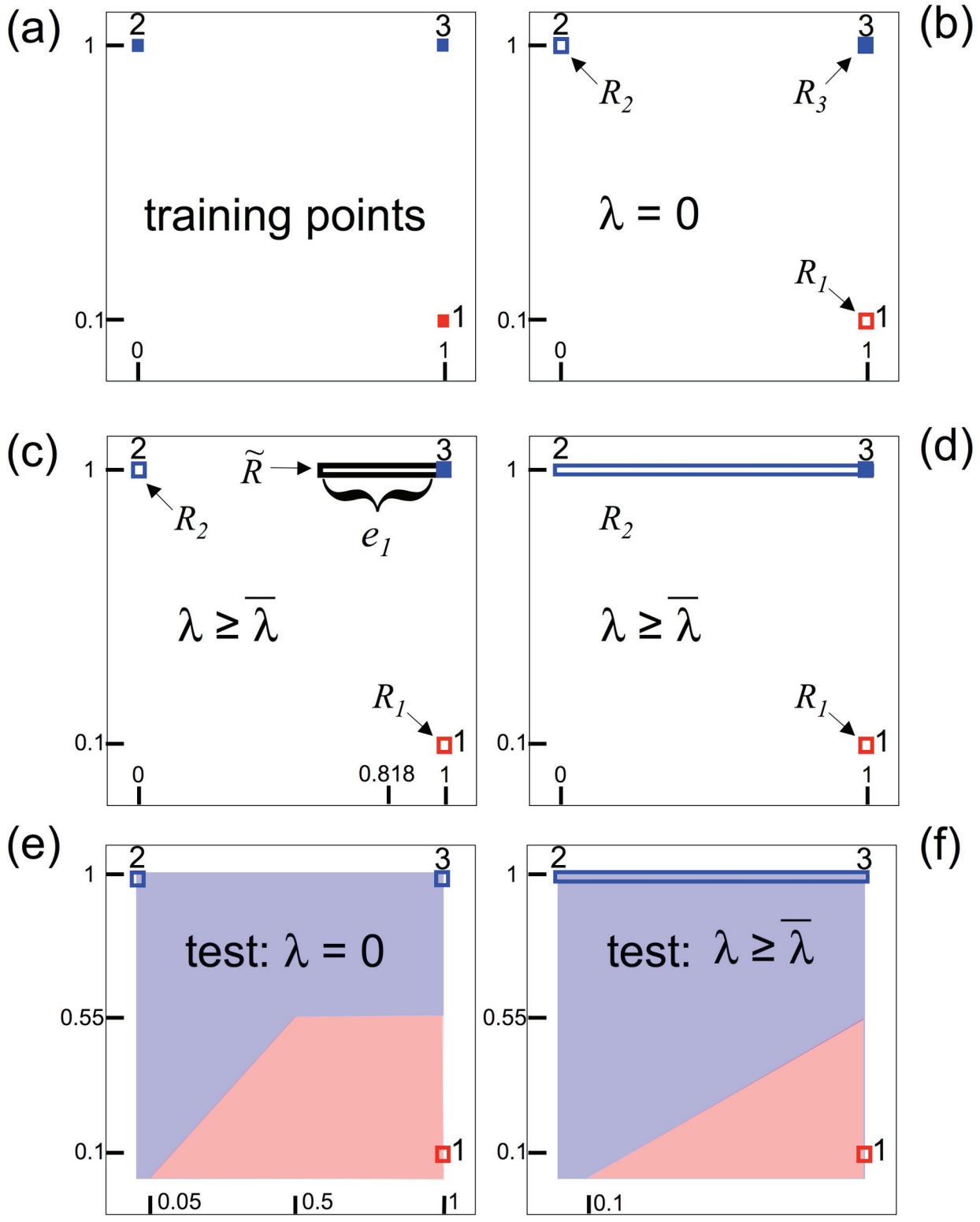

Figure 9. Three-point example. In (c), $\lambda = 1.231$ ($e_1 = 0.4$) . Bias parameters $\infty \geq \lambda \geq 0.335 = \bar{\lambda}$ $(0.725 \geq e_1 \geq 0.182)$ all give the same learned system and test results (f).

7.2. Calculations for Figure 9: Three-point example

7.2.1. Figure 9a

For training point #1, **a** = (1.0, 0.1) is mapped to class *red*, and for training points #2 and #3, **a** = (0, 1) and **a** = (1, 1) are mapped to class *blue*.

7.2.2. Figure 9b

For all $\lambda \geq 0$, training point #1 creates the point box R_1.

Training point #2 first chooses $J = 1$, which makes a predictive error, and creates the point box R_2.

Training point #3 first chooses J=1, which makes a predictive error, raising ρ to $\frac{|\mathbf{x}|}{|\mathbf{A}|} = \frac{M - |R_1 \oplus \mathbf{a}|}{M} = \frac{2 - 0.9}{2} = 0.55$.

For fuzzy ARTMAP ($\lambda = 0$), node J resets if

$$|R_J \oplus \mathbf{a}| > M(1 - \rho). \quad (1)$$

When $\lambda = 0$, the next chosen node $J = 2$ resets because $|R_2 \oplus \mathbf{a}| = 1 > M(1 - \rho) \cong 0.9$, so training point #3 creates the point box R_3.

7.2.3. Figure 9c

Biased ARTMAP learning is the same as fuzzy ARTMAP's, up to training point #3. Following activation of node J=1, match tracking raises ρ to 0.55, which switches the reset signal to $r = 1$. At this point, bARTMAP also raises e_i for attended features i, in this case for $i = 1$, as follows.

$\mathbf{A} = (1,1 \mid 0,0)$,

$\mathbf{w}_1 = (1,0.1 \mid 0,0.9)$,

$\mathbf{x} = \mathbf{A} \wedge \mathbf{w}_1 = (1,0.1 \mid 0,0)$,

$e_i(0) = 0$,

$\frac{1}{\Gamma}\frac{d}{dt}e_i = \left[\lambda\left[[x_i - e_i]^+ - \frac{|\mathbf{x}|}{2M}\right]^+ - e_i\right]$, and

$\frac{|\mathbf{x}|}{2M} = \frac{1.1}{4} = 0.275$.

Thus only e_1 increases, and

$$e_1 \to \frac{x_1 - \frac{|\mathbf{x}|}{2M}}{1 + \lambda^{-1}} = \frac{1 - 0.275}{1 + \lambda^{-1}} = \frac{0.725}{1 + \lambda^{-1}}.$$

The biased input #3 then becomes $\tilde{\mathbf{A}} = (1 - e_1, 1 \mid 0,0)$.

