Default ARTMAP 2

Gregory P. Amis and Gail A. Carpenter

***Abstract*—Default ARTMAP combines winner-take-all category node activation during training, distributed activation during testing, and a set of default parameter values that define a ready-to-use, general-purpose neural network system for supervised learning and recognition. Winner-take-all ARTMAP learning is designed so that each input would make a correct prediction if re-presented immediately after its training presentation, passing the "next-input test." Distributed activation has been shown to improve test set prediction on many examples, but an input that made a correct winner-take-all prediction during training could make a different prediction with distributed activation. Default ARTMAP 2 introduces a distributed next-input test during training. On a number of benchmarks, this additional feature of the default system increases accuracy without significantly decreasing code compression. This paper includes a self-contained default ARTMAP 2 algorithm for implementation.**

I. Introduction: ART and ARTMAP Systems

Adaptive Resonance Theory (ART) neural networks model real-time prediction, search, learning, and recognition. ART networks function both as models of human cognitive information processing [1]–[4] and as neural systems for technology transfer [5], [6]. A neural computation central to both the scientific and the technological analyses is the *ART matching rule* [7], which models the interaction between top-down expectation and bottom-up input, thereby creating a focus of attention which, in turn, determines the nature of coded memories.

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 [8], ART-EMAP [9], ARTMAP-IC [10], Gaussian ARTMAP [11], and distributed ARTMAP [3], [12]. Comparative analysis of these systems led to the identification of the *default ARTMAP* network, which features robust performance in many application domains [5], [13]. Default ARTMAP 2 adds the principled feature of distributed next-input-test training while retaining the essential characteristics of design simplicity and complete parameter specification.

This work was supported in part by grants from the Air Force Office of Scientific Research (AFOSR F49620-01-1-0423), the National Science Foundation (NSF SBE-0354378), the Office of Naval Research (ONR N00014-01-1-0624), and the National Geospatial-Intelligence Agency (NMA201-01-1-2016).

The authors are with the Department of Cognitive and Neural Systems, Boston University, Boston, MA 02215 USA (phone: 617-353-9481; fax: 617-353-7755; e-mail: [gamis,gail]@cns.bu.edu).

Technical Report CAS/CNS TR-2007-003, Boston, MA: Boston University.

Web: http://cns.bu.edu/~gail/Default_ARTMAP2_2007_.pdf

II. Default ARTMAP

The character of their code representations, distributed vs. winner-take-all, is a primary factor differentiating various ARTMAP networks. The original models [8], [14] employ winner-take-all coding during training and testing, as do many subsequent variations and the majority of ART systems that have been transferred to technology. Default ARTMAP codes a training input as a winner-take-all activation pattern, but codes a test input as a distributed activation pattern. For distributed coding, the transformation of the filtered bottom-up input to an activation pattern across a field of nodes is defined by the *increased-gradient CAM rule* [12]. The default network also implements the *MT–* search algorithm [10] and sets the *baseline vigilance parameter* $\bar{\rho}$ equal to zero, for maximal code compression. Other design choices for default ARTMAP include *fast learning*, whereby weights converge to asymptote on each learning trial; single-epoch training, which emulates on-line learning; a *choice-by-difference* signal function [15] from the input field to the coding field; and four-fold cross-validation.

ARTMAP's capacity for fast learning implies that the system can incorporate information from examples that are important but infrequent and can be trained incrementally. Fast learning also causes each network's memory to vary with the order of input presentation during training. *Voting* across several networks trained with different orderings of a given input set takes advantage of this feature, typically improving performance and reducing variability as well as providing a measure of confidence in each prediction [8]. While the number of voting systems is, in general, a free parameter, five voters have proven to be sufficient for many applications. Default ARTMAP thus trains five voting networks for each training set combination.

