Self-Supervised ARTMAP

Gregory P. Amis and Gail A. Carpenter

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

[gamis, gail] @cns.bu.edu

May, 2009

Neural Networks, in press.

Technical Report CAS/CNS TR-2009-006
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 CELEST, an NSF Science of Learning Center (SBE-0354378).

Self-Supervised ARTMAP

Abstract: Computational models of learning typically train on labeled input patterns (supervised learning), unlabeled input patterns (unsupervised learning), or a combination of the two (semi-supervised learning). In each case input patterns have a fixed number of features throughout training and testing. Human and machine learning contexts present additional opportunities for expanding incomplete knowledge from formal training, via self-directed learning that incorporates features not previously experienced. This article defines a new self-supervised learning paradigm to address these richer learning contexts, introducing a neural network called self-supervised ARTMAP. Self-supervised learning integrates knowledge from a teacher (labeled patterns with some features), knowledge from the environment (unlabeled patterns with more features), and knowledge from internal model activation (self-labeled patterns). Self-supervised ARTMAP learns about novel features from unlabeled patterns without destroying partial knowledge previously acquired from labeled patterns. A category selection function bases system predictions on known features, and distributed network activation scales unlabeled learning to prediction confidence. Slow distributed learning on unlabeled patterns focuses on novel features and confident predictions, defining classification boundaries that were ambiguous in the labeled patterns. Self-supervised ARTMAP improves test accuracy on illustrative low-dimensional problems and on high-dimensional benchmarks. Model code and benchmark data are available from: http://techlab.bu.edu/SSART/.

Keywords: Self-supervised learning, supervised learning, Adaptive Resonance Theory (ART), ARTMAP, unsupervised learning, machine learning

1. Supervised, unsupervised, semi-supervised, and self-supervised learning

Computational models of supervised pattern recognition typically utilize two learning phases. During an initial training phase, input patterns, described as values of a fixed set of features, are presented along with output class labels or patterns. During a subsequent testing phase, the model generates output predictions for unlabeled inputs, while no further learning takes place.

Although the supervised learning paradigm has been successfully applied for a wide variety of applications, it does not reflect many natural learning situations. Humans do learn from explicit training, as from a textbook or a teacher, and they do take tests. However, students do not stop learning when they leave the classroom. Rather, they continue to learn from experience, incorporating not only more information but new types of information, all the while building on their earlier classroom knowledge. The self-supervised learning system introduced here models such life-long experiences.

An *unsupervised* learning system clusters unlabeled input patterns. *Semi-supervised* systems (Chapelle, Schölkopf, & Zien, 2006) learn from unlabeled as well as labeled inputs during training. The *self-supervised* paradigm models two learning stages. During Stage 1 learning, the system receives all output labels, but only a subset of possible feature values for each input. During Stage 2 learning, the system receives all feature values for each input, but no output

labels (Table 1).

Table 1

Learning paradigms

Paradigm		Learning Training set: Input features	Learning Training set: Output labels	No learning Test set
Supervised		All	All	All input features / All output labels
Semi-supervised	Stage 1	All	All	
	Stage 2	All	None	
Self-supervised *New definition*	Stage 1	Some	All	
	Stage 2	All	None	
Unsupervised		All	None	

Preparing students for a lifetime of independent learning has always been a primary principle of good education. As described by Stephen Krashen, an expert in second language acquisition:

> An autonomous acquirer is not a perfect speaker of the second language, just good enough to continue to improve without us. This is, of course, the goal of all education – not to produce masters but to allow people to begin work in their profession and to continue to grow. (Krashen, 2004, p. 6)

Self-supervised learning models the continuing growth of the autonomous acquirer. By analogy with Stage 1 learning, a medical student may, for example, initially learn to treat diseases from case studies with clear diagnoses for a small sample of patients using a limited number of specified features such as major symptoms and test results. Later in practice (Stage 2), the doctor, no longer supervised, diagnoses patients with myriad symptoms, test results, and contexts that were not fully specified during training. The doctor not only successfully treats under these enhanced circumstances but also learns from the experiences.

In addition to modeling the human learning experience, self-supervised learning promises to be useful for technological applications such as web page classification. A supervised learning system that completes all training before making test predictions does not adapt to new information and varying contexts. An adaptive system that continues to learn unsupervised during testing risks degrading its supervised knowledge. The self-supervised learning system

developed here continues to learn from new experiences without, in most cases, degrading performance.

2. A computational example of self-supervised learning

A simplified medical diagnosis example (Figure 1) illustrates the dynamics of the self-supervised learning model. The problem is to identify exemplars as belonging to the output class *normal* or *hypothermic* or *septic*, given the input feature values *temperature* and *shock*. During Stage 1 learning, class labels are paired with inputs that specify only the feature *temperature* (Figure 1a). The system models incremental online learning that addresses just one case at a time. Projected onto the *temperature* axis, classes overlap, and, even with a large training set, optimal test accuracy is 65%. Although extreme temperatures unambiguously indicate *hypothermic* or *septic*, these class labels are densely mixed with *normal* at moderately low and moderately high temperatures.

During Stage 2 self-supervised learning, inputs specify both *temperature* and *shock*, but no class labels are provided (Figure 1b). The self-supervised learning system has an opportunity to incorporate *shock* information, but needs to be designed in such a way that it does not degrade the partial knowledge that it gained during Stage 1 learning.

Figure 1c illustrates results of a typical self-supervised learning simulation. Following Stage 1, where only one input feature (*temperature*) was available, the model attains a near-optimal test accuracy of 65%. During slow self-supervised Stage 2 learning on unlabeled (but fully featured) inputs, the model adds novel *shock* information to Stage 1 *temperature* knowledge. After 5,000 Stage 2 learning samples, the test set is labeled with 100% accuracy.

3. ART and ARTMAP

The self-supervised learning system of Figure 1 is based on Adaptive Resonance Theory (ART). ART neural networks model real-time prediction, search, learning, and recognition. Design principles derived from scientific analyses and design constraints imposed by targeted applications have jointly guided the development of many variants of the basic supervised learning ARTMAP networks, including fuzzy ARTMAP (Carpenter, Grossberg, Markuzon, Reynolds, & Rosen, 1992), ARTMAP-IC (Carpenter & Markuzon, 1998), and Gaussian ARTMAP (Williamson, 1998). A defining characteristic of various ARTMAP systems is the nature of their internal code representations. Early ARTMAP networks, including fuzzy ARTMAP, employed *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 mapped to its designated output class.

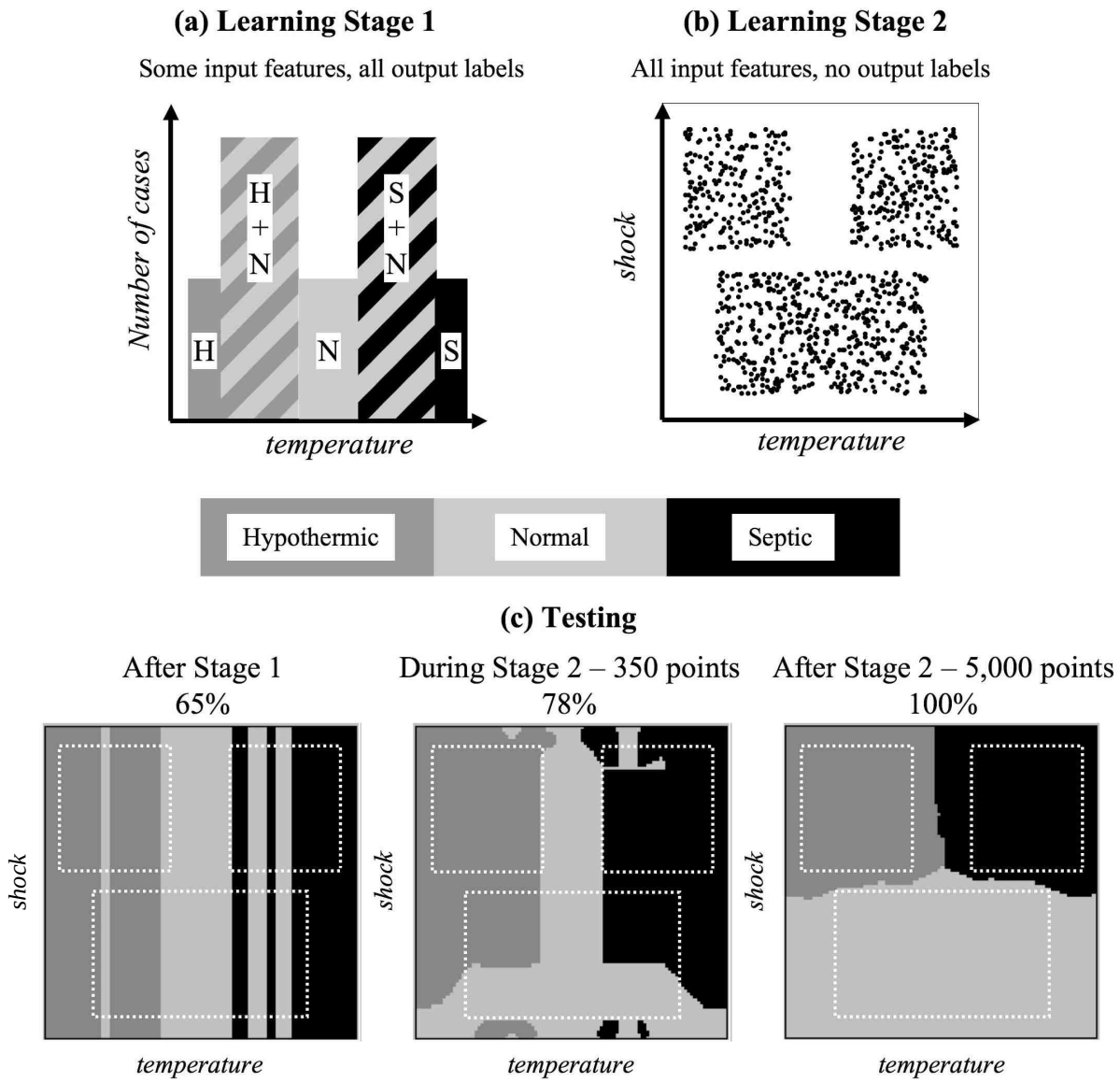


Figure 1. Simplified medical diagnosis example. (a) During Stage 1 self-supervised learning, each input, specified only by its *temperature* value, is paired with one of three output classes. (b) During Stage 2 learning, both input features but no class labels are specified. (c) Test predictions during a typical self-supervised model simulation trial. During Stage 1, test accuracy reaches an optimal 65% after 28 training points. Further learning attempts to correct errors in regions of overlap but does not improve test accuracy. The Stage 1 figure is the result of training on 50 points. Performance steadily improves during Stage 2 self-supervised learning, with most system updates occurring on unambiguous inputs, either at extreme temperatures, which clearly indicate *hypothermic* or *septic*, or at moderate temperatures, which indicate *normal*.

Starting with ART-EMAP (Carpenter & Ross, 1995), ARTMAP systems have used distributed coding during testing, which typically improves predictive accuracy while avoiding the computational challenges inherent in the use of distributed code representations during training. In order meet these challenges, distributed ARTMAP (Carpenter, 1997; Carpenter, Milenova, & Noeske, 1998) introduces a new network configuration and new learning laws that realize stable fast learning with distributed coding during both training and testing.

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; Amis & Carpenter, 2007), which features simplicity of design and robust performance in many application domains (http://techlab.bu.edu/). Default ARTMAP employs winner-take-all coding during training and distributed coding during testing within a distributed ARTMAP network architecture. During Stage 1, when the system can incorporate externally specified output labels with confidence, self-supervised ARTMAP (SS ARTMAP) employs winner-take-all coding and fast learning. During Stage 2, when the network internally generates its own output labels, codes are distributed and learning is slow, so that incorrect hypotheses do not abruptly override Stage 1 "classroom learning."

Sections 3.1 – 3.4 introduce the key ARTMAP computations that are implemented in the SS ARTMAP architecture.

3.1. Complement coding: Learning both absent and present features

Unsupervised ART and supervised ARTMAP networks employ a preprocessing step called *complement coding* (Carpenter, Grossberg, & Rosen, 1991). When the learning system is presented with a set of input features $\mathbf{a} \equiv (a_1 ... a_i ... a_M)$, complement coding doubles the number of input components, presenting to the network both the original feature vector and its complement.

Complement coding allows a learning system to encode 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 being learned become regarded as uninformative with respect to that category.

To implement complement coding, component activities a_i of an *M*-dimensional 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 $(1 - a_i)$. Both **a** and its complement $\mathbf{a}^c$ are concatenated to form the 2*M*-dimensional system input vector $\mathbf{A} = (\mathbf{a} \mid \mathbf{a}^c)$ (Figure 2). Subsequent network computations operate in this 2*M*-dimensional input space, which may be interpreted as representing ON-channel and OFF-channel dynamics.

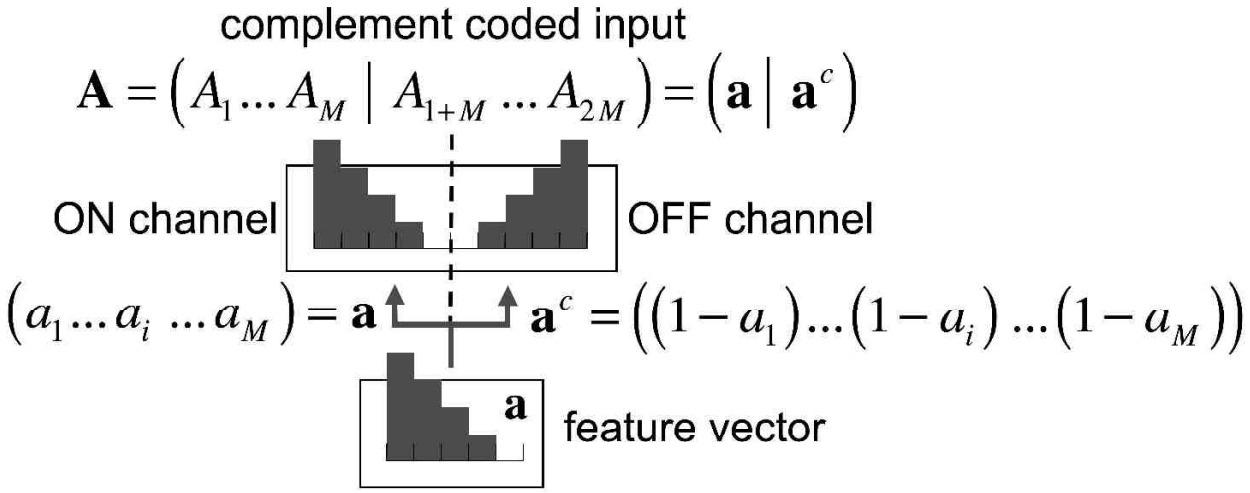


Figure 2. Complement coding transforms an *M*-dimensional feature vector **a** into a 2*M*-dimensional system input vector **A**. A complement-coded system input represents both the degree to which a feature *i* is present (a_i) and the degree to which that feature is absent $(1-a_i)$.

3.2. ARTMAP search and match tracking

Matching bottom-up inputs against top-down expectations is a fundamental computation of all ART networks. The ART matching process triggers either learning or a parallel memory search (Figure 3). If search ends at an established code, the memory representation may either remain the same or delete mismatched portions of the current input. While this dynamic applies to arbitrarily distributed activation patterns, the F_2 code will be described here as a single category node in a winner-take-all system.

