

Boston University

OpenBU

<http://open.bu.edu>

Cognitive & Neural Systems

CAS/CNS Technical Reports

2003-04

Default ARTMAP

<https://hdl.handle.net/2144/1905>

"Downloaded from OpenBU. Boston University's institutional repository."

Default ARTMAP

Gail Carpenter

April, 2003

Technical Report CAS/CNS-2003-008

Permission to copy without fee all or part of this material is granted provided that: 1. The copies are not made or distributed for direct commercial advantage; 2. the report title, author, document number, and release date appear, and notice is given that copying is by permission of the BOSTON UNIVERSITY CENTER FOR ADAPTIVE SYSTEMS AND DEPARTMENT OF COGNITIVE AND NEURAL SYSTEMS. To copy otherwise, or to republish, requires a fee and / or special permission.

Copyright © 2003

Boston University Center for Adaptive Systems and

Department of Cognitive and Neural Systems

677 Beacon Street

Boston, MA 02215

Default ARTMAP

Gail A. Carpenter

Department of Cognitive and Neural Systems, Boston University
677 Beacon Street, Boston, MA 02215
gail@cns.bu.edu

ABSTRACT - The default ARTMAP algorithm and its parameter values specified here define a ready-to-use general-purpose neural network system for supervised learning and recognition.

I. INTRODUCTION: ART TECHNOLOGY TRANSFER

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 [1,2,3] and as neural systems for technology transfer [4]. A neural computation central to both the scientific and the technological analyses is the *ART matching rule* [5], 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.

Sites of early and ongoing transfer of ART-based technologies include industrial venues such as the Boeing Corporation [6] and government venues such as MIT Lincoln Laboratory [7]. A recent report on industrial uses of neural networks [8] states:

[The] Boeing ... Neural Information Retrieval System is probably still the largest-scale manufacturing application of neural networks. It uses [ART] to cluster binary templates of aeroplane parts in a complex hierarchical network that covers over 100,000 items, grouped into thousands of self-organised clusters. Claimed savings in manufacturing costs are in millions of dollars per annum. (p. 4)

At Lincoln Lab, a team led by Waxman developed an image mining system which incorporates several models of vision and recognition developed in the Boston University

This research was supported by grants from the Air Force Office of Scientific Research (AFOSR F49620-01-1-0397 and AFOSR F49620-01-1-0423) and the Office of Naval Research (ONR N00014-01-1-0624). I also wish to thank and acknowledge the many BU/CNS students who have contributed to this work, including: Suhas Chelian, Marin Gjaja, Norbert Kopco, Natalya Markuzon, Siegfried Martens, Tim McKenna, Boriana Milenova, Ogi Ogas, Olga Parsons, John Reynolds, Brad Rhodes, David Rosen, Bill Ross, Byron Shock, Bill Streilein, Ah-Hwee Tan, and Matt Woods.

Web: http://cns.bu.edu/~gail/Default_ARTMAP_2003_.pdf

Department of Cognitive and Neural Systems (BU/CNS). Over the years a dozen CNS graduates (Aguilar, Baloch, Baxter, Bomberger, Cunningham, Fay, Gove, Ivey, Mehanian, Ross, Rubin, Streilein) have contributed to this effort, which is now located at Alphatech, Inc.

Customers for BU/CNS neural network technologies have attributed their selection of ART over alternative systems to the model's defining design principles. In listing the advantages of its THOT[®] technology, for example, American Heuristics Corporation (AHC) cites several characteristic computational capabilities of this family of neural models, including fast on-line (one-pass) learning, "vigilant" detection of novel patterns, retention of rare patterns, improvement with experience, "weights [which] are understandable in real world terms," and scalability (www.heuristics.com).

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 [9], ART-EMAP [10], ARTMAP-IC [11], Gaussian ARTMAP [12], and distributed ARTMAP [3,13]. Comparative analysis of these systems has led to the identification of a *default ARTMAP* network, which features simplicity of design and robust performance in many application domains [4]. Selection of one particular ARTMAP algorithm, specified here with a complete set of default parameter settings, is intended to facilitate ongoing technology transfer. A user may start with this version of the system, then, if necessary, adjust parameters to suit individual applications.

II. WINNER-TAKE-ALL VS. DISTRIBUTED CODE REPRESENTATIONS

The default ARTMAP algorithm (Section IV) outlines a procedure for labeling an arbitrary number of output classes in a supervised learning problem. A critical aspect of this algorithm is the distributed nature of its internal code representation, which produces continuous-valued test set predictions distributed across output classes.