III. Default ARTMAP 2: Passing the Distributed Next-Input Test

When trained using winner-take-all activation, ARTMAP is guaranteed to pass the *next-input test*: immediately following training on a given input-class pair (**a**, *K*), the system correctly predicts class *K* when presented with input **a**. There is no guarantee, however, that **a** would predict *K* if activation were distributed. Default ARTMAP 2 incorporates a *distributed next-input test*, in order to allow the system to learn as much as possible from each training input **a**. When presented with **a**, default ARTMAP 2 first

learns in winner-take-all mode. The network then switches to distributed activation to confirm that **a** now predicts the correct class. If not, the system reverts to winner-take-all mode to learn more from **a**, just as if the winner-take-all prediction had been incorrect. Default ARTMAP 2 hereby improves accuracy on test sets, while maintaining code compression comparable to the original default ARTMAP 1.

IV. ARTMAP GEOMETRY

A geometric interpretation of default ARTMAP represents each category as a box in M-dimensional space, where M is the number of components of input **a**. Consider an input set that consists of 2-dimensional vectors **a**. With complement coding,

$$\mathbf{A} = (\mathbf{a}, \mathbf{a}^c) = (a_1, a_2, 1 - a_1, 1 - a_2). \quad (1)$$

Each category j then has a geometric representation as a rectangle R_j (Fig. 1). Following (1), a complement-coded weight vector $\mathbf{w}_j$ takes the form:

$$\mathbf{w}_j = (\mathbf{u}_j, \mathbf{v}_j^c), \quad (2)$$

where $\mathbf{u}_j$ and $\mathbf{v}_j$ are 2-dimensional vectors. Vector $\mathbf{u}_j$ defines the lower left corner of a category rectangle R_j, and $\mathbf{v}_j$ defines the upper right corner. The size of R_j is:

$$|R_j| \equiv |\mathbf{v}_j - \mathbf{u}_j|, \quad (3)$$

which is equal to the height plus the width of R_j. In fact, $|R_j| = M - |\mathbf{w}_j|$.

In a fast-learn default ARTMAP system, $\mathbf{w}_J^{(\text{new})} = \mathbf{A} = (\mathbf{a}, \mathbf{a}^c)$ when J is an uncommitted node. The corners of $R_J^{(\text{new})}$ are then **a** and $(\mathbf{a}^c)^c = \mathbf{a}$. Hence $R_J^{(\text{new})}$ is just the point **a**. Learning increases the size of R_J, which grows as the size of $\mathbf{w}_J$ shrinks. Vigilance ρ determines the maximum box size, with $|R_j| \le 2(1 - \rho)$. During each fast-learning trial, R_J expands to $R_J^{(\text{old})} \oplus \mathbf{a}$, the minimum rectangle containing $R_J^{(\text{old})}$ and **a**. However, before R_J can expand to include **a**, reset chooses another category if $|R_J^{(\text{old})} \oplus \mathbf{a}|$ is too large. With fast learning, R_j is the smallest rectangle that encloses all points **a** that have chosen category j without reset.

Input to a coding node is governed by the choice-by-difference function:

$$T_j = \left|\mathbf{A} \wedge \mathbf{w}_j\right| + (1 - \alpha)\left(M - \left|\mathbf{w}_j\right|\right) \quad (4)$$

Geometrically, T_j may be written as a function of the input dimension (M), the city-block distance $d(R_j,\mathbf{a})$ between R_j and **a**, and the size of R_j:

$$T_j = M - d(R_j, \mathbf{a}) - \alpha\left|R_j\right|. \quad (5)$$

Thus, the input to category node j is maximal when $\mathbf{a} \in R_j$ ($d(R_j, \mathbf{a})=0$), and smaller category boxes are favored over larger ones.

Fig. 1 illustrates how default ARTMAP 2 may differ from default ARTMAP 1. The original algorithm (Fig. 1a–c) would incorrectly place point 7 in class A during distributed testing. With the distributed next-input test, default ARTMAP 2 (Fig. 1d–e) anticipates and corrects this error. Default ARTMAP 2 tests the distributed prediction following each learning cycle. If the prediction is incorrect, vigilance is raised, and the system learns again from the current input.