The next chosen node is $J = 2$.
Biased ARTMAP does *not* reset iff

$$\left|\tilde{R}_J \oplus \tilde{R}\right| \le M(1-\rho) + \rho\left|\tilde{R}\right| \ . \qquad (2)$$

In this example, $\left|\tilde{R}_2 \oplus \tilde{R}\right| = \left|R_2 \oplus \mathbf{a}\right| = 1$, and the left-hand side of equation (2) is the same as the left-hand side of equation (1).
However, $\left|\tilde{R}\right| = e_1 > 0$, so the right-hand side of equation (2) is greater than the right-hand side of equation (1), reflecting a bias in favor of feature $i = 2$.
Biased ARTMAP will thus *not* reset if $1 \le 2(1-0.55) + 0.55\left|\tilde{R}\right|$.
In this case, node $J = 2$ does *not* reset if $\left|\tilde{R}\right| = e_1 = \dfrac{0.725}{1+\lambda^{-1}} \ge \dfrac{0.1}{0.55} \cong 0.182$,
i.e., if $\lambda \ge \bar{\lambda} = \dfrac{0.182}{0.543} = 0.335$.
In Figure 9c, $\lambda = 1.231$ and $e_1 = 0.4$.

7.2.4. Figure 9d
Since node $J = 2$ does not reset when $\lambda \ge \bar{\lambda}$, and J=2 predicts the correct output class (*blue*), learning ensues, expanding R_2 to include input point #3.

7.2.5. Figure 9e,f
Comparing the *red* / *blue* test set response profiles, the bARTMAP bias favoring feature $i = 2$, which allowed the expansion of category box R_2 , is reflected in the larger number of test points that predict the class *blue*.

7.3. Binary example

Another small example, with inputs that specify six features a_i with values 0 or 1, indicates how biased ARTMAP works in the binary domain. For this example, fuzzy ARTMAP and biased ARTMAP ($\lambda > 9$) each commits three coding nodes during training on a sequence of inputs #1-6 (Table 3). Test predictions differ between the two ARTMAP systems on three of the six training patterns. As in the six-point example (Figure 5), fuzzy ARTMAP's focus on certain critical features at the expense of others causes learning of later patterns to obscure predictions of earlier patterns during online training.

During training on the binary example, fuzzy ARTMAP and biased ARTMAP activate the same category nodes and produce the same learned weight vectors for inputs #1-5:

$\mathbf{w}_1 = (\mathbf{000011} \mid 111100) \rightarrow$ *negative* Input #1

$\mathbf{w}_2 = (\mathbf{11}0\mathbf{1}00 \mid 000000) \rightarrow$ *positive* Inputs #2,3

$\mathbf{w}_3 = (\mathbf{1111}\mathbf{0}0 \mid 000010) \rightarrow$ *negative* Inputs #4,5

The binary vector $\mathbf{a} = (a_1 \ldots a_i \ldots a_M)$ indicates the "presence" of feature *i* when $a_i = 1$ and the "absence" of feature *i* when $a_i = 0$. The weight vector $\mathbf{w}_j$ "attends to" the presence or absence of *i* as a critical feature if $w_{ij} = 1$ or if $w_{(i+M)j} = 1$. If $w_{ij} = w_{(i+M)j} = 0$, neither the presence nor the absence of feature *i* influences the F_2 category node choice function T_j, and node *j* "does not care about" the value of a_i in the input. Critical features *i* = 1…*M* of node *J* are printed in boldface.

Table 3
Binary training set.
(a) Six binary inputs are presented once, in order 1-6. Inputs #2 and #3 are mapped to the output class *positive* (+), and the other inputs are mapped to the output class *negative* (–).

(a) Binary example: training set				
Input #	**Input** $\mathbf{A} = (\mathbf{a} \mid \mathbf{a}^c)$	**Traning output class**	**fuzzy ARTMAP (λ=0) test prediction**	**biased ARTMAP (λ>9) test prediction**
1	$(000011 \mid 111100)$	–	–	–
2	$(111110 \mid 000001)$	+	+/– tie	+
3	$(110101 \mid 001010)$	+	+	+
4	$(111100 \mid 000011)$	–	+/– tie	–
5	$(111101 \mid 000010)$	–	+/– tie	–
6	$(111011 \mid 000100)$	–	–	–

Table 3
(b) Fuzzy ARTMAP and biased ARTMAP each commits three nodes, but make different +/– predictions on 19 of the 64 binary test inputs. Three of these different predictions occur on training inputs, where fuzzy ARTMAP forgets the trained class labels, predicting +/– ties. Biased ARTMAP correctly predicts the classes of all six training inputs during testing.

(b) Binary example: test set		
$64 = 2^6$ test predictions	**fuzzy ARTMAP (λ=0) # predicted classes**	**biased ARTMAP (λ>9) # predicted classes**
+ $J = 2$	13	16
+ / – ties $J = 2 / 3$	21	12
– $J = 1 / 3$	30 13 $J = 1$ 17 $J = 3$	36 24 $J = 1$ 12 $J = 3$

In response to input #6, with **A** = (111011 | 000100), each system first chooses node $J = 2$, which incorrectly predicts the output class +. At F_1, the matched pattern is $\mathbf{x} = (\mathbf{11}0000 \mid 000000)$, so match tracking raises ρ to $\frac{|\mathbf{x}|}{M} = \frac{1}{3}$, triggering a search which next activates the F_2 node $J = 3$.

With J=3, fuzzy ARTMAP produces the matched pattern $\mathbf{x} = (\mathbf{111}000 \mid 000000)$, which meets the matching criterion, with $\frac{|\mathbf{x}|}{M} = \frac{1}{2} > \frac{1}{3} = \rho$. After learning, $\mathbf{w}_3 = \mathbf{x} = (\mathbf{111}000 \mid 000000)$. Testing proves, however, that the network's attention to critical features i = 1, 2, and 3 at this moment obscures much of what the system had previously learned. During testing, fuzzy ARTMAP predicts +/– ties for training inputs #2, #4, and #5.

When biased ARTMAP activates node $J = 2$ in response to input #6, the match tracking signal R, in addition to gating $\frac{d\rho}{dt}$ and raising ρ to $\frac{1}{3}$, also gates $\frac{de_i}{dt}$. With the matched pattern

$\mathbf{x} = (\mathbf{11}0000 \mid 000000)$, and attended critical features $i = 1$ and $i = 2$, the biasing terms e_1 and e_2 increase to $\frac{5/6}{1+\lambda^{-1}} \equiv e$.

Like fuzzy ARTMAP, biased ARTMAP then activates node $J = 3$ and produces the matched pattern $\mathbf{x} = (\mathbf{111}000 \mid 000000)$. Now, however, the biased matched pattern $\tilde{\mathbf{x}} = ((\mathbf{1}-e),(\mathbf{1}-e),\mathbf{1}000 \mid 000000)$ and the biased input pattern $\tilde{\mathbf{A}} = ((1-e),(1-e),1011 \mid 000100)$. Thus, $\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|} = \frac{3-2e}{6-2e}$, which is less than $\rho = \frac{1}{3}$ when $e = \frac{5/6}{1+\lambda^{-1}} > \frac{3}{4}$, i.e., when $\lambda > 9$. With $\lambda > 9$, mismatch reset keeps node J=3 from making a prediction (–), which would have been correct. Without a predictive error, R=0 and neither vigilance nor any biasing component e_i increases.