Before ARTMAP makes a class prediction, the bottom-up input **A** is matched against the top-down learned expectation that is read out by the active node (Figure 3b). 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 an input arrives, vigilance equals a baseline level $\bar{\rho}$. Baseline vigilance is set equal to zero by default, in order to maximize generalization. Vigilance rises only 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.

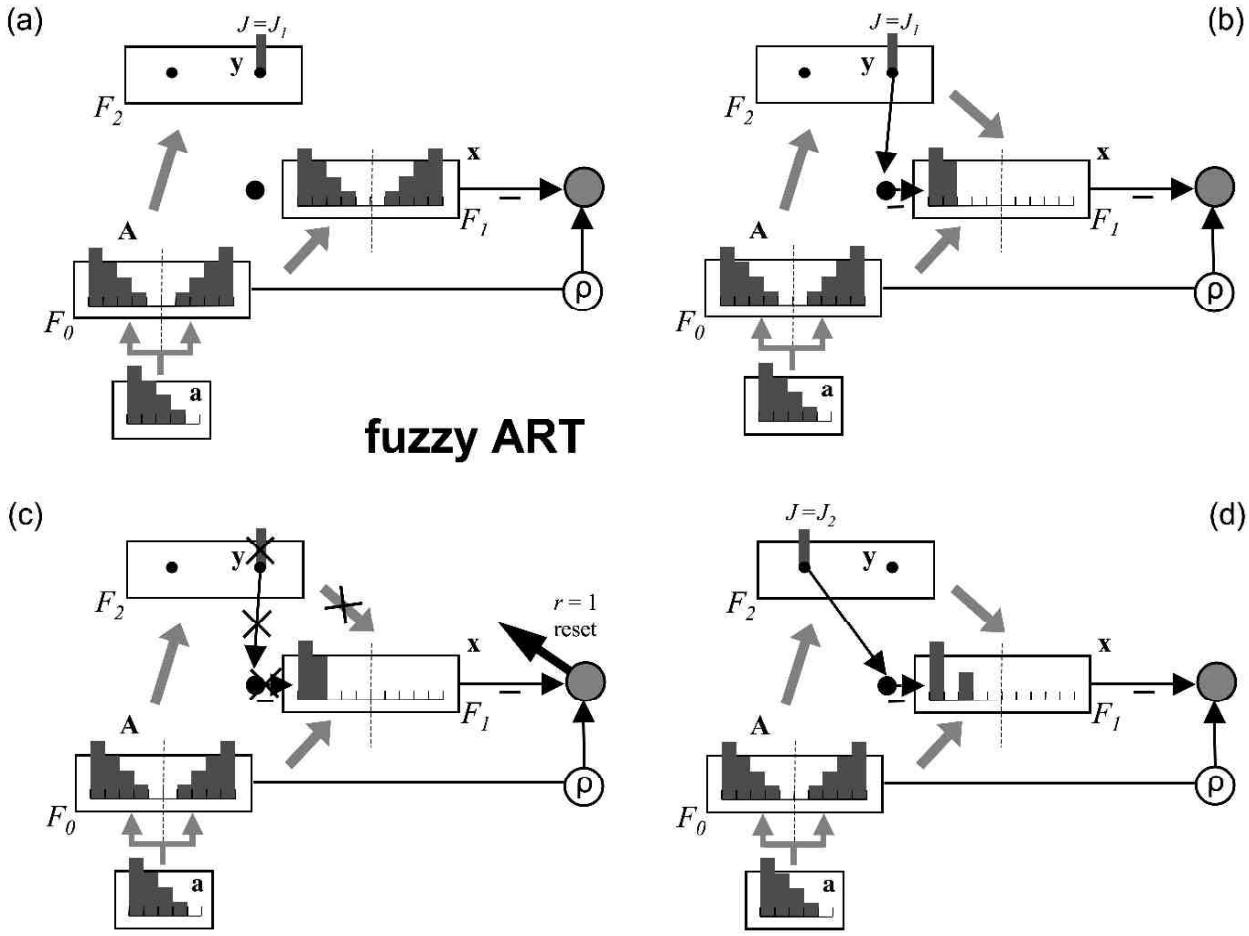


Figure 3. A fuzzy ART search cycle (Carpenter, Grossberg & Rosen, 1991), with a distributed ART network configuration (Carpenter, 1997). The ART 1 search cycle (Carpenter & Grossberg, 1987) is the same, but allows only binary inputs and did not originally feature complement coding. The match field F_1 represents the matched activation pattern $\mathbf{x} = \mathbf{A} \wedge \mathbf{w}_J$, where $\wedge$ denotes the component-wise minimum, or fuzzy intersection, between the bottom-up input **A** and the winner-take-all top-down expectation $\mathbf{w}_J$. If the matched pattern fails to meet the matching criterion, then the active code is reset at F_2, and the system searches for another code **y** that better represents the input. The match / mismatch decision is made in the ART orienting system. Each component of the input pattern **A** excites the orienting system with gain equal to the vigilance parameter ρ. Hence, with complement coding, the total excitatory input is $\rho|\mathbf{A}| = \rho\sum_{i=1}^{2M} A_i = \rho M$. Active cells in the matched pattern **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 occur. If $\rho|\mathbf{A}| - |\mathbf{x}| > 0$, then the reset signal $r = 1$, initiating search for a better matching code.

Match tracking forces an ARTMAP system not only to reset its mistakes, but to learn from them. With match tracking and fast learning, each ARTMAP network passes the Next Input Test, which requires that, if a training input were re-presented immediately after its learning trial, it would directly activate the correct output class, with neither predictive errors nor search. While match tracking guarantees that a winner-take-all network passes the Next Input Test, it does not also guarantee that a *distributed* prediction, such as those made during default ARTMAP testing, would pass the Next Input Test. A training step added to default ARTMAP 2 does provide such a guarantee (Amis & Carpenter, 2007).

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 formed as exceptions to the more general "rules" defined by the former. ARTMAP learning typically produces a wide variety of such mixtures, whose exact composition depends upon the order of training exemplar presentation.

Activity $\mathbf{x}$ at the ART field F_1 continuously computes the match between the field's bottom-up and top-down input patterns. The reset signal r shuts off the active F_2 node J when $\mathbf{x}$ 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 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 in the F_0-to-F_2 adaptive filter which biases the system against rechoosing a node that had just produced a reset.

3.3. Committed nodes and uncommitted nodes

With winner-take-all ARTMAP fast learning, a node J becomes *committed* the first time it is activated. The weight vector $\mathbf{w}_J$ converges to the input $\mathbf{A}$, and node J is permanently linked to the current output.

Unless they have already activated all their coding nodes, ART systems contain a reserve of nodes that have never been activated, with weights at their initial values. These *uncommitted nodes* receive bottom-up inputs and compete on an equal footing with the previously activated nodes. Any input may then choose an uncommitted node over poorly matched committed nodes (Figure 4).

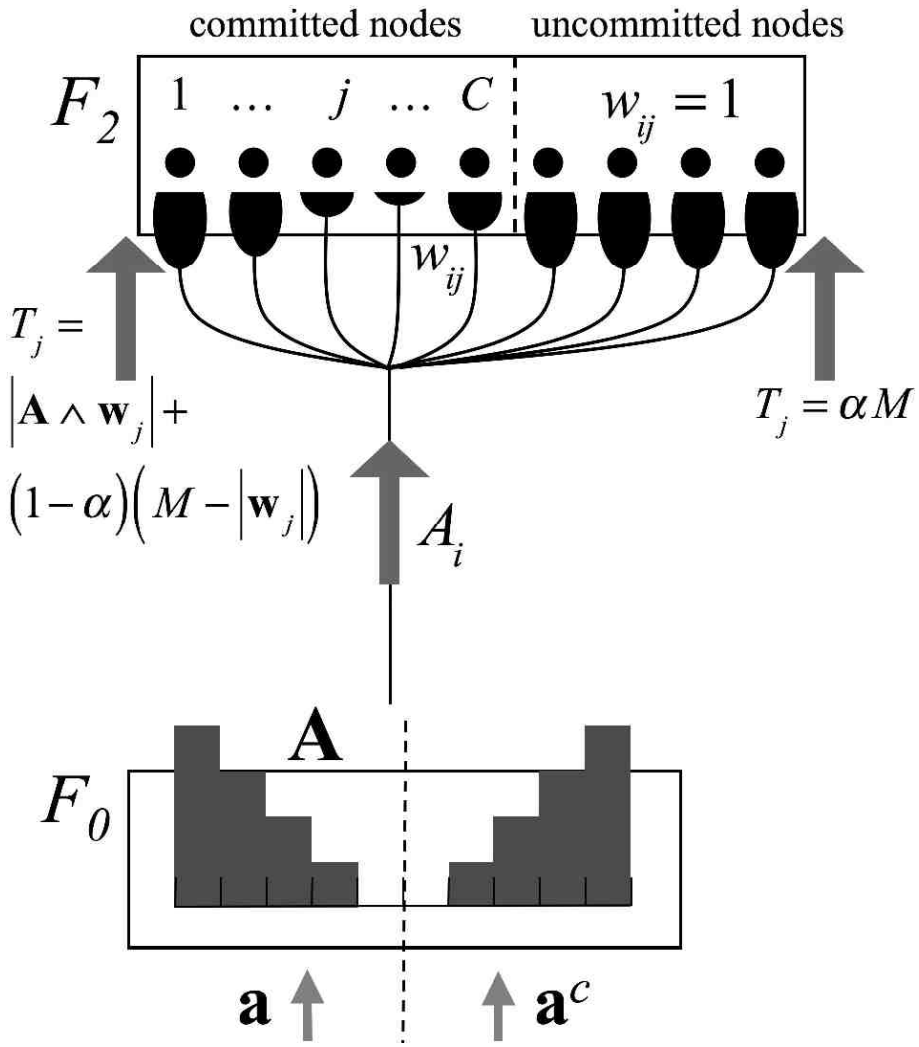


Figure 4. The ART *choice function* T_j specifies the total signal that the complement-coded input **A** sends to the F_2 node j, where α is a small parameter. The fuzzy intersection $\mathbf{A} \wedge \mathbf{w}_j$ represents the match between **A** and the weight vector $\mathbf{w}_j$, where $\wedge$ denotes the component-wise minimum of the two vectors. Uncommitted nodes have never won an F_2 competition. Their weights still equal their initial values $(w_{ij} = 1)$, plus small variations that serve to break ties with winner-take-all coding. In ARTMAP networks, a node is linked to the output class of the first input that activates it, while uncommitted nodes are linked to all possible output classes.

An ARTMAP design constraint specifies that an active uncommitted node should not reset itself. Weights begin with all $w_{ij} = 1$. Since $0 \le A_i \le 1$, when a winner-take-all active node J is uncommitted, match field activity $\mathbf{x} = \mathbf{A} \wedge \mathbf{w}_J = \mathbf{A}$. In this case, therefore, $\rho|\mathbf{A}| - |\mathbf{x}| = \rho|\mathbf{A}| - |\mathbf{A}| = (\rho - 1)|\mathbf{A}|$. Thus $\rho|\mathbf{A}| - |\mathbf{x}| \le 0$ and an uncommitted node does not trigger a reset, provided that $\rho \le 1$. While seemingly mundane, the requirement that an uncommitted node does not trigger a reset has proven to be a key design constraint across a variety of ART systems.

3.4. ART geometry

ART long-term memories are visualized as hyper-rectangles called *category boxes*. The weight vector $\mathbf{w}_J$ is represented geometrically as a box R_J whose ON-channel corner $\mathbf{u}_J$ and OFF-channel corner $\mathbf{v}_J$ are, in the format of the complement-coded input vector, defined by $(\mathbf{u}_J \mid \mathbf{v}_J^c) \equiv \mathbf{w}_J$ where $\mathbf{v}_J^c \equiv \mathbf{1} - \mathbf{v}_J$ (Figure 5a). When a node J is first committed, R_J equals the *point box* **a**.

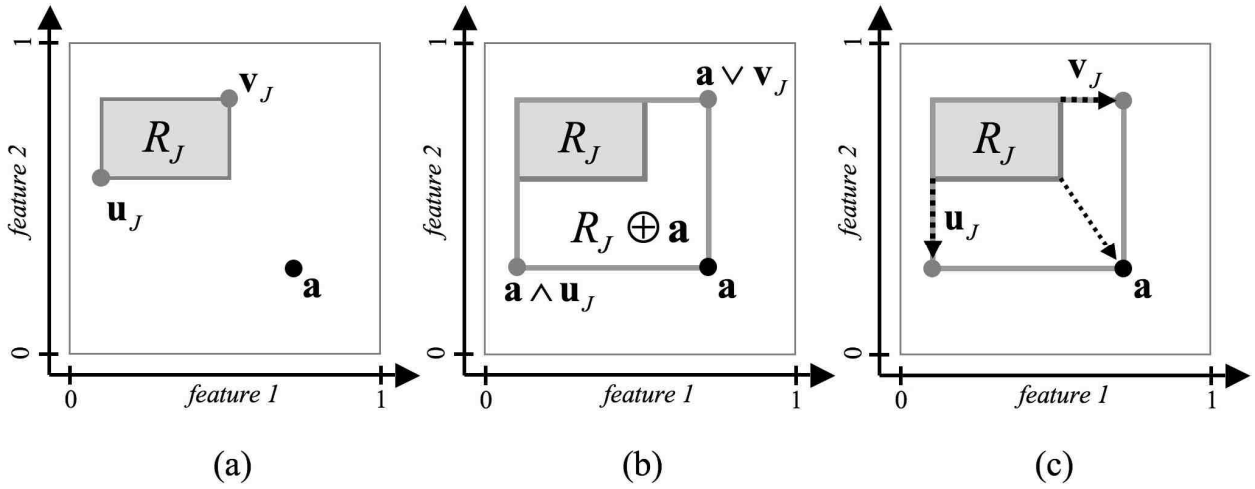


Figure 5. ART geometry. (a) The weight of a category node J is represented in complement-coding form as $\mathbf{w}_J = (\mathbf{u}_J \mid \mathbf{v}_J^c)$, and the M-dimensional vectors $\mathbf{u}_J$ and $\mathbf{v}_J$ define opposite corners of the category box R_J. When $M = 2$, the size of R_J equals its width plus its height. (b) $R_J \oplus \mathbf{a}$ is the smallest box that includes R_J and **a**. Its corners are $\mathbf{a} \wedge \mathbf{u}_J$ and $\mathbf{a} \vee \mathbf{v}_J$, where $\wedge$ denotes the component-wise minimum of two vectors and $\vee$ denotes the component-wise maximum. The active node J resets if $|R_J \oplus \mathbf{a}| > M(1 - \rho)$. (c) If $|R_J \oplus \mathbf{a}| \le M(1 - \rho)$, R_J expands toward $R_J \oplus \mathbf{a}$ during learning.