The character of their code representations, distributed vs. winner-take-all, is, in fact, a primary factor differentiating various ARTMAP networks. The original models [9,14] 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 is the same as fuzzy

ARTMAP during training, but uses a distributed code representation during testing. ARTMAP-IC [11] equals default ARTMAP plus *instance counting*, which biases a category node's test set output by the number of training set inputs coded by that node. Distributed ARTMAP (dARTMAP) employs a distributed code (and instance counting) during both training and testing [3,13]. Versions of these networks [4] form a nested sequence:

fuzzy ARTMAP \subset default ARTMAP \subset
 ARTMAP-IC \subset distributed ARTMAP

That is, distributed ARTMAP reduces to ARTMAP-IC when coding is set to winner-take-all during training; ARTMAP-IC reduces to default ARTMAP when counting weights are set equal to 1; and default ARTMAP reduces to fuzzy ARTMAP when coding is set to winner-take-all during testing as well as training.

ARTMAP variants with winner-take-all (WTA) coding and discrete target class predictions have shown consistent relative deficits in labeling accuracy and post-processing adjustment capabilities.

III. THE DEFAULT ARTMAP SYSTEM

Default ARTMAP codes the current input as a winner-take-all activation pattern during training and as a distributed activation pattern during testing. 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* [13]. The default network also implements the *MT-* search algorithm [11] and sets the *baseline vigilance parameter* 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 [9]. 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.

Even with the number of voters fixed, other design choices appear in systems where output activations may be distributed. In particular, default ARTMAP, which produces a continuous-valued distribution σ_k across output classes k for each test set item, presents options for combining

weighted predictions across voters to make a final class choice. One strategy sums the σ_k values of individual networks to produce a net distributed output pattern, which is then used to determine the predicted class. An alternative strategy first lets each voting network choose its own winning output class, then assigns the test set inputs on the basis of these individual votes. In most applications, the first of these two voting strategies produces better results.

IV. DEFAULT ARTMAP ALGORITHM

Fig. 1 and Table I summarize default ARTMAP notation. Table II 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.

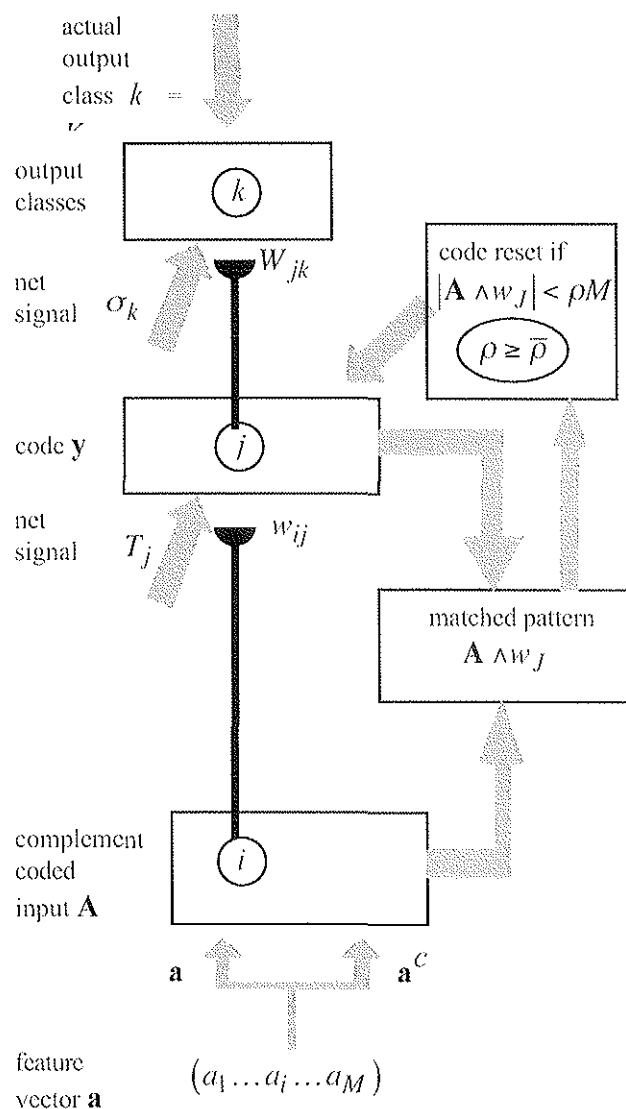


Fig. 1. Default ARTMAP notation.