Following reset of node J=3, the next active node $J = 1$, with $\mathbf{w}_1 = (\mathbf{000011} \mid 111100)$, produces the matched pattern $\mathbf{x} = (000\mathbf{011} \mid 000100)$, which focuses attention on the presence or absence of the critical features $i = 4$, 5, and 6. Since $e_4 = e_5 = e_6 = 0$, the biased matched pattern $\tilde{\mathbf{x}}$ is the same as $\mathbf{x}$. The biased input $\tilde{\mathbf{A}}$ is still equal to $((1-e),(1-e),1011 \mid 000100)$, and $\rho = \frac{1}{3}$. Thus $\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|} = \frac{3}{6-2e} > \frac{1}{2} > \rho$. Since node J =1 makes a correct output class prediction (–), learning ensues, after which $\mathbf{w}_1 = (000\mathbf{011} \mid 000100)$.

After training on inputs #1-6, the fuzzy ARTMAP weight vectors are:

$\mathbf{w}_1 = (\mathbf{000011} \mid 111100) \rightarrow$ *negative* Input #1

$\mathbf{w}_2 = (\mathbf{11}0\mathbf{1}00 \mid 000000) \rightarrow$ *positive* Inputs #2,3

$\mathbf{w}_3 = (\mathbf{111}000 \mid 000000) \rightarrow$ *negative* Inputs #4,5,6

All three committed nodes focus attention on features $i = 1$ and 2; two nodes attend to features $i = 3$ and 4; and only one node attends to features $i = 5$ and 6.

After training on the same six inputs, the biased ARTMAP $(\lambda > 9)$ weight vectors are:

$\mathbf{w}_1 = (000\mathbf{011} \mid 000100) \rightarrow$ *negative* Inputs #1,6

$\mathbf{w}_2 = (\mathbf{11}0\mathbf{1}00 \mid 000000) \rightarrow$ *positive* Inputs #2,3

$\mathbf{w}_3 = (\mathbf{1111}0\mathbf{0} \mid 000010) \rightarrow$ *negative* Inputs #4,5

Compared to fuzzy ARTMAP, biased ARTMAP pays *less* attention to features $i = 1$, 2, and 3; and *more* attention to features $i = 4$ and 5. It is the redistribution of attention among features

during search that results in biased ARTMAP's correctly learning the training set predictions. Note that this shift of attention is the result of featural biasing that occurred during training, and that no active biasing takes place during testing.

8. Two-dimensional simulations of real-time learning

Simulations now illustrate biased ARTMAP performance on three real-time learning examples, each mapping points in the unit square to output labels *red* or *blue*. Training inputs, selected one at a time at random, are presented once to a fast-learning winner-take-all network with λ=0 (fuzzy ARTMAP) or λ=10 (default biased ARTMAP) or $\lambda \to \infty$ (maximum bias). For each example, a *sparse* training set constitutes the initial training points in a corresponding *dense* training set. Sparse examples thus indicate how much learning of the dense examples occurred early in training.

A *category box overlap* index equals the average number of boxes R_J in which a test point is contained for a given simulation. Comparing category box geometry of the six-point example for fuzzy ARTMAP learning (Figure 5f) with the geometry of biased ARTMAP learning (Figure 8f) leads to the prediction that featural biasing reduces category box overlap. All simulations support this prediction.

8.1. Stripes example

In the stripes example, points are labeled *red* or *blue* in six alternating horizontal stripes (Figure 10). The sparse stripes training set contains 200 points and the dense stripes training set contains 1,000 points. In both the sparse and dense simulations, featural biasing cuts the test error rate almost in half. Setting λ to its default value reduces the error from 11% (λ=0) to 5.7% (λ=10) on the sparse training set and from 3.2% (λ=0) to 1.7% (λ=10) on the dense training set. For these examples, category box overlap of trained fuzzy ARTMAP systems is about 50% greater than category box overlap of corresponding biased ARTMAP systems.

8.2. Checkerboard example

In the checkerboard example, points are labeled as *red* or *blue* on a 6x6 checkerboard (Figure 11). The sparse checkerboard training set contains 500 points and the dense checkerboard training set contains 2,000 points. Given that randomly located points are presented one at a time during training, the checkerboard problem is a challenging example for a real-time fast-learning system. Indeed, after 500 training points, fuzzy ARTMAP test accuracy (53.4%) is barely above chance. At this stage of learning, biased ARTMAP (λ=10) also makes many errors, but its test accuracy (63.3%) is substantially above chance. After training on an additional 1 500 points, fuzzy ARTMAP has recruited more committed nodes (85 vs. 71), and its response profile visibly indicates the 56% greater category box overlap compared to biased ARTMAP. The dense

checkerboard is an exceptional case where the default biased ARTMAP system does not produce near-optimal accuracy, with a chance input ordering producing a relatively large error patch.

8.3. Circle-in-the-square example

In the circle-in-the-square (CIS) example, points inside a circle centered in the square are labeled *blue*, and points outside the circle are labeled *red* (Figure 12). The sparse CIS training set contains 100 points and the dense training set contains 1,000 points. Although biased ARTMAP improves test accuracy by a useful margin, the geometry of the CIS problem tends not to produce elongated category boxes, so this example does not benefit as much from biasing as do most other examples. The relatively small performance improvements are reflected in the relatively small reductions in category box overlap.

8.4. Biased ARTMAP vs. fuzzy ARTMAP performance comparisons

Table 4 summarizes simulation results for the sparse and dense stripes, checkerboard, and circle-in-the-square examples. In every case, biased ARTMAP with its default parameter value (λ=10) or maximum $(\lambda \to \infty)$ parameter value shows improved test accuracy compared to fuzzy ARTMAP (λ=0). In nearly every case biased ARTMAP performance dips slightly as $\lambda \to \infty$, compared to λ=10. Except for examples with the unusual convexity of CIS, fuzzy ARTMAP learning produces category boxes that overlap test points by about 50% more than biased ARTMAP, while the number of category nodes remains approximately constant across systems $(0 \le \lambda \le \infty)$ on a given problem.

For each simulation, Table 4 also shows the number of training points for which featural biasing occurs $(|\mathbf{e}| > 0)$ in response predictive errors. As expected, a larger fraction of training examples make predictive errors early in training, but substantial numbers continue to produce featural biasing as the systems continue to learn to correct predictive errors. For the sparse checkerboard, for example, biased ARTMAP (λ=10) errors trigger featural biasing on 46% of the first 500 training examples (sparse). Of the next 1 500 examples (dense), 25% make predictive errors, thus increasing e_i values during search.