For fuzzy ART, the choice-by-difference $F_0 \to F_2$ signal function (Figure 4) is defined as:

$$T_j = \left|\mathbf{A} \wedge \mathbf{w}_j\right| + (1-\alpha)\left(M - \left|\mathbf{w}_j\right|\right) \quad (1)$$

(Carpenter & Gjaja, 1994). The function T_j translates to a geometric interpretation of category choice. The size $\left|R_j\right|$ of box *j* is defined as the sum of the edge lengths $\sum_{i=1}^{M}\left(v_{ij} - u_{ij}\right)$, which implies that

$$\left|R_j\right| = M - \left|\mathbf{w}_j\right|.$$

$R_j \oplus \mathbf{a}$, defined as the smallest box enclosing both R_j and **a** (Figure 5b), has size:

$$\left|R_j \oplus \mathbf{a}\right| = M - \left|\mathbf{A} \wedge \mathbf{w}_j\right|,$$

and the city-block (L_1) distance from R_j to **a** is:

$$d\left(R_j, \mathbf{a}\right) = \left|R_j \oplus \mathbf{a}\right| - \left|R_j\right| = \left|\mathbf{w}_j\right| - \left|\mathbf{A} \wedge \mathbf{w}_j\right|.$$

Thus geometrically the choice function is:

$$T_j = M - d\left(R_j, \mathbf{a}\right) - \alpha\left|R_j\right|. \quad (2)$$

When the choice parameter α is small, maximizing T_j is almost equivalent to minimizing $d\left(R_j, \mathbf{a}\right)$, so an input **a** activates the node *J* of the closest category box R_J. In case of a distance tie, as when **a** lies in more than one box, the node with the smallest $\left|R_j\right|$ is chosen.

According to the ART vigilance matching criterion, the chosen node *J* will reset if:

$$\rho\left|\mathbf{A}\right| - \left|\mathbf{A} \wedge \mathbf{w}_J\right| = \rho M - \left|\mathbf{A} \wedge \mathbf{w}_J\right| > 0,$$

or, geometrically, if:

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

If node *J* is not reset, R_J expands toward $R_J \oplus \mathbf{a}$ during learning (Figure 5c). Vigilance thereby sets an upper bound on the size of R_J, with

$$\left|R_J\right| \le M\left(1-\rho\right) \le M\left(1-\bar{\rho}\right).$$

After a fast learning trial, $R_J^{new} = R_J^{old} \oplus \mathbf{a}$.

4. Self-supervised ARTMAP: Geometry

ART weights w_{ij} initially equal 1. With winner-take-all coding, while the category node *J* is active $\left(y_J = 1\right)$, bottom-up weights in $F_0 \to F_2$ paths decrease during learning according to a version of the instar (Grossberg, 1976) learning law:

$$\text{Instar} \quad \frac{d}{dt} w_{iJ} = y_J\left(x_i - w_{iJ}\right) = \left(A_i \wedge w_{iJ} - w_{iJ}\right) = -\left[w_{iJ} - A_i\right]^+ = \begin{cases} -\left(w_{iJ} - A_i\right) & \text{if } w_{iJ} > A_i \\ 0 & \text{otherwise} \end{cases}. \quad (3)$$

In equation (3), $i = 1...2M$ and $[...]^+$ denotes rectification, with $[p]^+ \equiv \max\{p, 0\}$. Top-down weights w_{ji} obey an *outstar* (Grossberg, 1968) learning law, but with fast learning, $w_{ji} = w_{ij}$.

Recall that, for $i = 1...M$, $A_i = a_i$ and $w_{ij} = u_{ij}$. Thus by (3), while node J is active, u_{iJ} values decrease during learning according to:

$$\frac{d}{dt}u_{iJ} = -\left[u_{iJ} - a_i\right]^+ = \begin{cases} -\left(u_{iJ} - a_i\right) & \text{if } u_{iJ} > a_i \\ 0 & \text{otherwise} \end{cases}.$$

Geometrically, $\mathbf{u}_J$ moves toward **a**, subject to the constraints of rectification (Figure 6a).

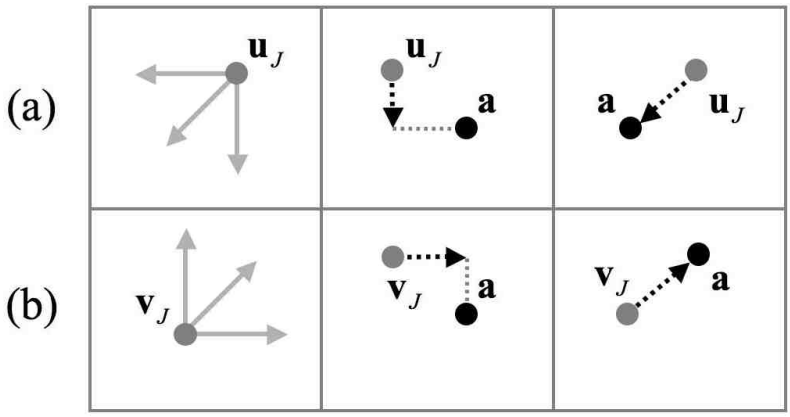


Figure 6. Geometry of ART learning. (a) During learning, components of the category box corner $\mathbf{u}_J$ can only decrease. (b) Similarly, components of the corner $\mathbf{v}_J$ can only increase during learning.

Also for $i = 1...M$, $A_{i+M} = \left(1 - a_i\right)$ and $w_{i+M,j} = \left(1 - v_{ij}\right)$. Thus while the category node J is active, v_{iJ} values increase during learning according to:

$$\frac{d}{dt}v_{iJ} = \left[a_i - v_{iJ}\right]^+ = \begin{cases} \left(a_i - v_{iJ}\right) & \text{if } v_{iJ} < a_i \\ 0 & \text{otherwise} \end{cases}.$$

Like $\mathbf{u}_J$, $\mathbf{v}_J$ moves toward **a**, subject to the constraints of rectification (Figure 6b).

With winner-take-all coding and fast learning, weights converge to their asymptotes on each input trial. In this case, $\mathbf{w}_J \rightarrow \mathbf{A}$ when node J is first activated, and $\mathbf{u}_J \rightarrow \mathbf{a}$ and $\mathbf{v}_J \rightarrow \mathbf{a}$. Thereafter, node J is committed. On subsequent learning trials that activate J (without reset), R_J expands just enough to incorporate input **a** (Figure 5c).

Before its first activation, an F_2 category node J is uncommitted and $|\mathbf{w}_J| = 2M$. At the corners of R_J , $u_{iJ} = 1$ and $v_{iJ} = 0$, so $u_{iJ} > v_{iJ}$ and the category box is *inverted* (Figure 7a). Formally,

$$|R_J| = \sum_{i=1}^{M} (v_{iJ} - u_{iJ}) = M - |\mathbf{w}_J| = -M$$

If learning is slow, R_J will remain inverted as long as $u_{iJ} > v_{iJ}$ for at least one feature i (Figure 7b).

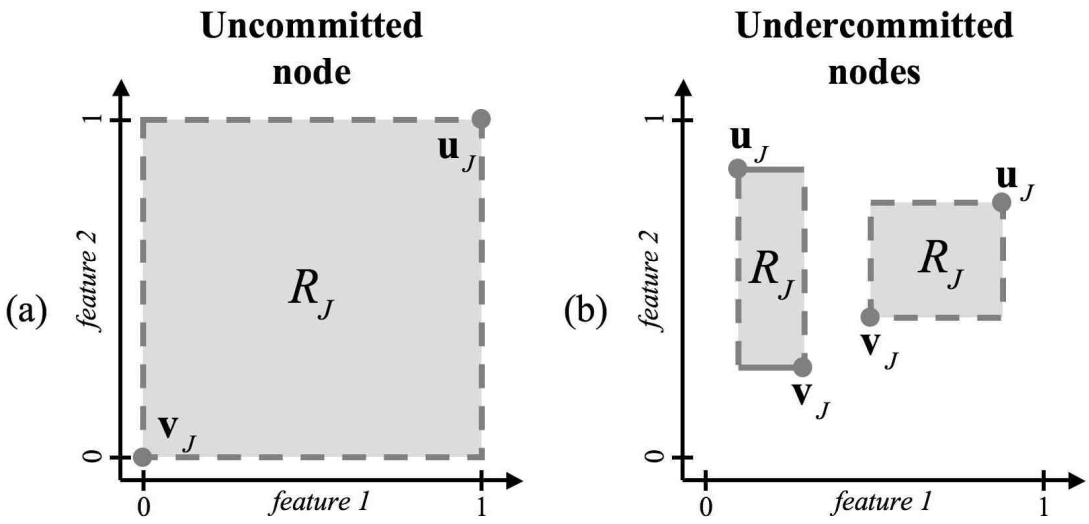


Figure 7. Geometric representations of undercommitted coding nodes. (a) Initially, all coding weights $w_{iJ} = 1$, and each node J is uncommitted. The corresponding category boxes R_J are inverted. (b) With slow learning, R_J remains inverted in the dimension of an undercommitted feature i for as long as $u_{iJ} > v_{iJ}$.

4.1. Undercommitted nodes and features

In order to describe the dynamics of slow self-supervised learning during Stage 2, SS ARTMAP defines a new node state between uncommitted and committed. During Stage 1 learning, a subset of feature values $i = 1 \ldots \bar{M}$ are specified, while the remaining feature values $i = (\bar{M} + 1) \ldots M$ are unspecified. The new definition supports the description of how features that were unspecified during Stage 1 are learned during Stage 2.

For a given node J, a feature i that is unspecified is called an *uncommitted feature* during Stage 1

learning. Whether or not a feature is specified, all weights obey the same learning law. By equation (3), the self-supervised learning model hypothesis that weights maintain their initial values at uncommitted features is equivalent to the hypothesis that $A_i = A_{i+M} = 1$ for the unspecified features $i = \left(\bar{M}+1\right) \ldots M$. Note that during Stage 1 learning **A** is complement coded only for the specified features $i = 1 \ldots \bar{M}$, and that $|\mathbf{A}| = \bar{M} + 2\left(M - \bar{M}\right) = 2M - \bar{M}$.

At the start of Stage 2 SS ARTMAP learning, all $u_{iJ} = 1$ and $v_{iJ} = 0$ for features $i = \left(\bar{M}+1\right) \ldots M$. With slow learning, these features do not immediately become fully committed, which requires that $u_{iJ} \le v_{iJ}$ (Figure 5). For as long as $u_{iJ} > v_{iJ}$, i is said to be an *undercommitted feature* with respect to node *J*, and node *J* is said to be an *undercommitted node* if its weight vector has one or more undercommitted features. Figure 8 illustrates the geometry of Stage 2 learning at undercommitted nodes.

Table 2 summarizes the definitions of feature commitment levels. The *degree of feature undercommitment*, defined as

$$\Phi_{iJ} = \left[u_{iJ} - v_{iJ}\right]^+,$$

quantifies the amount of learning needed for node *J* to become fully committed in feature *i*. The average

$$\Phi_J = \frac{1}{M}\sum_{i=1}^{M} \Phi_{iJ}$$

quantifies the *degree of undercommitment* of node *J*.

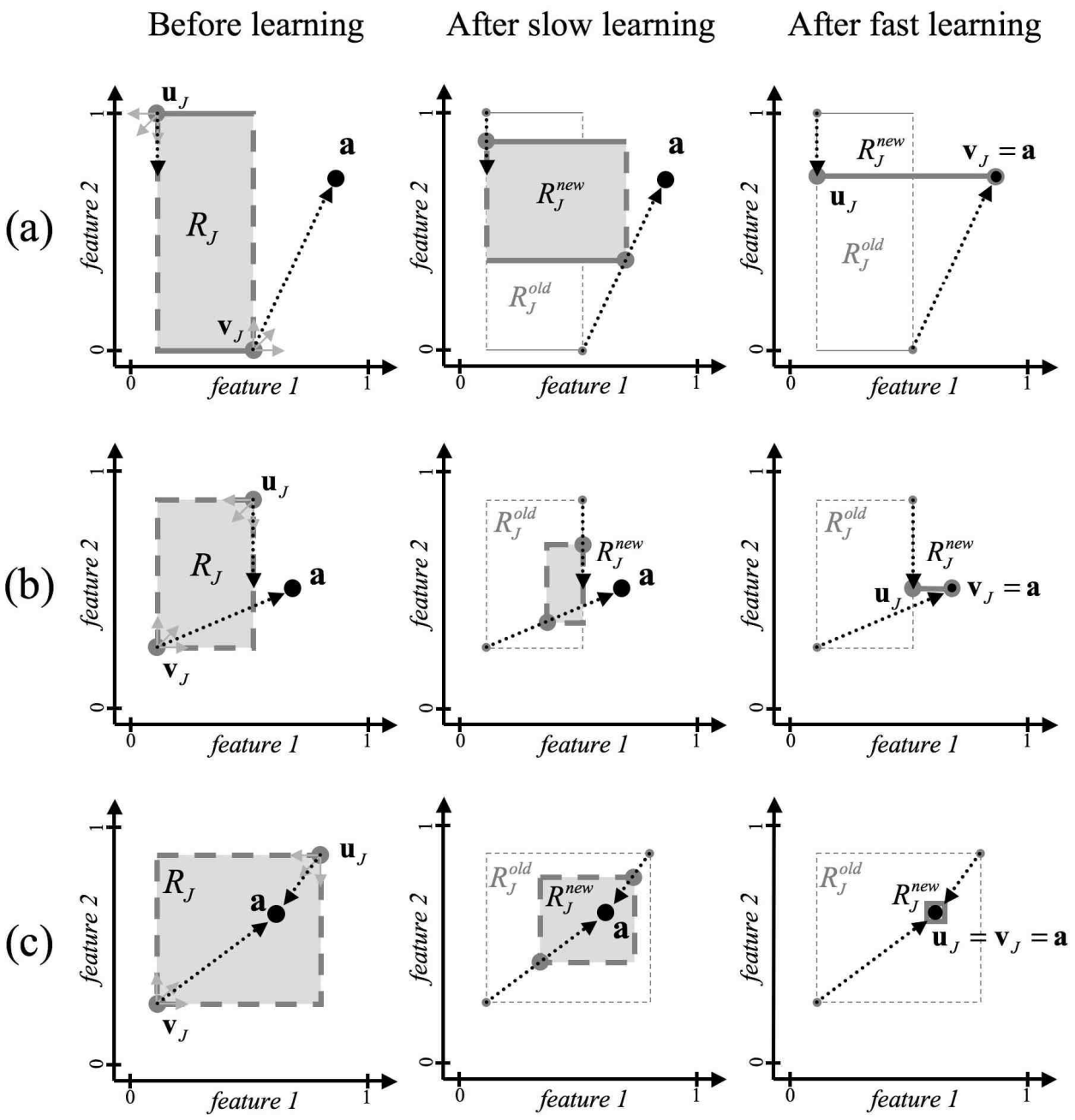
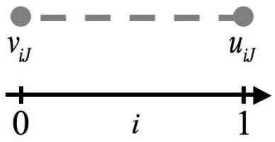
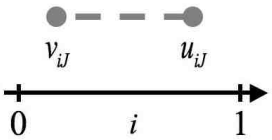
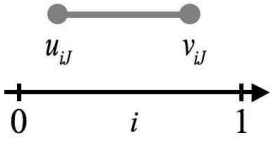


Figure 8. Stage 2 learning on undercommitted nodes activated by input **a**. (a) With $M = 2$ and $\bar{M} = 1$, when an input **a** first activates node J during Stage 2 learning, this node is committed in feature i=1 and uncommitted in feature i=2, and the coding box R_J is inverted in the vertical dimension (dashed lines). During Stage 2, both $\mathbf{u}_J$ and $\mathbf{v}_J$ move toward **a**, subject to the constraints of rectification. Slow learning contracts R_J

vertically and expands it horizontally, as $\mathbf{u}_J$ and $\mathbf{v}_J$ follow the dotted arrows. Fast learning collapses the inverted box to a line, making node *J* fully committed. (b) When input **a** activates a node *J* that is undercommitted in both feature *i*=1 and feature *i*=2, R_J is inverted in both dimensions. Slow learning contracts R_J toward **a** in both features, and fast learning collapses the box to a line. (c) When input **a** activates an inverted box R_J that contains **a**, slow learning contracts all sides of R_J. Fast learning completes the contraction to a fully committed point box with $\mathbf{u}_J = \mathbf{v}_J = \mathbf{a}$.