TABLE I
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) , $0 \leq a_i \leq 1$
\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
Λ, Λ'	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): $(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i)$
$ \cdot $	vector size (L_1 -norm): $ \mathbf{p} \equiv \sum_i p_i $
\mathbf{p}^c	vector complement: $(\mathbf{p}^c)_i \equiv 1 - p_i$

TABLE II
DEFAULT PARAMETER VALUES

NAME	PARAMETER	RANGE	DEFAULT VALUE	NOTES
signal rule parameter	α	$(0, \infty)$	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	$\epsilon < 0$ (MT-) codes inconsistent cases
baseline vigilance	$\bar{\rho}$	$[0, 1]$	0.0	$\bar{\rho} = 0$ maximizes code compression
CAM rule power	ρ	$(0, \infty]$	1.0	Increased Gradient (IG) CAM rule converges to WTA as $\rho \rightarrow \infty$
# 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	

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 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 IV.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 IV.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 [4] 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 Training (Winner-Take-All Code)

The default ARTMAP algorithm specified here is a special case of the distributed ARTMAP (dARTMAP) algorithm described in [13].

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

$$\mathbf{A} \equiv \left(\mathbf{a}, \mathbf{a}^c \right) \text{ and } |\mathbf{A}| = M$$
- 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 $j = C$:

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

$$W_{CK} = 1$$

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

$$\rho = \bar{\rho}$$

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 $j = 1 \dots C$:

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

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 $\left(\frac{|\mathbf{A} \wedge \mathbf{w}_j|}{M} \geq \rho\right)$, set $y_j = 1$ (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:

$$\rho = \frac{|\mathbf{A} \wedge \mathbf{w}_j|}{M} + \varepsilon$$

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:

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

Return to Step B.5 (next input).

C. *Default ARTMAP Testing (Distributed Code)*

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

C.2. Select the next input vector \mathbf{A} , with associated actual output class K

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

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

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

C.5. Let $\Lambda = \{\lambda = 1 \dots C: T_\lambda > \alpha M\}$ and $\Lambda' = \{\lambda = 1 \dots C: T_\lambda = M\} = \{\lambda = 1 \dots C: \mathbf{w}_j = \mathbf{A}\}$

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

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

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

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

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.
[ftp://cns-ftp.bu.edu/pub/diana/Gro.concog98.ps.gz](http://cns-ftp.bu.edu/pub/diana/Gro.concog98.ps.gz)
- [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*, in press, 2003.
<http://www.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.
http://cns.bu.edu/~gail/115_dART_NN_1997_.pdf
- [4] 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, 2003.
http://cns.bu.edu/~gail/ARTMAP_map_2003_.pdf
- [5] 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.
- [6] T.P. Caudell, S.D.G. Smith, R. Escobedo, and M. Anderson, "NIRS: Large scale ART 1 neural architectures for engineering design retrieval," *Neural Networks*, vol. 7, pp. 1339-1350, 1994.
http://cns.bu.edu/~gail/NIRS_Caudell_1994_.pdf
- [7] W. Streilein, A. Waxman, W. Ross, F. Liu, F., M. Braun, D. Fay, P. Harmon, and C.H. Read, "Fused multi-sensor image mining for feature foundation data," In *Proceedings of 3rd International Conference on Information Fusion*, Paris, vol. 1, 2000.
- [8] P. Lisboa, "Industrial use of safety-related artificial neural networks," *Contract Research Report 327/2001*, Liverpool John Moores University, 2001.
http://www.hse.gov.uk/research/crr_.pdf/2001/crr01327.pdf
- [9] 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.
http://cns.bu.edu/~gail/070_Fuzzy_ARTMAP_1992_.pdf
- [10] 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.
http://cns.bu.edu/~gail/097_ART-EMAP_1995_.pdf
- [11] G.A. Carpenter and N. Markuzon, "ARTMAP-IC and medical diagnosis: Instance counting and inconsistent cases," *Neural Networks*, vol. 11, pp. 323-336, 1998.
http://cns.bu.edu/~gail/117_ARTMAP-IC_1998_.pdf
- [12] J.R. Williamson, "Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps," *Neural Networks*, vol. 9, pp. 881-897, 1998.
http://cns.bu.edu/~gail/G-ART_Williamson_1998_.pdf
- [13] 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.
http://cns.bu.edu/~gail/120_dARTMAP_1998_.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.
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.