Additional simulations that add noise to each of the examples of Table 4 show similar performance patterns. These studies provide additional clues to how biased ARTMAP consistently improves test prediction accuracy compared to fuzzy ARTMAP on real-time fast-learning problems. In one such study, *red* / *blue* labels were switched on 10% of randomly chosen training points. Without knowing that some training labels are incorrect, fuzzy ARTMAP tends to create inappropriately large and overlapping category boxes early in training, as in the six-point example (Figure 5f). In response to the same training set, even with fast learning, featural biasing produces more resets in response to mislabeled inputs, which leads biased ARTMAP to create point boxes rather than expanded boxes that tend to obliterate prior learning. These studies indicate that biasing helps to limit the damage of training set noise or outliers.

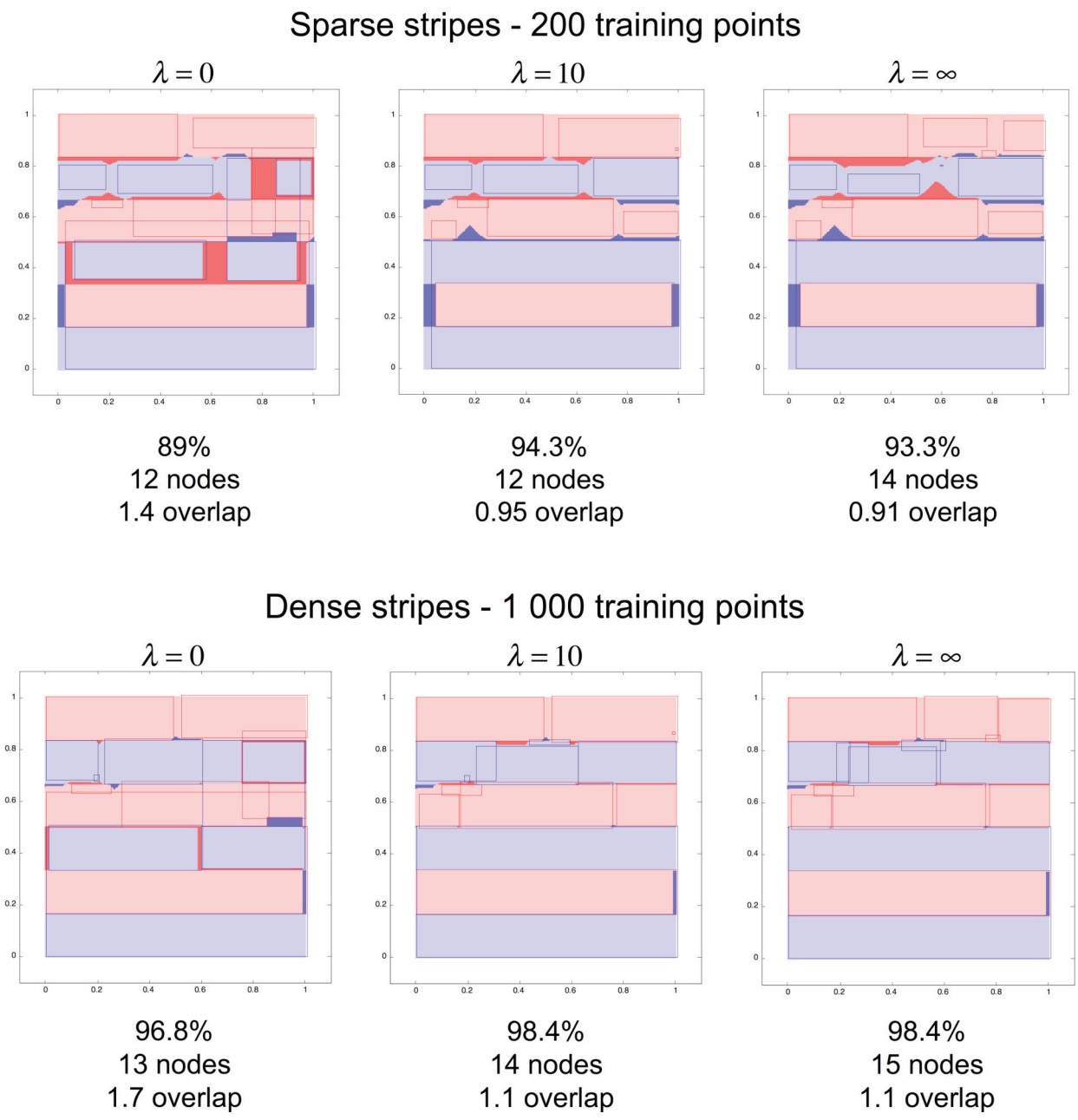

Figure 10. Stripes simulations. Figures 10-12 show predictions for a 250x250 grid of test points in the unit square. Light red and blue indicate correct *red* and *blue* predictions, and dark colors indicate errors.

Figure 11. Checkerboard simulations.