Table 2

Commitment definitions

Uncommitted feature		$u_{iJ} = 1,\ v_{iJ} = 0$ $w_{iJ} + w_{i+M,J} = 2$	Uncommitted node *All features are uncommitted* $\Phi_J = 1$
Undercommitted feature		$1 \geq u_{iJ} > v_{iJ} \geq 0$ $2 \geq w_{iJ} + w_{i+M,J} > 1$	Undercommitted node *At least one feature is undercommitted* $0 < \Phi_J \leq 1$
Committed feature		$u_{iJ} \leq v_{iJ}$ $0 \leq w_{iJ} + w_{i+M,J} \leq 1$	Committed node *All features are committed* $\Phi_J = 0$

Recall that $\mathbf{w}_J \equiv \left(\mathbf{u}_J, \mathbf{v}_J^c\right)$, so:

$$\Phi_J = \frac{1}{M}\sum_{i=1}^{M}\left[w_{iJ} - w^c_{i+M,J} \right]^+ = \frac{1}{M}\sum_{i=1}^{M}\left[w_{iJ} + w_{i+M,J} - 1 \right]^+ . \quad (4)$$

For an uncommitted node, all $w_{iJ} = 1$ and $\Phi_J = 1$. For a committed node, all $u_{iJ} \le v_{iJ}$ and $\Phi_J = 0$. Once a node J has been activated during Stage 1, features $i = 1...\bar{M}$ are committed $\left(u_{iJ} \le v_{iJ}\right)$ while features $i = \bar{M}+1...M$ remain uncommitted $\left(u_{iJ} = 1,\ v_{iJ} = 0\right)$, so $\Phi_J = \left(M - \bar{M}\right) \big/ M$.

5. Distributed Stage 2 learning

SS ARTMAP Stage 2 learning updates weights according to the learning laws of distributed ART architectures (Carpenter, 1997).

5.1. Distributed ART learning

Distributed ART networks are designed to support stable fast learning with arbitrarily distributed F_2 code patterns **y**. Achieving this computational goal led to the definition of a new unit of memory, a *dynamic weight* $\left[y_j - \tau_{ij} \right]^+$ equal to the degree to which the activity of coding node *j* exceeds an *adaptive threshold* τ_{ij}. The *distributed instar* (dInstar) updates the thresholds τ_{ij} in bottom-up pathways from F_0 to F_2, and the *distributed outstar* (dOutstar) updates thresholds τ_{ji} in top-down pathways from F_2 to F_1:

dInstar (bottom-up): $$\frac{d}{dt}\tau_{ij} = \left[\left[y_j - \tau_{ij} \right]^+ - A_i \right]^+ = \left[y_j - \tau_{ij} - A_i \right]^+ \quad (5)$$

dOutstar (top-down): $$\frac{d}{dt}\tau_{ji} = \left[y_j - \tau_{ji} \right]^+ \left(\sigma_i - x_i \right), \text{ where } \sigma_i = \sum_{\lambda}\left[y_\lambda - \tau_{\lambda i} \right]^+ \quad (6)$$

In the dOutstar equation (6), the top-down dynamic weight $\left[y_j - \tau_{ji} \right]^+$ is proportional to the input from the F_2 node *j* to the F_1 node *i*, and σ_i is the total input from F_2 to node *i* (Carpenter, 1994). During dOutstar learning, the top-down signal σ_i tracks F_1 activity x_i, with each adaptive threshold τ_{ji} increasing according to the contribution of its dynamic weight to the total signal, with fast as well as slow learning.

Formally defining the bottom-up weights $w_{ij} \equiv 1 - \tau_{ij}$ and the top-down weights $w_{ji} \equiv 1 - \tau_{ji}$

transforms equations (5) and (6) to:

dInstar (bottom-up): $\frac{d}{dt} w_{ij} = -\left[y_j - \left(1 - w_{ij}\right) - A_i \right]^+$

dOutstar (top-down): $\frac{d}{dt} w_{ji} = -\left[y_j - \left(1 - w_{ji}\right) \right]^+ \left(\sigma_i - x_i\right)$, where $\sigma_i = \sum_\lambda \left[y_\lambda - \left(1 - w_{\lambda i}\right) \right]^+$

With winner-take-all coding,

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

In this case, the bottom-up dynamic weight is

$$\left[y_j - \tau_{ij} \right]^+ = \begin{cases} \left(1 - \tau_{iJ}\right) & \text{if } j = J \\ 0 & \text{if } j \neq J \end{cases} = \begin{cases} w_{iJ} & \text{if } j = J \\ 0 & \text{if } j \neq J \end{cases}.$$

With winner-take-all fast learning, bottom-up and top-down weights are equal. In this case, with $x_i = A_i \wedge w_{iJ}$ when node $j=J$ is active, dInstar and dOutstar equations reduce to an instar learning law (equation (3)).

Even with fast learning, bottom-up weights of distributed ART do not generally equal top-down weights. This is because, according to distributed instar learning , a bottom-up weight w_{ij} depends only on the activity y_j at its target node. In contrast, according to distributed outstar learning, a top-down weight w_{ji} is a function of the entire active coding pattern **y**.

SS ARTMAP employs winner-take-all fast learning during Stage 1. Therefore at the start of Stage 2, bottom-up weights equal top-down weights. During Stage 2 learning, the F_2 code **y** is distributed, so bottom-up and top-down weights would diverge. Since top-down weights will not be used again, they are not computed in the current SS ARTMAP algorithm. Extensions of this system that alternate between Stage 1 and Stage 2 learning, or that need to compute the bottom-up/top-down match during Stage 2 or testing, would need to update the top-down weights via distributed outstar learning during Stage 2.

The dInstar and dOutstar equations are piecewise-linear and can be solved analytically. During Stage 2 learning, weights are updated according to the dInstar solution:

$$w_{ij} = w_{ij}^{\text{old}} - \overline{\overline{\beta}} \left[y_j - \left(1 - w_{ij}^{\text{old}}\right) - A_i \right]^+$$

where $0 < \overline{\overline{\beta}} \leq 1$. With fast learning, $\overline{\overline{\beta}} = 1$. With slow learning, $\overline{\overline{\beta}}$ is small. Simulations here set $\overline{\overline{\beta}} = 0.01$ for Stage 2 slow learning computations.

5.2. Distributed activation patterns

During SS ARTMAP testing and Stage 2 learning, the F_2 code is distributed, with the activation pattern **y** a contrast-enhanced and normalized version of the $F_0 \rightarrow F_2$ signal pattern **T**. While the transformation of **T** to **y** is assumed to be realized in a real-time network by a competitive field, simulations typically approximate competitive dynamics in their steady-state, according to an algebraic rule. For this purpose, the SS ARTMAP algorithm uses the *Increased Gradient Content-Addressable Memory (IG CAM) Rule*, which was developed for distributed ARTMAP (Carpenter et al., 1998).

The IG CAM Rule features a contrast parameter *p*. As *p* increases, the $\mathbf{T} \rightarrow \mathbf{y}$ transformation becomes increasingly sharp, approaching winner-take-all as $p \rightarrow \infty$. SS ARTMAP simulations set *p*=2 for distributed activation during Stage 2 and testing. The hypothesis that SS ARTMAP activation is winner-take-all during Stage 1 learning, when answers are provided, corresponds to the hypothesis that confidence modulates the competitive field, with high-confidence states corresponding to large values of the contrast parameter *p*.

With the IG CAM Rule, if **a** is in one or more category boxes, activation y_j is distributed among these nodes, with y_j largest where the boxes R_j that contain **a** are smallest. If **a** is not in any box, y_j is distributed according to the distance $d\left(R_j, \mathbf{a}\right)$, with y_j largest where **a** is closest to R_j.

6. Modifying the choice function for undercommitted coding nodes

During Stage 2, when learning is slow and distributed and inputs **a** are fully featured, the degree of undercommitment Φ_j decreases from $\left(1 - \bar{M}/M\right)$ toward 0 as a coding node *j* gradually becomes more committed. Within Stage 2, Φ_j values vary across nodes. The ART choice function T_j as defined by equation (1) would favor more committed nodes to the point that nodes that became active early in Stage 2 would block learning in all other nodes. Self-supervised ARTMAP therefore modifies the definition of T_j so that the less committed nodes are able to compete successfully with the more committed nodes, as follows.

6.1. Balancing committed and undercommitted nodes during Stage 2 learning

Fast-learn winner-take-all ARTMAP networks choose the coding node *J* that maximizes the choice-by-difference signal function:

$$T_j = M - \left(\left|\mathbf{w}_j\right| - \left|\mathbf{A} \wedge \mathbf{w}_j\right|\right) - \alpha\left(M - \left|\mathbf{w}_j\right|\right) = M - d\left(R_j, \mathbf{a}\right) - \alpha\left|R_j\right|$$

(Figure 4). Since α is small, the choice function T_j favors nodes j whose category boxes R_j are closest to **a**. Distance ties are broken in favor of smaller boxes. The example illustrated in Figure 9 shows that, in a learning system with mixed degrees of coding node commitment, this signal function also implicitly favors the nodes with greatest commitment. In order to focus on primary T_j design with respect to the distance from R_j to **a**, α is temporarily set equal to 0:

$$T_j \cong M - d\left(R_j, \mathbf{a}\right). \tag{7}$$

Figure 9 illustrates a hypothetical case such as might occur during Stage 2 self-supervised learning, when various coding nodes j have different degrees of undercommitment Φ_j (equation (4)). In this example, node j=1 is fully committed $\left(\Phi_1 = 0\right)$, while node j=2 is still uncommitted in feature i=2 $\left(\Phi_2 = 0.5\right)$. If the ARTMAP choice function (equation (7)) were used, the signal T_1 to the committed node j=1 would be greater than the signal T_2 to the undercommitted node j=2 for *all* inputs **a**, as indicated by the grey area of Figure 9a. This function excessively favors nodes that happened to become active, and therefore more committed, early in Stage 2. Many of the Stage 1 nodes would be left permanently inactive, unable either to contribute to predictions or to learn from Stage 2 experience.

To see why the signal function of Figure 9a favors committed over undercommitted nodes, note that, for any **a** at the start of Stage 2, all weights $w_{ij} = w_{i+M,j} = 1$ for features $i = \bar{M}+1 \ldots M$. Thus at this moment,

$$\begin{aligned}
&d\left(R_j, \mathbf{a}\right) \equiv \left|\mathbf{w}_j\right| - \left|\mathbf{A} \wedge \mathbf{w}_j\right| = \sum_{i=1}^{2M}\left[w_{ij} - A_i\right]^+ = \\
&\sum_{i=1}^{M}\left(\left[w_{ij} - a_i\right]^+ + \left[w_{i+M,j} - \left(1 - a_i\right)\right]^+\right) = \\
&\sum_{i=1}^{\bar{M}}\left(\left[w_{ij} - a_i\right]^+ + \left[w_{i+M,j} - \left(1 - a_i\right)\right]^+\right) + \sum_{i=\bar{M}+1}^{M}\left(\left(1 - a_i\right) + \left(1 - \left(1 - a_i\right)\right)\right) = \\
&\sum_{i=1}^{\bar{M}}\left(\left[w_{ij} - a_i\right]^+ + \left[w_{i+M,j} - \left(1 - a_i\right)\right]^+\right) + M - \bar{M} = \\
&\sum_{i=1}^{\bar{M}}\left(\left[w_{ij} - a_i\right]^+ + \left[w_{i+M,j} - \left(1 - a_i\right)\right]^+\right) + M\Phi_j
\end{aligned} \tag{8}$$

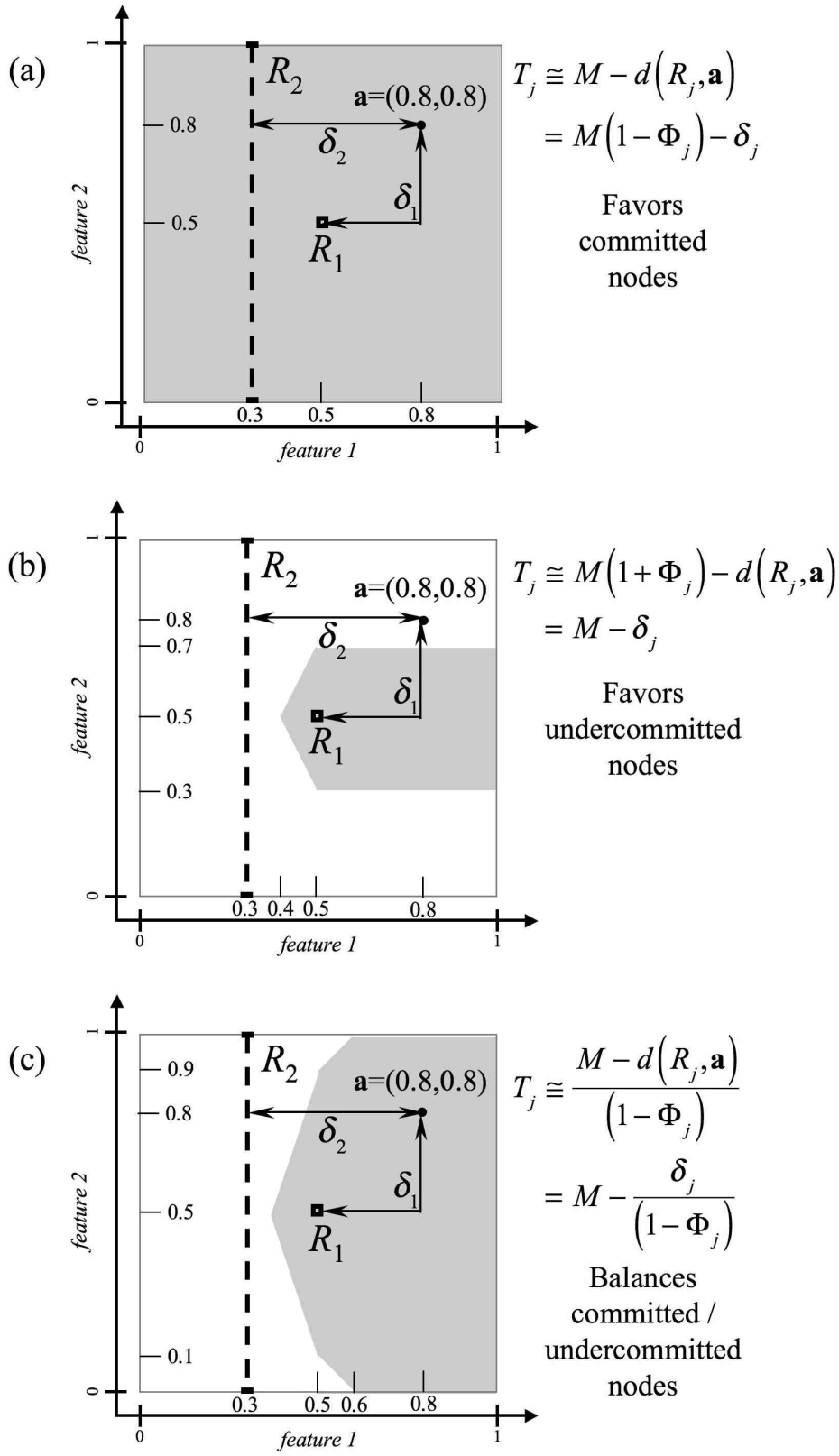


Figure 9. Choosing the Stage 2 SS ARTMAP signal function. The coding node j=1, corresponding to the weight vector $\mathbf{w}_1 = \left(0.5, 0.5 \mid 0.5, 0.5\right)$ and the point box R_1, is

committed $\left(\Phi_1 = 0\right)$. The coding node *j*=2, corresponding to the weight vector $\mathbf{w}_2 = \left(0.3,1 \mid 0.7,1\right)$ and the inverted box R_2, is undercommitted $\left(\Phi_2 = 0.5\right)$. Grey areas indicate the test points $\mathbf{a} = \left(a_1, a_2\right)$ that would activate node *j*=1 more than node *j*=2 for each alternative signal function T_j. Term δ_j is the distance from **a** to R_j *in committed features only*. The approximate signal functions set α equal to 0, in order to focus on the geometry of distance from input **a** to boxes R_j. (a) T_j favors committed over undercommitted nodes. (b) T_j favors undercommitted over committed nodes. (c) T_j strikes a balance between committed and undercommitted nodes.