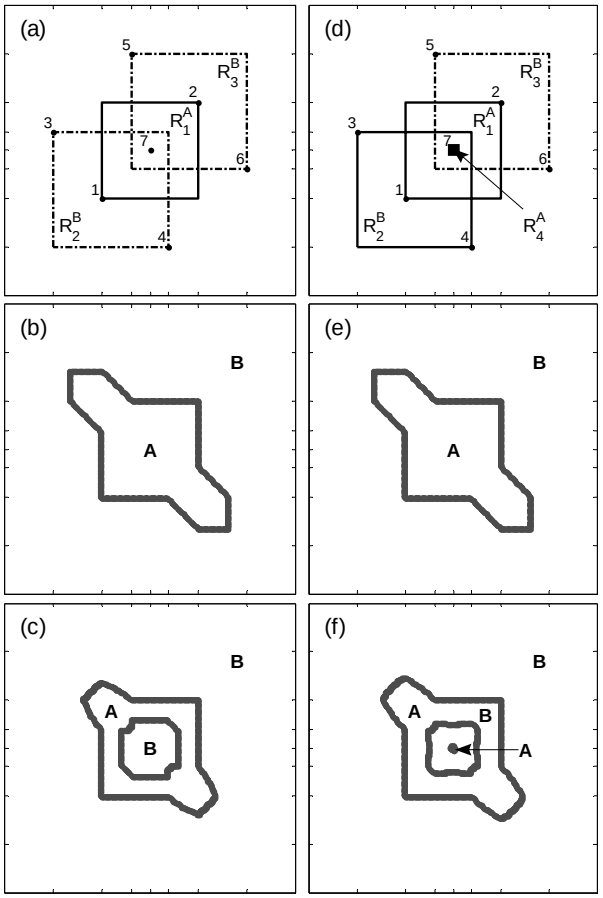

Fig. 1. Training points 1, 2, and 7 are associated with class A; and points 3–6 are associated with class B. (a) Default ARTMAP 1 category boxes after training on points 1…7. R_1 is associated with class A; R_2 and R_3 are associated with class B; and $|R_1| < |R_2| = |R_3|$. (b) Test set decision boundary with winner-take-all coding. (c) Test set decision boundary with distributed coding. Training point 7 is incorrectly placed in class B. (d–f) Default ARTMAP 2 creates the category box R_4, and correctly predicts class A for point 7 during testing.

V. BENCHMARK PERFORMANCE

On a variety of benchmark problems, default ARTMAP 2 improves classification accuracy without requiring significantly more coding nodes than the original system.

The *circle-in-the-square* (CIS) benchmark requires a system to identify those points in a unit square that lie within a circle placed at the center of the square and occupying half the area. The *Boston* benchmark is derived from satellite imagery of northeast Boston and suburbs and requires a system to classify pixels as having the label ocean, park, residential, industrial, etc., based on 41 features extracted from satellite data [4]. The *Frey-Slate letter recognition* benchmark requires a system to identify an input exemplar as one of 26 capital letters A–Z based on 16 numerical features extracted from black-and-white pixel images [16]. These data sets are available from http://cns.bu.edu/techlab.

Tables I and II and Fig. 2 summarize the new system's performance on these benchmarks compared to default ARTMAP 1. Each simulation uses default parameters (Table IV), with the exception of the number of voters V, and the number of training epochs E. In simulations without voting and with a single training epoch, default ARTMAP 2 improves the classification accuracy of the original default ARTMAP, from 1.8% for circle-in-the-square (using 100 training points) to 3.5% for Frey-Slate, while using 4−10% more coding nodes. Using five voters and two training epochs improves the testing accuracy of both versions of default ARTMAP, though default ARTMAP 2 still shows better classification accuracy, from 0.5% (CIS 100) to 1.6% (Boston).