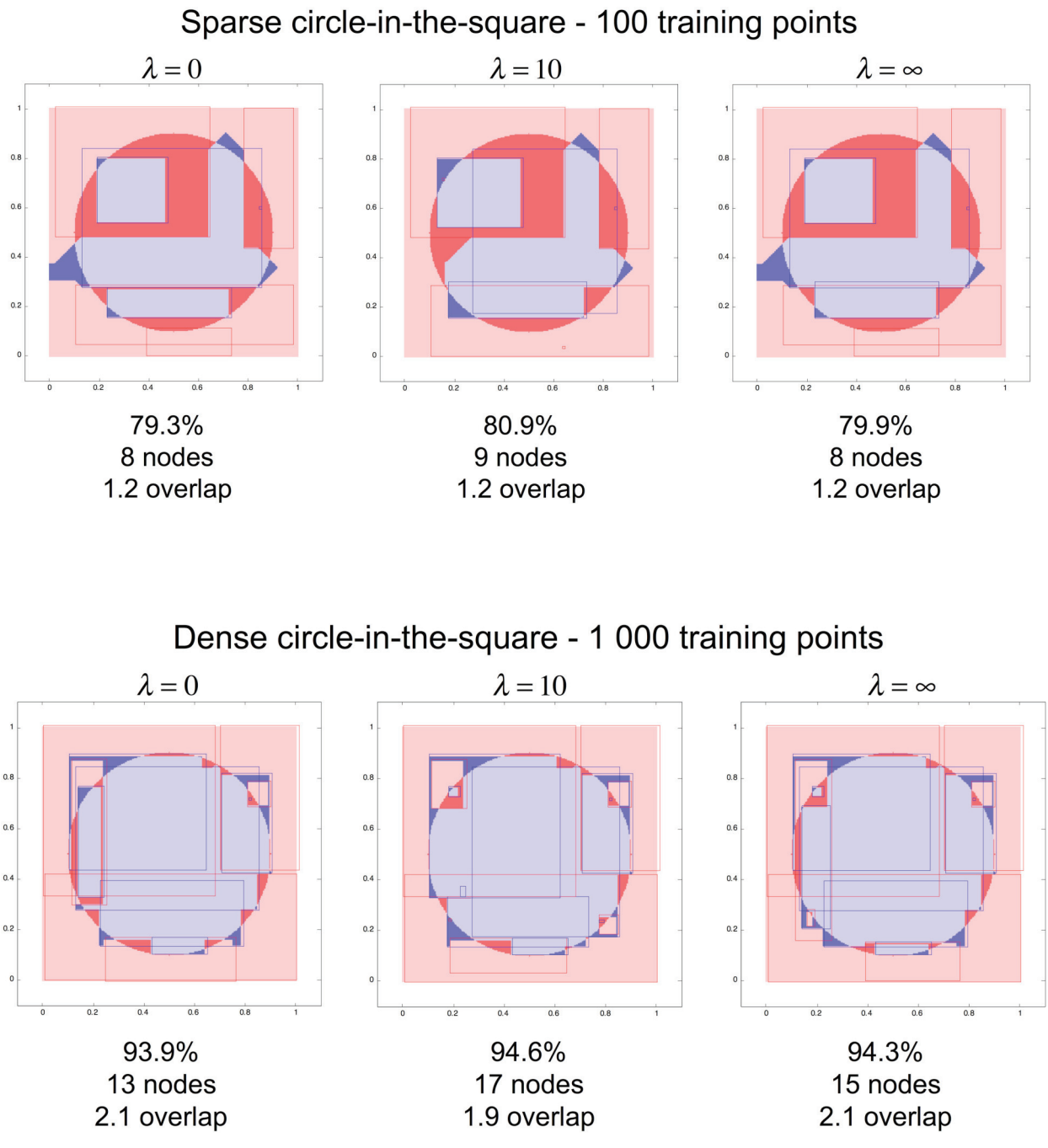

Figure 12. Circle-in-the-square simulations.

Table 4
Two-dimensional simulation comparisons of fuzzy ARTMAP (λ=0) and biased ARTMAP (λ=10 and $\lambda \to \infty$), as illustrated in Figures 10-12.

Example	λ	# train	# train |e| > 0	% correct	R_J overlap average	# nodes
Stripes sparse	0	200	0	89.0	1.4	12
	10		29	94.3	0.95	12
	∞		32	93.3	0.91	14
Stripes dense	0	1,000	0	96.8	1.7	13
	10		53	98.4	1.1	14
	∞		54	98.4	1.1	15
Checkerboard sparse	0	500	0	53.4	4.1	35
	10		230	63.3	3.2	38
	∞		240	60.7	3.5	50
Checkerboard dense	0	2,000	0	77.8	6.4	85
	10		611	78.2	4.1	71
	∞		632	81.4	4.5	84
CIS sparse	0	100	0	79.3	1.2	8
	10		26	80.9	1.2	9
	∞		26	79.9	1.2	8
CIS dense	0	1,000	0	93.9	2.1	13
	10		94	94.6	1.9	17
	∞		105	94.3	2.1	15

9. Large-scale simulations of real-time learning

Two simulation testbeds, the Boston remote sensing example and a new movie genre prediction example, now show that the introduction of featural bias improves performance on large-scale, as well as two-dimensional, benchmark problems.

9.1. Boston remote sensing example

The Boston testbed was developed as a challenging benchmark to assess performance of a variety of learning systems. The labeled dataset is available from the CNS Technology Lab Website (http://techlab.bu.edu/bART/). A canonical learning procedure (Parsons & Carpenter, 2003) divides the image into four vertical strips: two for training, one for validation (if needed), and one for testing (Figure 13). This protocol produces geographically distinct training and testing areas, to assess regional generalization. Typically, class distributions vary substantially across strips. For example, strip #4 contains a large fraction of *ocean* pixels, which are completely absent in strip #1.

Each Boston image pixel is described by 41 feature values: six Landsat 7 Thematic Mapper (TM) bands at 30*m* resolution; two thermal bands at 60*m* resolution; one panchromatic band with 15*m* resolution; and 32 derived bands representing local contrast, color, and texture. In the Boston dataset, each of 29,003 ground truth pixels is labeled as belonging to one of eight classes (*beach, ocean, ice, river, road, park, residential, industrial*). Biased ARTMAP simulations did not require a validation strip, so systems are trained on ground truth pixels from three strips and tested on the fourth. Each strip serves, in turn, as a test set for an independently trained network.

Accuracy of a class prediction is assessed using a measure that is independent of the mix of classes in the test set. In this way, accuracy is not artificially inflated by large numbers of "easy" pixels in a test region as, for example, *ocean* pixels in strip #4. For each class, the predictive accuracy on test pixels actually in that class is recorded. If more samples of this class were added to the test set, the fraction of correct predictions would remain approximately constant. Overall predictive accuracy of a system is taken to be the average accuracy across the eight output classes.

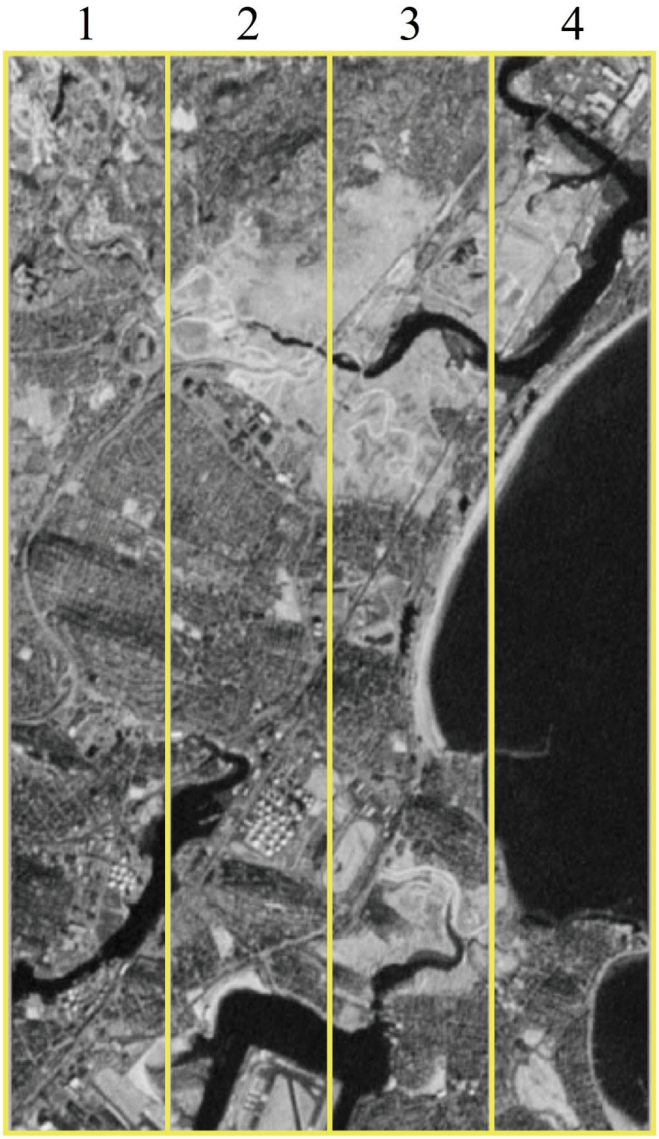

Figure 13. The Boston testbed image (Carpenter, Martens, & Ogas, 2005). The city of Revere is at the center, surrounded by (clockwise from lower right) portions of Winthrop, East Boston Chelsea, Everett, Malden, Melrose, Saugus, and Lynn. Logan Airport runways and Boston Harbor are at the lower center, with Revere Beach and the Atlantic Ocean at the right. The Saugus and Pines Rivers meet in the upper right, and the Chelsea River is in the lower left of the image. Vertical strips #1-4 define disjoint training, validation, and test regions. Dimensions: 360x600 pixels (15*m* resolution) == 5.4x9*km*.