Table 2 summarizes the definitions of feature commitment levels. The *degree of feature undercommitment*, defined as

$$\Phi_{iJ} = \left[u_{iJ} - v_{iJ} \right]^+ ,$$

quantifies the amount of learning needed for node *J* to become fully committed in feature *i*. The average

$$\Phi_J = \frac{1}{M} \sum_{i=1}^{M} \Phi_{iJ}$$

quantifies the *degree of undercommitment* of node *J*.

As long as $\mathbf{w}_j$ remains uncommitted in features $i = \bar{M} + 1 ... M$, the term

$$\delta_j \equiv \sum_{i=1}^{\bar{M}} \left(\left[w_{ij} - a_i \right]^+ + \left[w_{i+M,j} - \left(1 - a_i\right) \right]^+ \right)$$

in equation (8) may be interpreted as the distance from **a** to R_j in the committed features $i = 1 ... \bar{M}$. The *committed distances* δ_1 and δ_2 are illustrated in Figure 9 for the point **a** = (0.8,0.8), for which $\delta_1 = 0.6$ and $\delta_2 = 0.5$. This analysis suggests adding $M\Phi_j$ to the signal function $T_j = M - \left(\delta_j + M\Phi_j\right)$ (equation (7)) in order to favor nodes *j* that are closest to **a** in their committed features. The resulting signal function is then:

$$T_j \cong M\left(1 + \Phi_j\right) - d\left(R_j, \mathbf{a}\right) = M - \delta_j$$

(Figure 9b).

This second strategy raises the concern that a fully committed node with a good match in all features might lose to a highly undercommitted node with a good match in just one feature. This concern is validated by noting that $\Phi_j = 1$ and $\delta_j = 0$ at each uncommitted node *j*, so that $T_j = M$ would be *maximal* for these nodes, where no learning has ever taken place (Figure 4).

Figure 9c suggests an alternative strategy for biasing T_j in such a way that undercommitted nodes can compete for Stage 2 activation while respecting the specificity of learning at more committed nodes. Here,

$$T_j \cong \frac{M - d\left(R_j, \mathbf{a}\right)}{\left(1 - \Phi_j\right)} = \frac{M - \left(\delta_j + M\Phi_j\right)}{\left(1 - \Phi_j\right)} = M - \frac{\delta_j}{\left(1 - \Phi_j\right)}.$$

In Figure 9c, $\Phi_1 = 0$ and $T_1 = \left(2 - \delta_1\right)$ at the committed node *j*=1, and $\Phi_2 = 0.5$ and $T_2 = \left(2 - 2\delta_1\right)$ at the undercommitted node *j*=2. This third design strategy is implemented in SS ARTMAP, though the full definition of T_j is perturbed to break distance ties and avoid division by 0.

6.2. Balancing committed and uncommitted features during Stage 1 learning

Analysis of Stage 2 learning (Figure 9) supports the signal function definition:

$$T_j = \frac{M - \left(\left|\mathbf{w}_j\right| - \left|\mathbf{A} \wedge \mathbf{w}_j\right|\right)}{1 - \gamma\,\Phi_j} - \alpha\left(M - \left|\mathbf{w}_j\right|\right) = \frac{M - d\left(R_j, \mathbf{a}\right)}{1 - \gamma\,\Phi_j} - \alpha\left|R_j\right| \tag{9}$$

where $0 < \gamma < 1$. When a node is committed $\left(\Phi_j = 0\right)$, the signal function (9) reduces to the ART choice-by-difference signal function (1). This definition is problematic, however, for Stage 1 learning. During Stage 1, where only $\bar{M}$ of the *M* input features are specified,

$$A_i = \begin{cases} a_i & \text{if } i=1...\bar{M} \\ \left(1 - a_i\right) & \text{if } i=\left(1+M\right)...\left(\bar{M}+M\right) \\ 1 & \text{otherwise} \end{cases}.$$

At uncommitted nodes *j*, all $w_{ij} = 1$, so $\left|\mathbf{w}_j\right| = 2M$ and the degree of undercommitment $\Phi_j = 1$. Thus if *j* is an uncommitted node, the denominator in equation (9) equals $\left(1 - \gamma\right)$, which is typically small. Hence the only way to keep uncommitted nodes from overwhelming the coding pattern **y** is to require that the signal function numerator equal 0 when $\Phi_j = 1$. For an uncommitted node *j* during Stage 1,

$$d\left(R_j, \mathbf{a}\right) = \left|\mathbf{w}_j\right| - \left|\mathbf{A} \wedge \mathbf{w}_j\right| = 2M - \left(2M - \bar{M}\right) = \bar{M}\ .$$

This analysis suggests that the Stage 2 signal function of equation (9) needs to be modified for Stage 1 to:

$$T_j = \frac{\bar{M} - \left(\left|\mathbf{w}_j\right| - \left|\mathbf{A} \wedge \mathbf{w}_j\right|\right)}{1 - \gamma\,\Phi_j} - \alpha\left(M - \left|\mathbf{w}_j\right|\right) = \frac{\bar{M} - d\left(R_j, \mathbf{A}\right)}{1 - \gamma\,\Phi_j} - \alpha\left|R_j\right| \tag{10}$$

For the Stage 1 signal function (10), $T_j = \alpha M$ when *j* is an uncommitted node. For nodes

$j = 1 \ldots C$ which have been chosen at least once during Stage 1, $\Phi_j = \left(1 - \bar{M}/M\right)$. For these nodes, the numerator in equation (10) equals $\left(\bar{M} - \delta_j\right)$ and the denominator is a constant that is independent of *j*. Thus, except for distance ties, this signal function would choose the category node for which **a** is closest to R_j in the specified input dimensions $i = 1 \ldots \bar{M}$ during Stage 1 training.

The signal function (10) proposed for Stage 1 and the signal function (9) proposed for Stage 2 are reconciled by noting that

$$2M - |\mathbf{A}| = \begin{cases} \bar{M} & \text{during Stage 1} \\ M & \text{during Stage 2} \end{cases}.$$

6.3. The self-supervised ARTMAP signal function

A signal function that realizes the computational requirements of SS ARTMAP is:

$$T_j = \frac{\left(2M - |\mathbf{A}|\right) - \left(|\mathbf{w}_j| - |\mathbf{A} \wedge \mathbf{w}_j|\right)}{1 - \gamma \Phi_j} - \alpha\left(M - |\mathbf{w}_j|\right) \tag{11}$$

where $0 < \gamma < 1$ and $\alpha > 0$. Stage 1 specifies only $\bar{M}$ features of input **a**, so $|\mathbf{A}| = 2M - \bar{M}$. For the nodes $j = 1 \ldots C$ chosen during Stage 1, $\Phi_j = \left(1 - \bar{M}/M\right)$. During Stage 2, which specifies all *M* features of **a**, $|\mathbf{A}| = M$ and Φ_j values slowly decrease toward 0 at active nodes *j*. Thus for $j = 1 \ldots C$:

Stage 1
$$T_j = \frac{\bar{M} - \left(|\mathbf{w}_j| - |\mathbf{A} \wedge \mathbf{w}_j|\right)}{\mu} - \alpha\left(M - |\mathbf{w}_j|\right) = \frac{\bar{M} - d\left(R_j, \mathbf{a}\right)}{\mu} - \alpha |R_j| \tag{12}$$

Stage 2
$$T_j = \frac{M - \left(|\mathbf{w}_j| - |\mathbf{A} \wedge \mathbf{w}_j|\right)}{1 - \gamma \Phi_j} - \alpha\left(M - |\mathbf{w}_j|\right) = \frac{M - d\left(R_j, \mathbf{a}\right)}{1 - \gamma \Phi_j} - \alpha |R_j| \tag{13}$$

where the denominator $\mu \equiv 1 - \gamma\left(1 - \bar{M}/M\right)$ is constant throughout Stage 1. During testing, T_j is the same as during Stage 2. For each uncommitted node *j*, T_j equals the constant αM throughout Stage 1 and Stage 2 learning, and testing.

7. Self-supervised learning in the medical diagnosis example

The simplified medical diagnosis example (Figure 1) suggests that Stage 2 learning should favor confident predictions, where category boxes of different classes do not overlap, and produce little (if any) weight change on low-confidence predictions. Self-supervised ARTMAP accomplishes this computational goal by reflecting the degree of overlap of category boxes that contain the input **a**. Inputs that are in just one box indicate confident class membership and

produce most of the Stage 2 learning.

Figure 10a illustrates the five category boxes created by the first 28 inputs of Stage 1 learning on the specified feature *i*=1 (*temperature*). At this stage of learning, test accuracy is a near-optimal 65%. Each box R_j is inverted in the uncommitted feature *shock*. The *hypothermic* box R_1 overlaps the *normal* box R_5, and the *septic* box R_2 overlaps both the *normal* box R_3 and the *septic* box R_4. Continued Stage 1 learning creates three more small boxes within the regions of overlap.

During Stage 2 self-supervised learning, distributed activation at the coding field F_2 represents the degree of overlap of category boxes containing the input **a**. In Figure 10a, $\mathbf{a} = (0.12, 0.75)$ is contained only in R_1, clearly indicating the class *hypothermic* based on the low value of the feature *temperature* alone. During Stage 2, according to the IG CAM Rule, when an input is inside just one category box R_J, $y_J \cong 1$ and other $y_j \cong 0$. In this case, distributed ART learning reduces to winner-take-all learning. Weight changes are then limited to $\mathbf{w}_J$, with the greatest changes occurring in components of the previously uncommitted features $i = \bar{M} + 1...M$ (Figure 10b).

Input $\mathbf{a} = (0.28, 0.25)$ is inside both R_1 and R_5, indicating ambiguity between *normal* and *hypothermic*. During Stage 2, the distributed inputs $(T_1 \cong T_5)$ split the distributed code $(y_1 \cong y_5 \cong 0.5)$, reducing possible learning in $\mathbf{w}_1$ and $\mathbf{w}_5$. R_1 and R_5 therefore maintain the balance between their output classes, reflecting low confidence in any prediction in this temperature range (Figure 10c). The distributed instar and distributed outstar learning laws (equations (5) and (6)) prevent strong learning on weak predictions, regardless of how often unlabeled ambiguous points are presented.

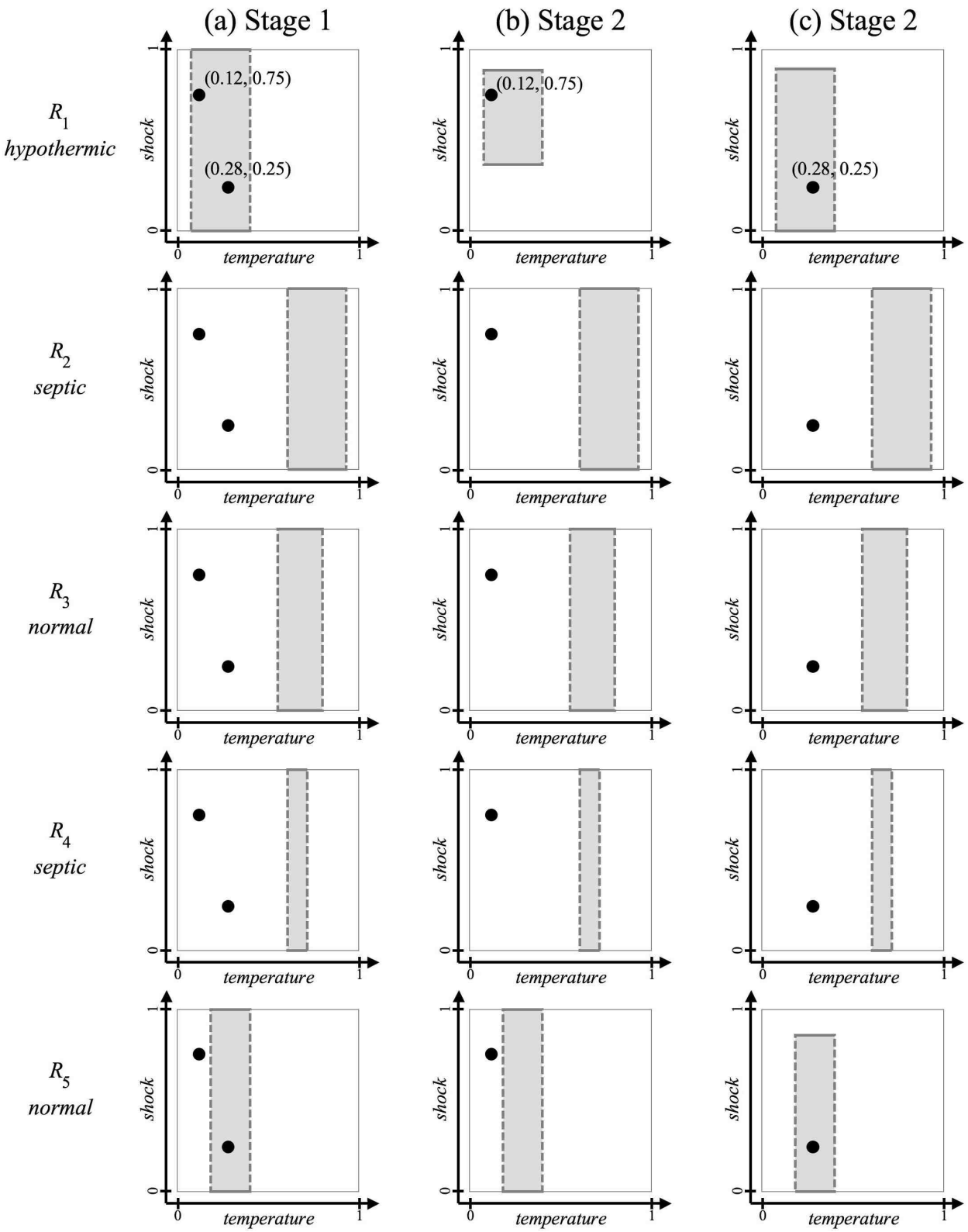


Figure 10. Self-supervised Stage 1 and Stage 2 learning. (a) Category boxes created by the first 28 Stage 1 training points on the simplified medical diagnosis problem

(Figure 1). Boxes are inverted in the uncommitted feature *i*=1 (*shock*), as indicated by the dashed edges. (b) When the unlabeled point **a** = (0.12, 0.75) is presented during Stage 2, it activates only node *j*=1. During learning, R_1 (*hypothermic*) contracts toward the high *shock* value of this input. (c) When the point **a** = (0.28, 0.25) is presented during Stage 2, it activates nodes *j*=1 and *j*=5, which share distributed activations, allowing relatively small weight updates in this case of ambiguous class predictions. All parameter values are as specified in Table 5, except that the Stage 2 learning rate is uncharacteristically high $\left(\bar{\bar{\beta}} = 0.5\right)$, which speeds up learning so as to make weight changes visible in this illustration.