VI. Default ARTMAP 2 Algorithm

Fig. 3 and Table III summarize default ARTMAP notation, and Table IV lists default parameter values. A user who wishes to explore network variations might begin by varying the baseline vigilance, $\bar{\rho}$. In some cases, higher values of $\bar{\rho}$ increase predictive accuracy but may decrease code compression.

A. Classification Methodology

This section outlines a canonical classification procedure for training and evaluating supervised learning systems, including ARTMAP.

A.1 List output classes for the supervised learning problem.

A.2 If possible, estimate an *a priori* distribution of output classes.

A.3 If not provided, create a ground truth set for each class by assigning output labels to a designated set of input vectors.

A.4 Divide the ground truth set into F disjoint subsets.

A.5 In each of the F subsets, designate either all ground

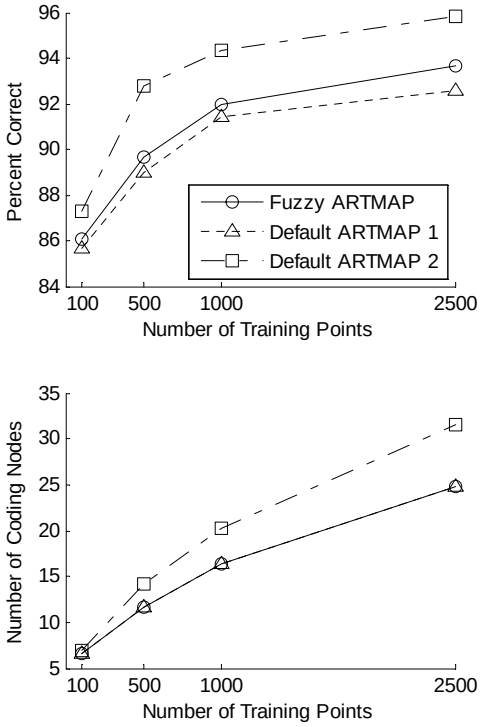

Fig. 2: Performance of ARTMAP on the circle-in-square benchmark for different training set sizes.

TABLE II
BENCHMARK PERFORMANCE (FIVE VOTERS, TWO TRAINING EPOCHS)

	Percent Correct		
Data Set	Default ARTMAP 1	Default ARTMAP 2	Difference
CIS 100	90.7% ± 2.5	91.3% ± 2.1	0.5% ± 1.1
CIS 1,000	95.8% ± 1.4	97.0% ± 0.5	1.2% ± 1.4
Boston	92.2% ± 1.6	93.8% ± 1.2	1.6% ± 0.8
Frey-Slate	91.3% ± 0.5	92.6% ± 0.2	1.3% ± 0.5

TABLE I[a]
BENCHMARK PERFORMANCE (NO VOTING, ONE TRAINING EPOCH)

	Percent Correct[b]			Number of Coding Nodes		
Data Set	Default ARTMAP 1	Default ARTMAP 2	Difference[c]	Default ARTMAP 1	Default ARTMAP 2	Difference
CIS (100 train pts)	86.0% ± 6.9	87.8% ± 5.1	1.8% ± 4.6	6.6 ± 2.3	7.0 ± 2.3	0.3 ± 0.7
CIS (1,000 train pts)	91.2% ± 4.9	94.3% ± 1.4	3.1% ± 4.6	16.3 ± 2.9	20.1 ± 3.2	3.8 ± 2.9
Boston	91.5% ± 2.8	93.5% ± 2.1	2.0% ± 1.8	13.2 ± 1.6	14.1 ± 1.5	1.0 ± 0.8
Frey-Slate	84.7% ± 1.2	88.2% ± 0.5	3.5% ± 1.2	562 ± 16	618 ± 19	57 ± 18

[a] The values presented are the mean ± the standard deviation over trials of different training points (CIS) or different training point presentation order (Boston and Frey-Slate).