Table 5 shows overall class prediction accuracies for each of the four strips, following one epoch of training on the other three strips by fuzzy ARTMAP (λ=0), and by biased ARTMAP with its default parameter setting (λ=10) and with $\lambda \to \infty$. Boston testbed results show the same performance improvement patterns as seen in the two-dimensional simulation examples. Compared to fuzzy ARTMAP, biased ARTMAP improves performance on every strip, in most cases by a substantial margin. For example, biased ARTMAP reduces the error rate from 8.0% to 4.4% on strip #1, and from 16.9% to 11.9% on strip #4. As $\lambda \to \infty$, biased ARTMAP accuracy remains the same as or slightly below that of the default value (λ=10), again following the pattern of the two-dimensional examples. Neither are Boston testbed performance improvements accomplished at the expense of additional memory load: each simulation produces 13, 14, or 15 committed nodes across the four strips and for λ = 0, 10, or ∞.

Table 5
Boston remote sensing example.

Boston remote sensing example: Test				
	Test strip #1	**Test strip #2**	**Test strip #3**	**Test strip #4**
λ	**Class % accuracy**			
0	92.0	81.0	90.6	83.1
10	95.6	85.2	91.0	88.1
∞	95.6	85.2	90.9	87.3

9.2. Movie genre example

The Netflix® Prize is a competition to improve movie recommendation accuracy (Bennet & Lanning, 2007). Rules of the competition are available at http://www.netflixprize.com/index. Netflix provides a dataset containing 100,480,507 ratings by 480,189 users for 17,770 movies. Movie recommendation engines trained on these data are tested on an undisclosed data set to gauge prediction success.

To create a biased ARTMAP benchmark example, the Netflix dataset was augmented with a primary genre label for each of the 17,770 movies. Genre labels were obtained from the movie synopses on the Netflix website. The term used for such data collection is *crawling*. Movies are classified as belonging to one of 21 genres (Table 6).

The Netflix dataset forms a sparse ratings matrix of size 17,770 movies x 480,189 users, with only 100,480,507 (0.01%) of the matrix elements populated by ratings. While the prize competition seeks to predict the missing ratings in this matrix, this project uses the enhanced dataset to define a different problem: predicting primary movie genres from ratings data.

Table 6
Primary Netflix movie genres. The seven genres shown in italics are infrequent, accounting for only 980 of the 17,770 movies.

Netflix movie genres		
!!Uncensored	Drama	*NA*
Action&Adventure	*Faith&Spirituality*	Romance
Anime&Animation	Foreign	Sci-Fi&Fantasy
Children&Family	*Gay&Lesbian*	*SpecialInterest*
Classics	Horror	*Sports&Fitness*
Comedy	*Independent*	Television
Documentary	Music&Musicals	Thrillers

9.2.1. Movie feature derivation

In the absence of directly interpretable features, most Netflix prediction algorithms rely on user and/or movie similarities to generate predictions. The broad class of such approaches is termed *collaborative filtering*. Informative features describing movies and users may be obtained indirectly by factoring the ratings matrix into a user matrix and a movie matrix, where users and movies correspond to rows and columns in the matrices resulting from decomposition. Matrix factorization minimizes the difference between the populated elements of a reconstructed matrix and the actual ratings matrix. An incremental singular value decomposition (SVD) factorization algorithm uses error minimization to produce user and movie matrices. This preprocessing technique is a minor variant of the technique introduced to the Netflix Prize competition by Funk (2006).

The incremental SVD algorithm produces a 64-dimensional feature vector for each movie. The 17,770 movie feature vectors and their genre labels constitute the *genre-augmented dataset*. Omitting movies with genres that appear infrequently produces a dataset with 16,840 movies, each classified as belonging to one of 14 genres (Table 7)

Table 7
Primary genres of 16,840 movies in the genre-augmented dataset.

Benchmark movie genres		
Action&Adventure	Documentary	Romance
Anime&Animation	Drama	Sci-Fi&Fantasy
Children&Family	Foreign	Television
Classics	Horror	Thrillers
Comedy	Music&Musicals	

9.2.2. Genre prediction by fuzzy ARTMAP and biased ARTMAP

Each of the 64 movie feature vector components was linearly scaled to the [0,1] range. The dataset was partitioned into a training set of 14,314 movies and a test set of 2,526 movies. This dataset is available from http://techlab.bu.edu/bART/.

Performance of Netflix genre prediction was assessed using the same class-based accuracy measure as for the Boston testbed, so that evaluation is independent of particular mix of genre exemplars in the test set. For each genre, predictive accuracy on test set movies actually in that genre was recorded, and the pattern of errors recorded in one row of a confusion matrix. Overall

predictive accuracy is the average accuracy across the 14 genre classes. With this method, if genres were assigned to movies randomly, prediction accuracy would equal 7.1%.

After training for one epoch, fuzzy ARTMAP (λ=0) produces a test genre prediction accuracy of 39.6%, with biased ARTMAP (λ=10) improving genre prediction accuracy to 44.4%. While still making many erroneous primary genre predictions, both systems performed well above chance. Moreover, erroneous predictions were "sensible" in that, although each movie was given only one genre label, many actually belong to multiple genres. For example, the movie *The Seventh Seal* might be labeled *Foreign* in the dataset, while its genres are also listed as *Drama* and *Classic* on the Netflix users' website. In fact, of the 288 movies with the primary genre label *Foreign*, biased ARTMAP correctly labeled 44% *Foreign*, while labeling 12% each *Comedy* or *Drama*, and labeling another 6% *Classics*. These error patterns suggest an expansion of the genre problem to multi-class predictions, as discussed in Section 10.3.

10. Discussion

This section considers points of possible concern raised by detailed consideration of biased ART dynamics.

10.1. Bias update does not disrupt orderly search

Bias update begins when the reset signal r switches to 1, at the moment when ρ rises above $\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|}$ during match tracking. Bias update will continue, however, only so long as r remains equal to 1, which in turn depends upon $\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|}$ remaining less than ρ as bias terms e_i increase. If this were not the case, bias update *per se* could shut off reset, leading to an alternating cycle of increasing ρ and increasing e_i. In fact, $\frac{\partial}{\partial e_i}\left(\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|}\right) = 0$ for Cases 1 and 2 (Section 5.8). For Case 3, $\frac{\partial}{\partial e_i}|\tilde{\mathbf{x}}| = \frac{\partial}{\partial e_i}|\tilde{\mathbf{A}}| = -1$, so $\frac{\partial}{\partial e_i}\left(\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|}\right) = \frac{-\left(|\tilde{\mathbf{A}}| - |\tilde{\mathbf{x}}|\right)}{|\tilde{\mathbf{A}}|^2} \le 0$. Thus $\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|}$ remains less than ρ as bias terms increase, and the real-time bARTMAP network performs as described by the algorithm and as illustrated by Figure 7.