Figure 11 illustrates the result of slow, distributed SS ARTMAP learning at the end of Stage 2. Driven by strong activations on confident predictions at low temperatures, R_1 (*hypothermic*) stabilizes at the top of *shock*. Similarly, R_2 and R_4 (*septic*) stabilize at the top of *shock*. Test accuracy has improved to nearly 100% and coding nodes $j = 1...5$ are almost fully committed. Category boxes do not expand nearly as much as they would in winner-take-all training. Rather, inputs outside the boxes produce distributed coding field activations, so distributed ART learning inhibits further weight changes. Stage 2 learning thus converges as the degree of node undercommitment falls to 0.

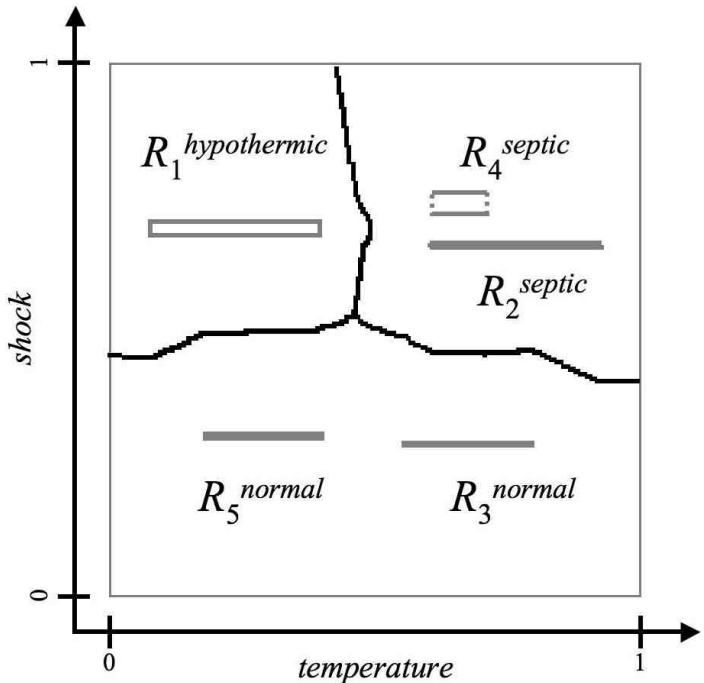


Figure 11. SS ARTMAP category boxes at the end of Stage 2 learning on the simplified medical diagnosis example, with Stage 2 training beginning with the five category boxes of Figure 10a. After 10,000 training points and slow learning $\left(\bar{\bar{\beta}} = 0.01\right)$, test accuracy is 100%.

8. The Boston satellite imagery testbed

The two-dimensional medical diagnosis problem visually illustrates how SS ARTMAP meets key challenges presented by self-supervised learning of unlabeled inputs. Higher-dimensional problems reveal additional challenges. A self-supervised learning benchmark based on the Boston satellite imagery testbed (Carpenter, Martens, & Ogas, 2005) provides such an example (Figure 12). The problem requires a system to label pixels as belonging to one of seven classes (*beach, ocean, ice, river, park, residential, industrial*) from satellite data for northeast Boston and suburbs. The testbed is partitioned into four vertical strips (Figure 12a), such that training and test sets are not only disjoint but drawn from geographically separate locations. The first Boston self-supervised learning benchmark specifies only the five input features related to blue image data $\left(\overline{M} = 5\right)$ during Stage 1 training on three strips, and all thirty-eight input features (M=38) during Stage 2 training and testing on the remaining strip. Cross-validation provides the total measure of benchmark performance by averaging across four independent simulations, each using a different strip for Stage 2 training and testing.

The Boston benchmark presents challenges to a self-supervised learning system beyond those of the two-dimensional example. Because testbed pixels have a 15m resolution, many actually cover multiple classes. Whereas the labeled training set consists primarily of clear exemplars of each class, unlabeled Stage 2 pixels are typically mixtures. Many pixels, for example, include both *ice* and *residential* patches. Stage 2 vectors thus mix and distort features from multiple classes, placing many of the unlabeled feature vectors far from the distinct class clusters of the Stage 1 training set. As a result, following Stage 1 training, winner-take-all ARTMAP would, for example, overpredict *residential* on the self-supervised Boston benchmark by an order of magnitude. As in Figure 10, distributed Stage 2 activation minimizes the influence of mixed pixels, so that most learning occurs only on unambiguous cases, despite the fact that the unlabeled pixels provide no external indices of class ambiguity.

8.1. Boston benchmark self-supervised learning

Each Boston benchmark simulation follows a standardized image cross-validation protocol (Parsons & Carpenter, 2003). With one strip reserved for testing, up to 250 labeled pixels per class are selected at random for Stage 1 training. All pixels of the remaining strip are presented in random order without labels during Stage 2. Testing calculates the accuracy for each class on this strip, with the overall accuracy set equal to the average of these class-wise percentages. This performance measure is, like the c-index, independent of the particular class mixture in a given test set. Each simulation result is the average of the independent accuracies across the four test strips. This procedure is repeated 500 times, each with a different random Stage 1 pixel selection and Stage 2 ordering.

The first set of Boston benchmark simulations take the Stage 1 feature vector to be the five input components related to blue. Self-supervised Stage 2 learning dramatically improves performance on the Boston benchmark (Figure 13). On every one of the 500 individual simulations, Stage 2 learning improved test accuracy, as unlabeled fully featured inputs consistently expanded

knowledge from Stage 1 training. Following Stage 1 (dark bars, Figure 13a), the class-based test accuracy was between 50% and 58% across the 500 simulations. Stage 2 self-supervised learning, with all 38 features specified, improved accuracy on each simulation to above 70%, with a mean of 90% (light bars). Figure 13b displays test accuracy on each class for a typical SS ARTMAP trial. Note, for example, that Stage 2 learning improves accuracy of *ice* ground-truth pixels from 0% (Stage 1) to 99%.

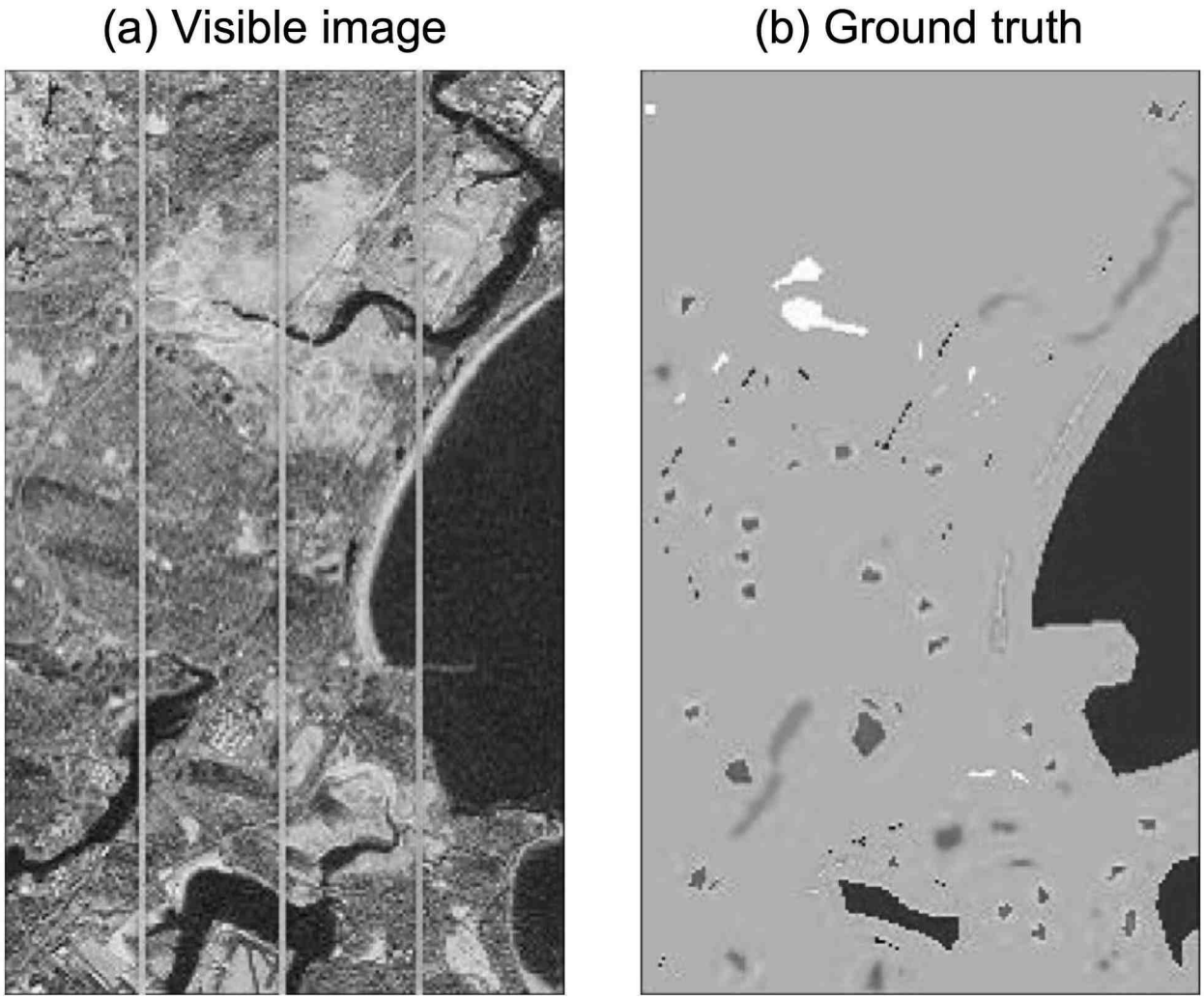


Figure 12. The satellite image and labeled ground-truth pixels for the Boston testbed (http://techlab.bu.edu/classer/data_sets). Only 28,735 of the 216,000 pixels are labeled, of which only 6,634 represent classes other than *ocean*. (a) Cross-validation divides the testbed into four vertical strips. (b) Class distributions vary substantially across strips. For example, the right-hand strip contains mainly *ocean* pixels (dark), while the left-hand strip contains neither *ocean* nor *beach* pixels.

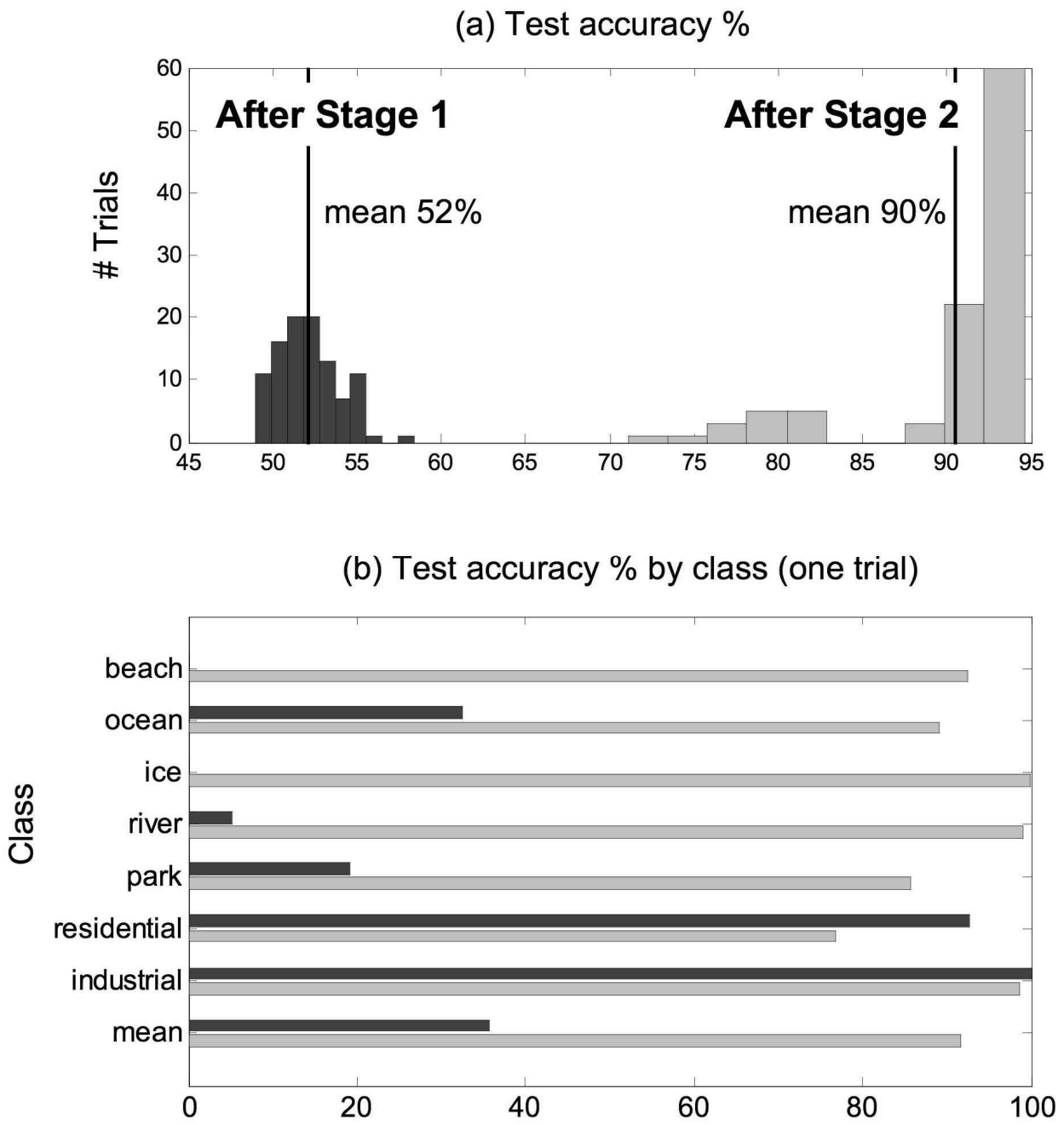


Figure 13. ARTMAP performance on the Boston self-supervised learning benchmark. Stage 1 training specifies class labels and five blue-related features in three strips. Stage 2 training specifies all 38 input features but no class labels for pixels in the remaining strip. Testing evaluates the performance of the system on fully featured inputs from the fourth strip. (a) Performance histograms for 500 randomized trials. (b) For one trial, the fraction of pixels labeled as belonging to a class that were predicted as in that class. Compared to Stage 1, per-class performance in Stage 2 declined only in classes that had been over-predicted in Stage 2 (*residential*, *industrial*).

Confusion matrices for one of the Boston simulation examples further illustrate the pattern of Stage 2 performance improvements. Following Stage 1 learning (Figure 14a), SS ARTMAP's per-class test accuracy was high for *residential* and *industrial* pixels but low for others. For example, all *beach* and *ice* pixels were misclassified as *industrial*. If the system were to learn on unlabeled pixels as if its own winner-take-all class predictions were ground-truth labels—the "self-training" of Chapelle et al. (2006, p. 3)—mistakes on classes like *ice* would degrade correct prior knowledge. Instead, distributed activation of SS ARTMAP focuses Stage 2 learning on confident predictions and novel features, fixing nearly all the incorrect *industrial* predictions on the *beach* and *ice* pixels while maintaining *industrial* accuracy (Figure 14b).