[b] Because the Boston data set does not have an equal number of test points for each class, the percent correct values presented here are normalized to control for the mixture of classes. Both CIS and Frey-Slate have an equal number of test points for each class.

c Difference values are not simply a subtraction of the default ARTMAP 1 mean from the default ARTMAP 2 mean. Rather, for each training set order, performance of the two systems was calculated. The values here represent mean per-trial differences.

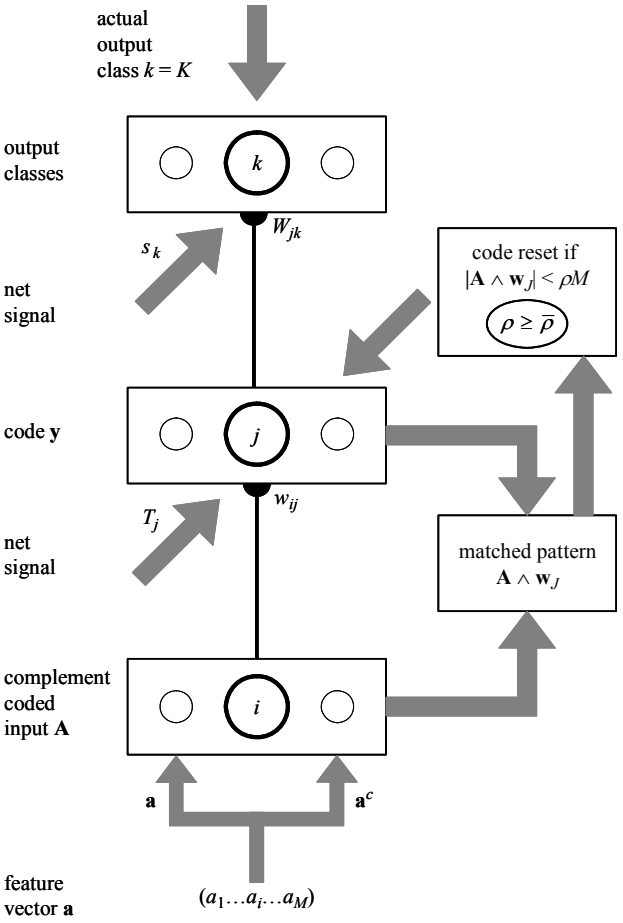

Fig. 3: Default ARTMAP notation.

truth inputs in that set; or P randomly chosen labeled inputs for each output class (or all inputs in a given class if fewer than P have been labeled). Fix random orderings of designated inputs in each subset.

A.6 Choose one subset for validation, one for testing, and the rest for training.

A.7 Train V systems (*voters*), each with E presentations of input vectors from one of the ordered training sets (Section VI.B).

A.8. For each voter, choose parameters by validation (if parameter choice is required).

A.9. Present to each voter all test set inputs. Produce an output class prediction σ_k for each test input (Section VI.C).

A.10. Sum the distributed output class predictions across the V voters.

A.11. Label inputs by one of three methods (breaking ties by random choice):

A.11.a. *Baseline*: Assign the input to the output class k with the largest summed prediction.

A.11.b. *Prior probabilities*: Select an output class at random according to the estimated *a priori* distribution in the data set. Assign that class label to the still-unlabeled input with the largest summed prediction for this class.

A.11.c. *Validation*: Bias the summed output class distribution, evaluating performance on the validation set. One such method [5] selects decision thresholds for each output class, with an upper bound of 10% set for each false alarm rate. Alternatively, the distributed prediction of each voter (or of the sum) could be weighted by a steepest descent algorithm. Use the biased summed distribution to label the input by the baseline or prior probabilities method.

A.12. *Post-training output class adjustments:*

A.12.a. *Standard post-processing methods*: Mapping tasks, for example, may benefit from local image smoothing. Post-processing for speckle removal may be implemented as a simple voting filter which assigns to each pixel the label originally assigned to a majority of its eight neighbors plus three copies of itself.