Similarly, $|\mathbf{e}|$ is always less than M and $|\tilde{\mathbf{A}}| > 0$ for all inputs $\mathbf{A}$.

10.2. Biasing may produce Next Input Test failures

Match tracking implies that fuzzy ARTMAP satisfies the Next Input Test: after a fast-learning trial, an input that is re-presented is guaranteed to make the correct prediction without search (Section 2.2). Inputs trained with biased ARTMAP may, however, fail the Next Input Test. In the checkerboard example, for instance, this failure occurs for certain inputs that lie in the intersection of long thin perpendicular category boxes. In this case, it is possible that **e** may cause $\tilde{\mathbf{A}}$ to decrease more than $\tilde{\mathbf{x}}$, and that $\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|} \geq \rho > \frac{|\mathbf{x}|}{|\mathbf{A}|}$ during search.

Some such cases produce occasional Next Input Test failures on training exemplars, even though overall test performance is still better. The five-point example (Table 8) shows how biased ARTMAP may sometimes fail to learn a correct training point prediction. The example also indicates why such failures tend not to harm performance, and may even help, by treating some noisy training inputs as outliers.

In Figure 14a, in response to input #5 where **a** = (0.9, 0.85), fuzzy ARTMAP creates the point box R_3. This point passes the Next Input Test: if re-presented, **a** will choose node $J = 3$ and correctly predict the output class *red*. However, this point box does not influence the classification decisions of any surrounding points, which all predict *blue*.

Table 8
The five-point example illustrates how biased ARTMAP may fail the Next Input Test.

Five-point example		
Input #	**Input a**	**Output class**
1	(1,0)	*red*
2	(1, 1)	*red*
3	(0.2,0.8)	*blue*
4	(1,0.9)	*blue*
5	(0.9,0.85)	*red*

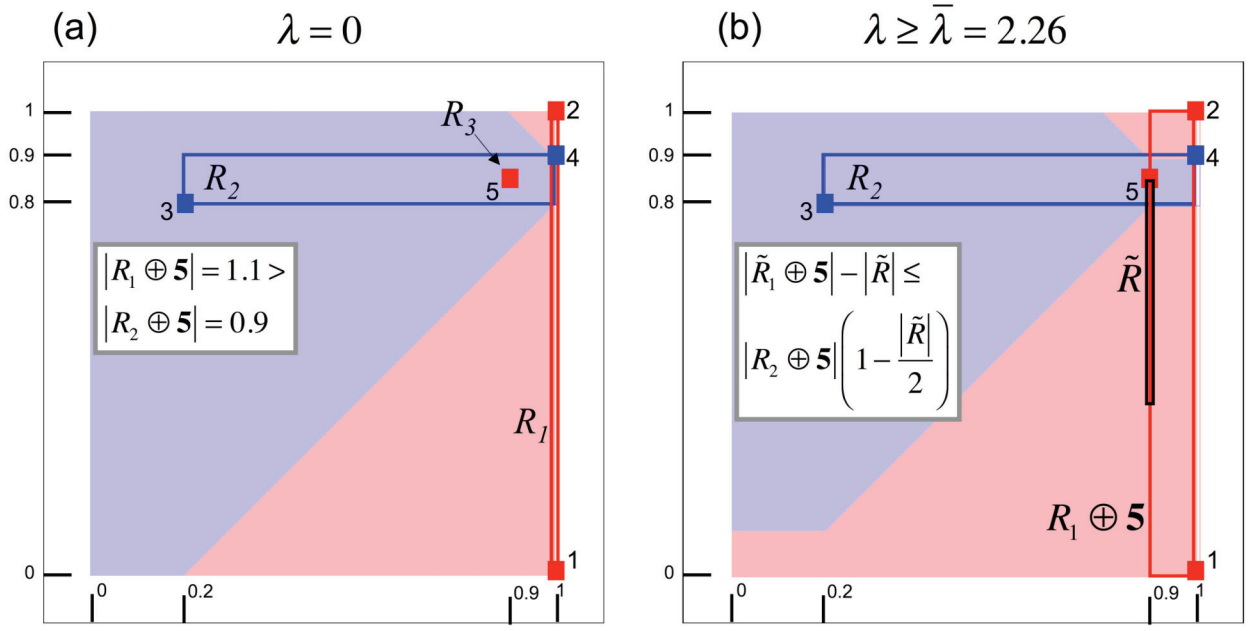

Figure 14. Five-point example test predictions. (a) Fuzzy ARTMAP commits three category nodes, and the five training inputs correctly predict their output classes during testing. (b) Biased ARTMAP commits only two category nodes, which leads to more test-set *red* labels, even though training input #5 incorrectly predicts *blue* during testing.

Biased ARTMAP fails the Next Input Test at input point #5 when $e_2 = |\tilde{R}| > 4/11$, i.e., when $\lambda \geq \bar{\lambda} = 2.26$ (Figure 14b). When input #5 chooses node J=2, e_2 increases, producing a bias against the feature i=2. As e_2 increases, both $|\tilde{\mathbf{x}}|$ and $|\tilde{\mathbf{A}}|$ decrease, and $\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|} \leq \frac{|\mathbf{x}|}{|\mathbf{A}|} < \rho$.

However, when input #5 then chooses node J=1, $\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|} > \frac{|\mathbf{x}|}{|\mathbf{A}|}$. Because category box R_1 is elongated in the direction of i =2, node J=1 is indifferent to the fact that e_2>0. Thus after bARTMAP resets J=2 and chooses J=1, $|\tilde{\mathbf{x}}| = |\mathbf{x}|$, i.e., $|\tilde{R}_1 \oplus \mathbf{5}| = |R_1 \oplus \mathbf{5}|$. On the other hand, the bias does make $|\tilde{\mathbf{A}}| < |\mathbf{A}|$, so that $\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|} > \frac{|\mathbf{x}|}{|\mathbf{A}|}$. If ρ lies between these two values, biased ARTMAP will not reset, while fuzzy ARTMAP does reset node J=1. The five-point example is constructed to demonstrate that this possibility may, in fact, occur.

With biased ARTMAP, when node J = 1 passes the vigilance test the category box R_1 expands to include input #5. If this **a** were re-presented, it would choose node J = 2, since R_2 is smaller

than R_1, and hence incorrectly predict *blue*. Despite this failure, biased ARTMAP learning in response to **a** paradoxically expands the set of more distant test points predicting *red*, compared to the response profile of fuzzy ARTMAP (Figure 14).