(a) SS ARTMAP – Stage 1: 35.7%									
	Predicted class								%
		beach	*ocean*	*ice*	*river*	*park*	*residential*	*industrial*	
Actual class	*beach*	**0**						67	0.0
	ocean		**294**		584		10	14	32.6
	ice			**0**				879	0.0
	river				**17**		306		5.3
	park					**36**	152		19.1
	residential						**231**	18	92.8
	industrial							**431**	100.0
	Class-based accuracy								**35.7**

(b) SS ARTMAP – Stage 2: 91.7%%									
	Predicted class								%
		beach	*ocean*	*ice*	*river*	*park*	*residential*	*industrial*	
Actual class	*beach*	**62**				1		4	92.5
	ocean		**804**		98				89.1
	ice	1		**878**					99.9
	river		3		**320**				99.1
	park	27				**161**			85.6
	residential				18	7	**191**	33	76.7
	industrial	3					3	**425**	98.6
	Class-based accuracy								**91.7**

Figure 14. Confusion matrices for one self-supervised ARTMAP simulation of the Boston benchmark. (a) After Stage 1 learning on labeled inputs with the five blue-related feature values. (b) After Stage 2 learning on inputs with all 38 features but no class labels.

Table 3 and Figure 15 show SS ARTMAP performance on the Boston benchmark with different feature subsets selected for Stage 1 training. For the Full or Visible subsets, which specify all or most of the input features, Stage 1 training produced near-perfect test performance, which was either unchanged (Full) or slightly harmed (Visible) by Stage 2 self-supervised learning. The subsets Raw, Blue, and Red-Green specify between five and nine of the input features during Stage 1 training, where Blue is the feature subset used in Figures 13 and 14. Each of these training protocols produced Stage 1 test accuracy of about 50%, which improved to about 80-90% during Stage 2. The remaining two feature subsets specified only one (Raw Blue) or two (Raw Red-Green) Stage 1 features. Although Stage 1 performance was similar to that of the Raw, Blue, and Red-Green Subsets, Stage 1 learning did not provide a sufficient foundation on which SS ARTMAP could build useful additional knowledge during self-supervised Stage 2 learning.

Table 3

Self-supervised Boston benchmark feature subsets specified during Stage 1 training

Subset name	**Number of features** $\bar{M}$	**Features**
Full	38	All features
Visible	26	All features except those related to infrared and panchromatic bands
Raw	9	Unprocessed features
Blue (B)	5	Features related to blue
Red-green (RG)	7	Features related to red or green
Raw blue	1	Blue
Raw red-green	2	Green, red

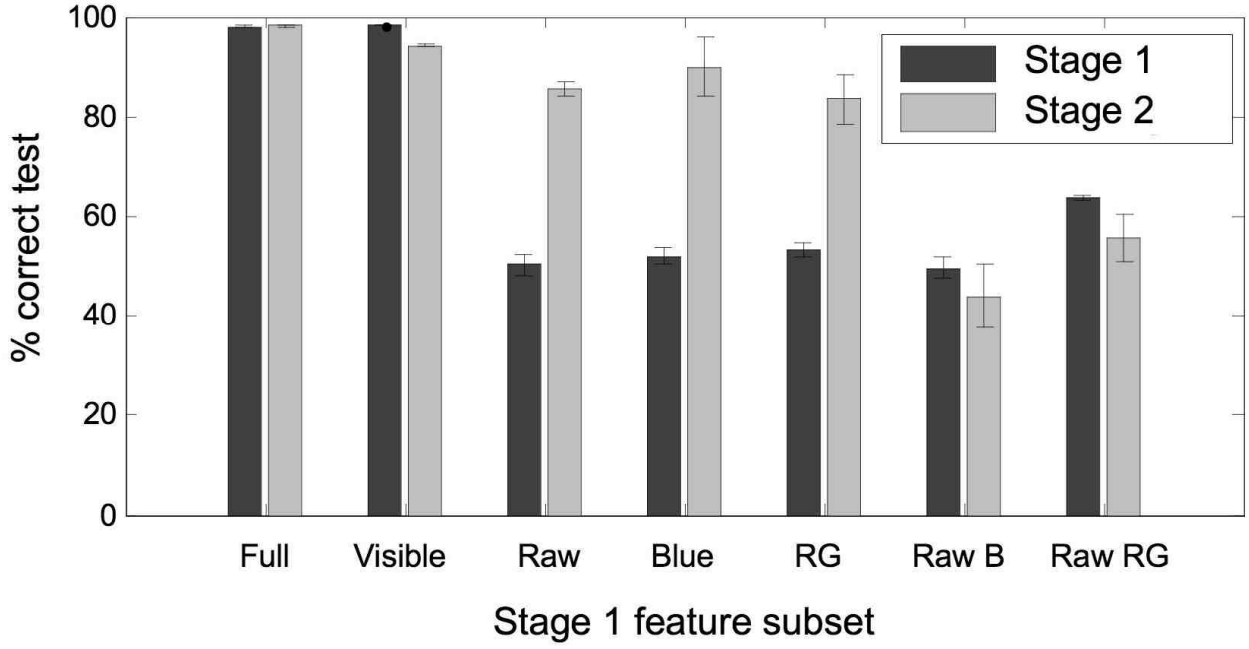


Figure 15. Boston benchmark class-based accuracy after SS ARTMAP Stage 1 (dark bars) and Stage 2 (light bars) across 50 randomly chosen training sets. Stage 1 feature subsets are specified in Table 3.

9. Experimental studies of self-supervised learning

Research on aging summarized by Healey, Campbell, and Hasher (2008) supports the basic SS ARTMAP hypothesis that humans increase the number of attended input features over their lifetimes. Experiments comparing the reaction times of younger and older subjects show that older subjects are more easily distracted by task-irrelevant information. For decades these observations were interpreted negatively, as impairment of older subjects' ability to ignore distractors. Recent experiments (Kim, Hasher, & Zacks, 2007) suggest, however, that older subjects make better use of contextual information when its task relevance is not specified *a priori*. Younger subjects show a narrower focus of attention, which improves their performance on tasks that require ignoring distractors but impairs their performance on tasks for which peripheral information was, in fact, relevant.

Li and DiCarlo (2008) suggest that self-supervised learning by neurons in the inferotemporal cortex (ITC) may underlie view-invariant object recognition. A subject sees only one view of an object when it is paired with a label (Stage 1) but many views of the same object when the teaching signal is absent (Stage 2). Single-cell recordings in monkey ITC show increases in neurons' object selectivity and tolerance to different object orientations and positions as a result of Stage 2 learning.

Smith and Minda (2000) summarized thirty experiments on human pattern learning and classification. In each experiment, subjects unlearn in a characteristic way exemplars they had learned to classify perfectly during training. In Amis, Carpenter, Ersoy, and Grossberg (2009), ARTMAP simulations fit the experimental data and explain how classification patterns are learned and forgotten in real time. The model tests two competing hypotheses: (a) that the observed forgetting is the result of an unsupervised learning process; and (b) that the forgetting is the result of a corruption of long-term memory (LTM) traces by noise. The noise hypothesis (b) better fits the data and is also consistent with the mechanisms of self-supervised ARTMAP. Distributed learning during Stage 2 focuses LTM adaptation on input features whose values were unspecified during Stage 1. The unsupervised hypothesis (a) is equivalent to Stage 2 learning but with no novel input features specified, and so not enough additional learning occurs to model the data.

10. Related models of self-supervised learning

SS ARTMAP introduces a new definition of self-supervised learning: a system trained on labeled data with limited features continues to learn on an expanded but unlabeled feature set. As such, no prior work directly addresses this learning paradigm. There are, however, many studies related to this form of self-supervised learning.

Semi-supervised learning incorporates labeled and unlabeled inputs in its supervised training set, but unlike self-supervised learning, all inputs have the same number of specified feature values. Without any novel features from which to learn, semi-supervised learning systems use the unlabeled data to refine the model parameters defined using the labeled data. Reviews of semi-supervised learning (Chapelle, Schölkopf, & Zien, 2006; Zhu, 2005) find that its success is sensitive to underlying data distributions, in particular to whether data cluster or lie on a lower-dimensional manifold. Typically, models must be carefully selected and tuned for each problem space, using *a priori* knowledge of the domain. Chapelle et al. (2006) conclude that none of the semi-supervised models they reviewed is robust enough to be considered general purpose, and that semi-supervised learning remains an open problem. The main difficulty is that, whenever the unlabeled data are different enough from the labeled data to merit learning, it is unclear whether differences are useful, or whether they are the result of noise or other distortions that would damage the system's performance.

Another class of models admits inputs with different numbers of features, but in a different context than self-supervised ARTMAP learning. These methods assume that feature values are missing randomly and infrequently, and thus can be filled in based on a statistical analysis of other exemplars in the training set (Little & Rubin, 2002). Some applications add missing feature estimates to neural network architectures such as fuzzy ARTMAP (Lim, Leong, & Kuan, 2005) or multilayer perceptrons (Tresp, Neuneier, & Ahmad, 1995). Whereas feature estimates can succeed for problems with stationary statistics and redundant training data, challenging problems risk inappropriate learning on filled-in values. For this reason, exemplars with missing feature values are often excluded from the training set. In many situations, however, data points for an

entire cluster, class, or dataset have a common set of unspecified feature values. Structurally missing features are either ignored or substituted, and typical solutions are specific to a particular model architecture. For example, Chechik, Heitz, Elidan, Abbeel, and Koller (2008) created a support vector machine (SVM) that redefines the margin maximization objective to ignore unspecified feature values. Ishibuchi, Miyazaki, Kwon, and Tanaka (1993) modified a multilayer perceptron such that hidden layer nodes track feature value intervals rather than scalar values, representing a missing feature value as the interval [0,1].

11. Conclusion

Self-supervised ARTMAP defines a novel problem space that may be tested on almost any existing dataset designed for supervised learning. The system learns during an initial "textbook" supervised learning phase that is followed by a "real world" unsupervised learning phase. The novel aspect of the SS ARTMAP learning protocol is that inputs of the second, self-supervised phase specify more features than did the labeled inputs of the first phase. The expanded feature set provides to a learning system the capacity to teach the network important new information, so that accuracy may improve dramatically compared to that of the initial trained system, while avoiding performance deterioration from unlabeled data. New problem domains include applications in which a trained system performs in novel contexts, such as an image recognition problem in which an initial set of sensor data is later augmented on-line with unlabeled data from a new sensor.

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 CELEST, an NSF Science of Learning Center (SBE-0354378).

References

Amis, G.P., & Carpenter, G.A. (2007). Default ARTMAP 2. *Proceedings of the International Joint Conference on Neural Networks* (*IJCNN'07 – Orlando, Florida*), 777-782. http://techlab.bu.edu/members/gail/articles/152_Default_ARTMAP2_2007_.pdf

Amis, G.P., Carpenter, G.A., Ersoy, B., & Grossberg, S. (2009). Cortical learning of recognition categories: a resolution of the exemplar vs. prototype debate. *Technical Report, CAS/CNS TR-2009-002*, Boston, MA: Boston University.

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. (1991a). 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. (1991b). 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

Chapelle, O., Schölkopf, B., & Zien, A. (Eds.). (2006). *Semi-supervised learning.* Cambridge, MA: MIT Press.

Chechik, G., Heitz, G., Elidan, G., Abbeel, P., & Koller, D. (2008). Max Margin classification of incomplete data. *Journal of Machine Learning Research, 9*, 1-21. http://portal.acm.org/citation.cfm?id=1390681.1390682

Grossberg, S. (1968). Some nonlinear networks capable of learning a spatial pattern of arbitrary complexity. *Proceedings of the National Academy of Sciences, 59*, 368-372. http://cns.bu.edu/~steve/Gro1968PNAS59.pdf

Grossberg, S. (1976). Adaptive pattern classification and universal recoding. I: Parallel development and coding of neural feature detectors. *Biological Cybernetics, 23*, 121-134. http://cns.bu.edu/~steve/Gro1976BiolCyb_I.pdf

Healey, M.K., Campbell, K.L., & Hasher, L. (2008). Cognitive aging and increased distractibility: costs and potential benefits. In W.S. Sossin, J.-C. Lacaille, V.F. Castellucci, & S. Belleville (Eds.), *Progress in Brain Research, 129*, 353-363. http://www.ncbi.nlm.nih.gov/pubmed/18394486

Ishibuchi, H., Miyazaki, A., Kwon, K., & Tanaka, H. (1993). Learning from incomplete training data with missing values and medical application. *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'93)*, 1871-1874. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=717020

Kim, S., Hasher, K., & Zacks, R.T. (2007). Aging and a benefit of distractibility. *Psychonomic Bulletin and Review*. *14*, 301-305. http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2121579

Krashen, S. (2004, November). *Applying the comprehension hypothesis: Some suggestions*. Presented at 13th International Symposium and Book Fair on Language Teaching (English Teachers Association of the Republic of China), Taipei, Taiwan. http://www.sdkrashen.com/articles/eta_paper/

Li, N., & DiCarlo, J.J. (2008). Unsupervised natural experience rapidly alters invariant object representation in visual cortex. *Science*, *321*(5895), 1502-1507. http://www.sciencemag.org/cgi/content/abstract/321/5895/1502

Lim, C., Leong, J., & Kuan, M. (2005). A hybrid neural network system for pattern classification tasks with missing features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(4), 648-653. http://www2.computer.org/portal/web/csdl/doi/10.1109/TPAMI.2005.64

Little, R., & Rubin, D. (2002). *Statistical analysis with missing data* (2nd Ed.). New York: Wiley Interscience.

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

Smith, J.D., & Minda, J.P. (2000). Thirty categorization results in search of a model. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *26*, 3-27.

Tresp, V., Neuneier, R., & Ahmad, S. (1995). Efficient methods for dealing with missing data in supervised learning. *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, *7*, 689-696. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.62.6142&rep=rep1&type=pdf

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

Zhu, X. (2005). Semi-supervised learning literature survey. *Technical Report 1530, Computer Sciences, University of Wisconsin-Madison.* Updated July 19, 2008. http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf

Appendix A

Self-supervised ARTMAP algorithm

Figure 16 and Table 4 summarize self-supervised ARTMAP notation, and Table 5 lists default parameter values.