A.12.b. *Class distribution adjustment*: Starting with the output class predictions produced by any method (Step A.11), target distribution percentages may be adjusted up or down (e.g., based on inspection of resulting classes), and class labels recomputed by the prior probabilities method.

A.12.c. *False alarm rate adjustment*: A decision threshold for an over-represented class may be increased to reduce the validation set false alarm rate.

A.13. *Classifier evaluation:* Compute average performance statistics across all combinations of training subsets (each with V voters). Classifier evaluation measures include test set output class distributions, hit and false alarm rates for each class, overall accuracy on the test set, performance variability between tasks, product appearance (e.g., for mapping, overall and by overlays for each class), and degree of improvement by post-processing.

B. Default ARTMAP 2 Training, with Distributed Next-Input Test

B.1. Complement code M-dimensional training set feature vectors $\mathbf{a}$ to produce $2M$-dimensional input vectors $\mathbf{A}$:

$$\mathbf{A} \equiv (\mathbf{a}, \mathbf{a}^c) \text{ and } |\mathbf{A}| = M$$

B.2. Set initial values: $w_{ij} = 1$, $W_{jk} = 0$, $C = 1$

B.3. Select the first input vector $\mathbf{A}$, with associated actual output class K

B.4. Set initial weights for the newly committed coding node $j = C$:

$$\mathbf{w}_C = \mathbf{A}$$

$$\mathbf{W}_{CK} = 1$$

B.5. Set vigilance ρ to its baseline value:

$$\rho = \overline{\rho}$$

TABLE III
DEFAULT ARTMAP NOTATION

NOTATION	DESCRIPTION
i	input component index
j	coding node index
k	output class index
M	number of input features
$\mathbf{a}$	feature vector (a_i), $0 \le a_i \le 1$
$\mathbf{A}$	complement coded input vector: $\mathbf{A} \equiv (\mathbf{a}, \mathbf{a}^c)$
K	actual output class of training input
$\mathbf{y}$	coding field activation pattern (CAM): (y_j)
J	chosen coding node (winner-take-all)
C	number of committed coding nodes
$\mathbf{\Lambda}$, $\mathbf{\Lambda}'$	committed node subsets
T_j	signal from input field to coding node j
σ_k	signal from coding field to output node k
$\mathbf{w}_j$	coding node weight vector j: (w_{ij})
$\mathbf{W}_k$	output class weight vector k: (W_{jk})
ρ	vigilance variable
$\wedge$	component-wise minimum (fuzzy intersection): $(\mathbf{p} \wedge \mathbf{q})_i = \min(p_i, q_i)$
$\lvert\cdot\rvert$	vector size (L_1-norm): $\lvert\mathbf{p}\rvert \equiv \sum_i \lvert p_i \rvert$
$\mathbf{p}^c$	vector complement: $(\mathbf{p}^c)_i \equiv 1 - p_i$

TABLE IV
DEFAULT PARAMETER VALUES

NAME	PARA-METER	RANGE	DEFAULT VALUE	NOTES
signal rule parameter	α	$(0,\infty)$ $(0,1)$	0.01	$\alpha = 0^+$ maximizes code compression
learning fraction	β	[0,1]	1.0	$\beta = 1$ implements fast learning
match tracking	ε	(-1,1)	– 0.001	$\varepsilon < 0$ (MT–) codes inconsistent cases
baseline vigilance	$\bar{\rho}$	[0,1]	0.0	$\bar{\rho} = 0$ maximizes code compression
CAM rule power	p	$(0,\infty]$	1.0	Increased Gradient (IG) CAM rule converges to WTA as $p \to \infty$
# training epochs	E	≥ 1	1	$E = 1$ simulates on-line learning
# data subsets	F	≥ 3	4	F-fold cross-validation
# voting systems	V	≥ 1	5	

and reset the code:

$\mathbf{y} = \mathbf{0}$

B.6. Select the next input vector $\mathbf{A}$, with associated actual output class K (until the last input of the last training epoch)

B.7. Calculate signals to committed coding nodes $j = 1 \ldots C$:

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

B.8. *Search order*: Sort the committed coding nodes with $T_j > \alpha M$ in order of T_j values (max to min)

B.9. Search for a coding node J that meets the matching criterion and predicts the correct output class K, as follows:

B.9.a. *Code*: For the next sorted coding node ($j = J$) that meets the matching criterion

$$\left(\frac{\left|\mathbf{A} \wedge \mathbf{w}_J\right|}{M} \ge \rho\right), \text{ set } y_J = 1 \text{ (WTA)}$$

B.9.b. Output class prediction:

$$\sigma_k = \sum_{j=1}^{C} W_{jk} y_j = W_{Jk}$$

B.9.c. *Correct prediction*: If the active code J predicts the actual output class K ($\sigma_K = W_{JK} = 1$), go to Step B.11 (learning)

B.9.d *Match tracking*: If the active code J fails to predict the correct output class ($\sigma_K = 0$), raise vigilance:

$$\rho = \frac{\left|\mathbf{A} \wedge \mathbf{w}_J\right|}{M} + \varepsilon$$

Return to Step B.9.a (continue search)

B.10. After unsuccessfully searching the sorted list, increase C by 1 (add a committed node)

Return to Step B.4

B.11. *Learning*: Update coding weights:

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

B.12. *Distributed next-input test*: Verify that the input makes the correct prediction with distributed coding.

B.12.a *Make prediction*: Generate an output class prediction K' for the current training input $\mathbf{A}$ using distributed activation, as prescribed for testing (Section VI.C)

$K' = \arg\max_k \sigma_k$

B.12.b. *Correct prediction*: If distributed activation predicts class K, return to Step B.5 (next input).

B.12.c. *Match tracking*: If distributed activation fails to predict the correct output class ($K' \neq K$), raise vigilance:

$$\rho = \frac{\left| \mathbf{A} \wedge \mathbf{w}_J \right|}{M} + \varepsilon$$

Return to Step B.9.a (continue search)

C. Default ARTMAP Testing (Distributed Code)

C.1. Complement code M-dimensional test set feature vectors **a** to produce $2M$-dimensional input vectors **A**

C.2. Select the next input vector **A**, with associated actual output class K

C.3. Reset the code: $\mathbf{y} = \mathbf{0}$

C.4. Calculate signals to committed coding nodes $j = 1 \ldots C$:

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

C.5. Let $\mathbf{\Lambda} = \{ \lambda = 1 \ldots C : T_\lambda > \alpha M \}$ and

$\mathbf{\Lambda}' = \{ \lambda = 1 \ldots C : T_\lambda = M \} = \{ \lambda = 1 \ldots C : \mathbf{w}_j = \mathbf{A} \}$

C.6. *Increased Gradient (IG) CAM Rule*:

C.6.a. *Point box case*: If $\mathbf{\Lambda}' \neq \phi$ (i.e., $\mathbf{w}_j = \mathbf{A}$ for some j), set $y_j = \frac{1}{|\mathbf{\Lambda}'|}$ for each j $\in \mathbf{\Lambda}'$

C.6.b. If $\mathbf{\Lambda}' = \phi$, set

$$y_j = \frac{\left[\frac{1}{M - T_j} \right]^p}{\sum_{\lambda \in \mathbf{\Lambda}} \left[\frac{1}{M - T_\lambda} \right]^p} \text{ for each j} \in \mathbf{\Lambda}$$

C.7. Calculate distributed output class predictions:

$$\sigma_k = \sum_{j=1}^{C} W_{jk} y_j$$

C.8. Until the last test input, return to Step C.2

C.9. Predict output classes from σ_k values, according to the chosen labeling method (see Step A.11)

REFERENCES

[1] S. Grossberg, "The link between brain, learning, attention, and consciousness," *Consciousness and Cognition,* vol. 8**,** pp. 1−44, 1999. Available: http://cns.bu.edu/Profiles/Grossberg/Gro1999ConCog.pdf