If passing the Next Input Test on all training exemplars were a necessary condition for a given problem, bARTMAP could be modified to meet this constraint. For example, the system could be defined to reset if either $\frac{|\tilde{\mathbf{x}}|}{|\tilde{\mathbf{A}}|} < \rho$ or $\frac{|\mathbf{x}|}{|\mathbf{A}|} < \rho$, which would produce more bARTMAP resets. However, all examples considered so far indicate that this additional condition is not necessary, and that requiring that a system strictly pass the Next Input Test on 100% of training trials may even worsen test performance.

10.3. Multi-class predictions

The Netflix movie genre prediction example trains each movie with just one genre label, and a system's accuracy is evaluated in terms of its success in predicting primary genre labels of test set movies. Confusion matrices that indicate reasonable predictions of secondary genre labels suggest augmenting the benchmark problem to consider both primary and secondary genre predictions. Assessment of such predictions would require additional information from Netflix or other sources. ARTMAP systems, which are designed to learn one-to-many maps, as well as many-to-one maps, are intrinsically suited to such multi-class prediction tasks and to discovering rule-like relationships among classes (Carpenter, Martens, & Ogas, 2005).

11. Conclusions

The new biasing mechanism introduced here allows a learning system to redistribute attention among features active in a top-down / bottom-up matched pattern when the system detects an error. While the featural biasing step could, in principle, readily augment any system, the central role of top-down / bottom-up interactions is virtually a defining characteristic of ART networks. In order to highlight its specific contributions, biased ARTMAP dynamics and performance have here been closely compared with those of fuzzy ARTMAP. Fuzzy ARTMAP has, in turn, been widely used in technological applications and in other comparative studies. While computational examples demonstrate how featural biasing in response to predictive errors improves performance on supervised learning tasks, the error signal that gates biasing could also originate from other sources, as in reinforcement learning.

Learning systems may be presented with labeled input patterns only once. Examples show that, while ARTMAP attention to early critical features may later distort learned memory representations, biased ARTMAP corrects this distortion by focusing on previously unattended features when the system detects an error. Both first principles and consistent performance improvements on a variety of simulation studies suggest that featural biasing should be incorporated by default in all ARTMAP systems.

References

Bennet. J., & Lanning, S. (2007). The Netflix Prize. http://www.cs.uic.edu/~liub/KDD-cup-2007/NetflixPrize-description.pdf .

Carpenter, G.A. (1994). A distributed outstar network for spatial pattern learning. *Neural Networks*, *7*, 159-168. http://techlab.bu.edu/members/gail/articles/083_dOutstar_1994_.pdf

Carpenter G. A. (1997). Distributed learning, recognition, and prediction by ART and ARTMAP neural networks. *Neural Networks*, *10*, 1473–1494. http://techlab.bu.edu/members/gail/articles/115_dART_NN_1997_.pdf

Carpenter, G. A. (2003). Default ARTMAP. *Proceedings of the international joint conference on neural networks (IJCNN'03)*, 1396–1401. http://techlab.bu.edu/members.gail/articles/142_Default_ARTMAP_2003_.pdf

Carpenter, G.A., & Gjaja, M.N. (1994). Fuzzy ART choice functions. *Proceedings of the World Congress on Neural Networks (WCNN-94)*, Hillsdale, NJ: Lawrence Erlbaum Associates, *I*-713-722. http://techlab.bu.edu/files/resources/articles_cns/CarpenterGjaja1994.pdf

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. http://techlab.bu.edu/members/gail/articles/026_ART_1_CVGIP_1987.pdf

Carpenter, G. A., & Grossberg, S. (1990). ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, *4*, 129-152. http://techlab.bu.edu/members/gail/articles/042_ART_3_1990_.pdf

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. http://techlab.bu.edu/members/gail/articles/070_Fuzzy_ARTMAP_1992_.pdf

Carpenter G. A., Grossberg S., & Reynolds J. H. (1991). ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, *4*, 565–588. http://techlab.bu.edu/members/gail/articles/054_ARTMAP_1991_.pdf

Carpenter, G. A., Grossberg S., & Rosen D. B. (1991). Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, *4*, 759–771. http://techlab.bu.edu/members/gail/articles/056_Fuzzy_ART_1991_.pdf

Carpenter, G.A., & N. Markuzon, N. (1998). ARTMAP-IC and medical diagnosis: Instance counting and inconsistent cases. *Neural Networks*, *11*, 323-336. http://techlab.bu.edu/members/gail/articles/117_ARTMAP-IC_1998_.pdf

Carpenter, G.A., Martens, S., & Ogas, O.J. (2005). Self-organizing information fusion and hierarchical knowledge discovery: a new framework using ARTMAP neural networks. *Neural Networks*, *18*, 287-295. http://techlab.bu.edu/members/gail/articles/148_2005_InfoFusion_CarpenterMartensOgas_.pdf

Carpenter, G.A., Milenova, B.L., & Noeske, B.W. (1998). Distributed ARTMAP: a neural network for fast distributed supervised learning. *Neural Networks, 11*, 793-813. http://techlab.bu.edu/members/gail/articles/120_dARTMAP_1998_.pdf

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, 1995. http://techlab.bu.edu/members/gail/articles/097_ART-EMAP_1995_.pdf

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. http://portal.acm.org/citation.cfm?id=195890

Funk, S. (2006). Netflix update: Try this at home. http://sifter.org/~simon/journal/20061211.html .

Grossberg, S. (1999). The link between brain, learning, attention, and consciousness. *Consciousness and Cognition*, *8*, 1-44. http://www.cns.bu.edu/Profiles/Grossberg/Gro1999ConCog.pdf

Grossberg, S. (2003). How does the cerebral cortex work? Development, learning, attention, and 3D vision by laminar circuits of visual cortex. *Behavioral and Cognitive Neuroscience Reviews*, *2*, 47-76. http://www.cns.bu.edu/Profiles/Grossberg/Gro2003BCNR.pdf

Hurvich, L. M., & Jameson, D. (1957). An opponent-process theory of color vision. *Psychological Review*, *64*, 384- 390.

Lisboa, P. (2001). Industrial use of saftey-related artificial neural netoworks. *Contract research report 327/2001,* Liverpool: John Moores University.

Parsons, O., & Carpenter, G. A. (2003). ARTMAP neural networks for information fusion and data mining: map production and target recognition methodologies. *Neural Networks*, *16*, 1075-1089. http://techlab.bu.edu/members/gail/articles/143_ARTMAP_map_NN_2003_.pdf

Schiller, P.H. (1982). Central connections of the retinal ON and OFF pathways. *Nature*, *297*, 580-583. http://www.nature.com/nature/journal/v297/n5867/abs/297580a0.html

Williamson, J.R. (1996). Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps. *Neural Networks*, *9*(5), 881-897. http://techlab.bu.edu/files/resources/articles_tt/Gaussian%20ARTMAP,%20A%20neural%20network%20for%20past%20incremental%20learning%20of%20noisy%20multidimensional%20maps.pdf