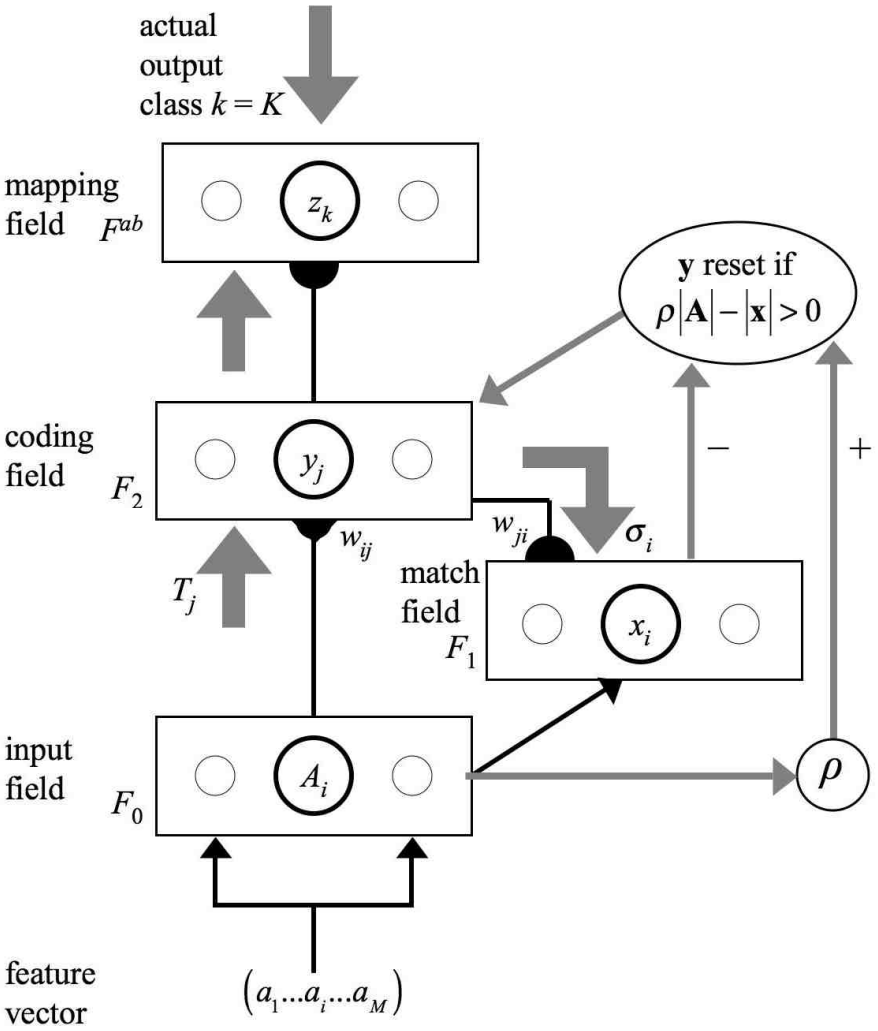


Figure 16. Self-Supervised ARTMAP network.

Table 4

Self-Supervised ARTMAP Notation

Notation	Description
i	input component index
j	coding node index
k	output class index
M	number of input features specified during Stage 2 learning and testing
$\bar{M}$	number of features specified during Stage1 learning, $\bar{M} \le M$
$\mathbf{a}$	feature vector (a_i), $0 \le a_i \le 1$, $i = 1,\ldots,M$
$\mathbf{A}$	complement coded input vector (A_i), $i = 1,\ldots,2M$
K	actual output class for an input
K'	predicted output class
J	chosen coding node (winner-take-all)
C	number of activated coding nodes
$\mathbf{\Lambda}, \mathbf{\Lambda}'$	coding node index subsets
T_j	signal from input field to coding node j
$\mathbf{y}$	coding field activation pattern (y_j)
σ_i	signal from coding field to match field node i
σ_k	signal from coding field to output node k
w_{ij}	$F_0 \to F_2$ bottom-up weights from input node i to coding node j
w_{ji}	$F_2 \to F_1$ top-down weights from coding node j to match field node i

Φ_j	degree of undercommitment of coding node *j*
κ_j	output class associated with coding node *j*
ρ	vigilance variable
$\wedge$	component-wise minimum (fuzzy intersection): $(\mathbf{p} \wedge \mathbf{q})_i = \min\{ p_i, q_i \}$
$\vee$	component-wise maximum (fuzzy union): $(\mathbf{p} \vee \mathbf{q})_i = \max\{ p_i, q_i \}$
$\lvert \cdot \rvert$	vector size (L_1-norm): $\lvert \mathbf{p} \rvert \equiv \sum_i \lvert p_i \rvert$
$[\cdot]^+$	rectification: $[p]^+ = \max\{ p, 0 \}$

Table 5

Self-supervised ARTMAP parameters

Name	**Para-meter**	**Range**	**Default value**	**Value for Boston benchmark**	**Notes**
signal rule parameter	α	$(0,1)$	0.01	0.01	$\alpha = 0^+$ maximizes code compression
Stage 1 learning fraction	$\bar{\beta}$	$(0,1]$	1.0	1.0	$\bar{\beta} = 1$ implements fast learning
Stage 2 learning fraction	$\bar{\bar{\beta}}$	$[0,1]$	0.01	0.01	
signal rule undercommitment factor	γ	$(0,1-\alpha]$	$1-\alpha$	$1-\alpha$	Setting $\gamma \le (1-\alpha)$ ensures that $T_j \le M$ for the IG CAM Rule
match tracking parameter	ε	$(-1,1)$	-0.01	-0.01	$\varepsilon < 0$ (MT–) codes inconsistent cases
baseline vigilance	$\bar{\rho}$	$[0,1]$	0.0	0.0	$\bar{\rho} = 0$ maximizes code compression
CAM rule power Stage 2 & testing	p	$(0,\infty]$	2	2	IG CAM Rule converges to winner-take-all as $p \to \infty$
# Stage 1 training epochs	$\bar{E}$	≥ 1	1	1	$\bar{E} = 1$ simulates on-line learning
# Stage 2 training epochs	$\bar{\bar{E}}$	≥ 1	1	2	
# voting systems	V	≥ 1	5	5	
# cross-validation strips	F	≥ 1	4	4	
maximum # Stage 1 training points per class	P	≥ 1	250	250	

A.1. Self-supervised learning – Stage 1

Each input $\mathbf{a} = \left(a_1 ... a_i ... a_M\right)$ is associated with an output class *K*. Feature values $a_i \ \left(i = 1 ... \bar{M}\right)$ are specified. Coding field activation is winner-take-all.

A.1.1. Initialize the system

All coding nodes *j* are uncommitted.

1. Set the initial number of coding nodes that have been activated one or more times: $C = 0$
2. Set weights to their initial values: $w_{ij} = 1$
3. Set the initial coding field signals to all coding nodes: $T_j = \alpha M$
4. Set the initial degree of undercommitment of coding nodes: $\Phi_j = 1$

A.1.2. Stage 1 learning

Repeat the following loop $\bar{E}$ times, where $\bar{E}$ is the number of Stage 1 training epochs.

1. Choose a new training pair $\mathbf{a} \rightarrow K$
2. Set vigilance at its baseline value: $\rho = \bar{\rho}$
3. Complement code the $\bar{M}$ specified features of **a**:

   For $i = 1,..., \bar{M}$ $\quad A_i = a_i, \ A_{i+M} = 1 - a_i$

   For $i = \bar{M} + 1,..., M$ $\quad A_i = A_{i+M} = 1$

   $\left|\mathbf{A}\right| = 2M - \bar{M}$
4. Calculate the coding field signals T_j :

$$T_j = \frac{\left(2M - \left|\mathbf{A}\right|\right) - \left(\left|\mathbf{w}_j\right| - \left|\mathbf{A} \wedge \mathbf{w}_j\right|\right)}{1 - \gamma \Phi_j} - \alpha\left(M - \left|\mathbf{w}_j\right|\right) \quad \text{for } j = 1...C$$

5. Define the initial index set $\boldsymbol{\Lambda}$ of candidate nodes:

$$\boldsymbol{\Lambda} = \left\{j : \ T_j > \alpha M\right\} \subseteq \left\{1,...,C\right\}$$

6. If $\boldsymbol{\Lambda}$ is empty, choose an uncommitted node: Go to Step 8
7. If $\boldsymbol{\Lambda}$ is not empty, choose a candidate node:

   Choose coding field winner: $J = \arg\max_{j \in \boldsymbol{\Lambda}} \left\{T_j\right\}$, breaking ties randomly

   Activate the coding field (winner-take-all): $y_J = 1$, all other $y_j = 0$

   If $\dfrac{\left|\mathbf{A} \wedge \mathbf{w}_J\right|}{\left|\mathbf{A}\right|} < \rho$, *J* fails to meet the matching criterion:

   Reset *J*: Remove *J* from $\boldsymbol{\Lambda}$

   Continue search: Go to Step 6

   If $\kappa_J = K$, *J* predicts the correct class. Learning:

   Supervised learning of weights in $F_0 \rightarrow F_2$ (bottom-up) paths and $F_2 \rightarrow F_1$

(top-down) paths: $w_{iJ} = w_{Ji} = w_{iJ}^{\text{old}} - \bar{\beta}\left[w_{iJ}^{\text{old}} - A_i \right]^+$ for i = 1,…,2M.

With fast learning $\left(\bar{\beta} = 1\right)$, $w_{iJ} = w_{Ji} = A_i \wedge w_{iJ}^{\text{old}}$.

Update the degree of undercommitment of node J:

$$\Phi_J = \frac{1}{M} \sum_{i=1}^{M} \left[w_{iJ} - \left(1 - w_{i+M,J}\right) \right]^+$$

Continue to the next training pair: Go to Step 1

If $\kappa_J \neq K$, J predicts an incorrect class. Match tracking and reset:

Match track: $\rho = \dfrac{\left|\mathbf{A} \wedge \mathbf{w}_J\right|}{\left|\mathbf{A}\right|} + \varepsilon$

Reset J: Remove J from $\mathbf{\Lambda}$

Continue search: Go to Step 6

8. Activate a new node:

Add 1 to the activated node count C

$w_{iC} = A_i \ \left(i = 1,...,2M \right)$, $\kappa_C = K$

Continue to next training pair: Go to Step 1

A.2. Self-supervised learning – Stage 2

All feature values a_i are specified $\left(i = 1...M\right)$. Coding field activation is distributed, and learning is slow. No output classes are specified. The vigilance parameter $\rho = 0$ during Stage 2 and during testing, so weight updates in top-down pathways are not computed in the algorithm.

Repeat the following loop $\bar{\bar{E}}$ times, where $\bar{\bar{E}}$ is the number of Stage 2 training epochs.

1. Choose a new input **a**
2. Reset coding field activation: **y** = **0**
3. Complement code all features of **a**:

For $i = 1,...,M$: $A_i = a_i$, $A_{i+M} = 1 - a_i$

$\left|\mathbf{A}\right| = M$

4. Calculate coding field inputs for $j = 1...C$:

$$T_j = \frac{\left(2M - \left|\mathbf{A}\right|\right) - \left(\left|\mathbf{w}_j\right| - \left|\mathbf{A} \wedge \mathbf{w}_j\right|\right)}{1 - \gamma \Phi_j} - \alpha\left(M - \left|\mathbf{w}_j\right|\right)$$

5. Define index subsets of F_2 nodes j:

Nodes with input above inputs to uncommitted nodes: $\mathbf{\Lambda} = \left\{ j : T_j > \alpha M \right\}$

Point box nodes with R_j identical to **a**: $\mathbf{\Lambda}' = \left\{ j : \mathbf{w}_j = \mathbf{A} \right\}$

6. If $\mathbf{\Lambda}$ is empty, go to Step 1 (no learning).

7. Activate the distributed code **y** at the field F_2

If $\mathbf{\Lambda}'$ is not empty, activation is distributed evenly at nodes whose point boxes equal the input **a**:

$$y_j = \begin{cases} \dfrac{1}{|\mathbf{\Lambda}'|} & \text{for } j \in \mathbf{\Lambda}' \\ 0 & \text{otherwise} \end{cases}$$

If $\mathbf{\Lambda}'$ is empty, F_2 activation **y** is distributed:

$$y_j = \begin{cases} \dfrac{\left[M - T_j \right]^{-p}}{\sum_{\lambda \in \mathbf{\Lambda}} \left[M - T_\lambda \right]^{-p}} & \text{for } j \in \mathbf{\Lambda} \\ 0 & \text{otherwise} \end{cases}$$

If **a** is in one or more boxes , activation y_j is distributed among the corresponding nodes, with y_j largest where the boxes R_j are smallest. If **a** is not in any box, y_j is distributed according to $d\left(R_j, \mathbf{a}\right)$, with y_j largest where **a** is closest to R_j.

8. Self-supervised distributed instar learning:

$$w_{ij} = w_{ij}^{\text{old}} - \overline{\overline{\beta}} \left[y_j - \left(1 - w_{ij}^{\text{old}}\right) - A_i \right]^+ \text{ for } i = 1,...,2M,\ j \in \mathbf{\Lambda}$$

9. Update the degree of undercommitment for nodes $j \in \Lambda$:

$$\Phi_j = \frac{1}{M} \sum_{i=1}^{M} \left[w_{ij} - \left(1 - w_{i+M,j}\right) \right]^+$$

10. Continue to the next input: Go to Step 1

A.3. Testing

All feature values a_i are specified. Coding field activation is distributed. No learning occurs.

1. Choose a new test input **a**
2. Complement code all features of **a**:

   For $i = 1,...,M$: $A_i = a_i,\ A_{i+M} = 1 - a_i$

   $|\mathbf{A}| = M$
3. Reset coding field activation: **y** = **0**
4. Calculate coding field inputs for $j = 1...C$:

$$T_j = \frac{\left(2M - |\mathbf{A}|\right) - \left(|\mathbf{w}_j| - |\mathbf{A} \wedge \mathbf{w}_j|\right)}{1 - \gamma \Phi_j} - \alpha \left(M - |\mathbf{w}_j|\right)$$

5. Define index subsets of F_2 nodes *j*:

   Nodes with input above inputs to uncommitted nodes: $\mathbf{\Lambda} = \left\{ j : T_j > \alpha M \right\}$

Point box nodes with R_j identical to **a**: $\mathbf{\Lambda}' = \left\{ j : \mathbf{w}_j = \mathbf{A} \right\}$

6. If $\mathbf{\Lambda}$ is empty, go to Step 1 (no prediction).
7. Activate the distributed code **y** at the field F_2

If $\mathbf{\Lambda}'$ is not empty, activation is distributed evenly at nodes whose point boxes equal the input **a**:

$$y_j = \begin{cases} \dfrac{1}{|\mathbf{\Lambda}'|} & \text{for } j \in \mathbf{\Lambda}' \\ 0 & \text{otherwise} \end{cases}$$

If $\mathbf{\Lambda}'$ is empty, F_2 activation **y** is distributed:

$$y_j = \begin{cases} \dfrac{\left[M - T_j \right]^{-p}}{\sum\limits_{\lambda \in \mathbf{\Lambda}} \left[M - T_\lambda \right]^{-p}} & \text{for } j \in \mathbf{\Lambda} \\ 0 & \text{otherwise} \end{cases}$$

8. Activate the mapping field: $\sigma_k = \sum\limits_{\left\{ j \in \mathbf{\Lambda}:\ \kappa_j = k \right\}} y_j$
9. Make an output class prediction: $K' = \arg\max\limits_k \left\{ \sigma_k \right\}$, breaking ties randomly
10. If the ground-truth label *K* is available, record prediction accuracy
11. Continue to the next test input: Go to Step 1

A.4. Voting

1. Apply self-supervised learning Stages 1 and 2 to *V* ARTMAP systems, each with a different sampling or ordering of inputs
2. During testing:
   Generate a class prediction for each voter
   Predict the class with the most votes

A.5. Cross-validation of image dataset (Boston testbed)

Given a dataset of labeled and unlabeled input vectors, divided into *F* disjoint strips.
For each strip *f* :

A.5.1. Stage 1 training

Create a training data source by concatenating labeled inputs from the other $F - 1$ strips (all except for *f*). If a fully featured dataset is being used for evaluation (e.g., the Boston testbed), identify a feature subset to use in Stage 1. Remove all other feature values from each Stage 1 training input.

Train (Stage 1) *V* ARTMAP voters on a random selection of at most *P* inputs per class.

A.5.2. Stage 2 training

Train (Stage 2) each voter on a random ordering of all (fully featured) inputs in strip *f*, including labeled and unlabeled inputs.

A.5.3. Testing

Generate ensemble class predictions on the fully featured, labeled inputs in *f*.
Record per-class and average class accuracy.