[2] S. Grossberg, "How does the cerebral cortex work? Development, learning, attention, and 3D vision by laminar circuits of visual cortex," *Behavioral and Cognitive Neuroscience Reviews*, vol. 2, pp. 47−76, 2003. Available: http://cns.bu.edu/Profiles/Grossberg/Gro2003BCNR.pdf

[3] G. A. Carpenter, "Distributed learning, recognition, and prediction by ART and ARTMAP neural networks," *Neural Networks*, vol. 10, pp. 1473−1494, 1997. Available: http://cns.bu.edu/~gail/115_dART_NN_1997_.pdf

[4] G. A. Carpenter, S. Martens, and O. J. Ogas, "Self-organizing information fusion and hierarchical knowledge discovery: a new framework using ARTMAP neural networks," *Neural Networks*, vol. 18, pp. 287−295, 2005. Available: http://cns.bu.edu/techlab/Docs/CarpenterMartensOgas2005.pdf

[5] O. Parsons and G. A. Carpenter, "ARTMAP neural networks for information fusion and data mining: map production and target recognition methodologies," *Neural Networks*, vol. 16, pp. 1075−1089, 2003. Available: http://cns.bu.edu/~gail/aARTMAP_map_2003_.pdf

[6] P. Lisboa, "Industrial use of safety-related artificial neural networks," *Contract Research Report 327/2001,* Liverpool John Moores University, 2001. Available: http://www.hse.gov.uk/research/crr_pdf/2001/crr01327.pdf

[7] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54−115, 1987. Available: http://cns.bu.edu/techlab/Docs/Carpenter1987.pdf

[8] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Transactions on Neural Networks*, vol. 3, pp. 698−713, 1992. Available: http://cns.bu.edu/~gail/070_Fuzzy_ARTMAP_1992_.pdf

[9] G. A. Carpenter and W. D. Ross, "ART-EMAP: A neural network architecture for object recognition by evidence accumulation," *IEEE Transactions on Neural Networks*, vol. 6, pp. 805−818, 1995. Available: http://cns.bu.edu/~gail/097_ART-EMAP_1995_.pdf

[10] G. A. Carpenter and N. Markuzon, "ARTMAP-IC and medical diagnosis: Instance counting and inconsistent cases," *Neural Networks*, vol. 11, pp. 323−336, 1998. Available: http://cns.bu.edu/~gail/117_ARTMAP-IC_1998_.pdf

[11] J. R. Williamson, "Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps," *Neural Networks*, vol. 9, pp. 881−897, 1998. Available: http://cns.bu.edu/~gail/G-ART_Williamson_1998_.pdf

[12] G. A. Carpenter, B. L. Milenova, and B. W. Noeske, "Distributed ARTMAP: a neural network for fast distributed supervised learning," *Neural Networks,* vol. 11, pp. 793−813, 1998. Available: http://cns.bu.edu/~gail/120_dARTMAP_1998_.pdf

[13] G.A. Carpenter, "Default ARTMAP," *Proceedings of the International Joint Conference on Neural Networks (IJCNN'03)*, Portland, Oregon, pp. 1396−1401, 2003. Available: http://cns.bu.edu/~gail/Default_ARTMAP_2003_.pdf

[14] G.A. Carpenter, S. Grossberg, and J.H. Reynolds, "ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network," *Neural Networks*, vol. 4, pp. 565−588, 1991. Available: http://cns.bu.edu/~gail/054_ARTMAP_1991_.pdf

[15] G. A. Carpenter and M. N. Gjaja, "Fuzzy ART choice functions," *Proceedings of the World Congress on Neural Networks (WCNN-94)*, Hillsdale, NJ: Lawrence Erlbaum Associates, vol. 1, pp. 713−722, 1994.

[16] P. W. Frey and D. J. Slate, "Letter recognition using Holland-style adaptive classifiers," *Machine Learning*, vol. 6, pp. 161−182, 1991.