

# Neural Computing

## Theory and Practice

Philip D. Wasserman

*ANZA Research, Inc.*

Copyright © 1989 by Van Nostrand Reinhold

Library of Congress Catalog Card Number 88-34842

ISBN 0-442-20743-3

All rights reserved. No part of this work covered by the copyright hereon may be reproduced or used in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems—without written permission of the publisher.

Printed in the United States of America

Van Nostrand Reinhold  
115 Fifth Avenue  
New York, New York 10003

Van Nostrand Reinhold International Company Limited  
11 New Fetter Lane  
London EC4P 4EE, England

Van Nostrand Reinhold  
480 La Trobe Street  
Melbourne, Victoria 3000, Australia

Macmillan of Canada  
Division of Canada Publishing Corporation  
164 Commander Boulevard  
Agincourt, Ontario M1S 3C7, Canada

16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data  
Wasserman, Philip D., 1937-

Neural computing : theory and practice / Philip D. Wasserman.  
p. cm.

Includes bibliographies and index.

ISBN 0-442-20743-3

1. Neural computers. I. Title.

QA76.5.W353 1989

006.3—dc19

88-34842  
CIP



VAN NOSTRAND REINHOLD  
New York

## Adaptive Resonance Theory

The human brain performs the formidable task of sorting a continuous flood of sensory information received from the environment. From a deluge of trivia, it must extract vital information, act upon it, and perhaps file it away in long-term memory. Understanding human memorization presents serious problems; new memories are stored in such a fashion that existing ones are not forgotten or modified. This creates a dilemma: how can the brain remain plastic, able to record new memories as they arrive, and yet retain the stability needed to ensure that existing memories are not erased or corrupted in the process?

Conventional artificial neural networks have failed to solve the stability-plasticity dilemma. Too often, learning a new pattern erases or modifies previous training. In some cases, this is unimportant. If there is only a fixed set of training vectors, the network can be cycled through these repeatedly and may eventually learn them all. In a backpropagation network, for example, the training vectors are applied sequentially until the network has learned the entire set. If, however, a fully trained network must learn a new training vector, it may disrupt the weights so badly that complete retraining is required.

In a real-world case, the network will be exposed to a constantly changing environment; it may never see the same training vector twice. Under such circumstances, a backpropagation network will often learn nothing; it will continuously modify its weights to no avail, never arriving at satisfactory settings.

Even worse, Carpenter and Grossberg (1986) have shown examples of a network in which only four training patterns, presented cyclically, will cause network weights to change continuously, never converging. This temporal instability is one of the main factors that led Grossberg and his associates to explore radically different configurations. Adaptive resonance theory, or ART, is one result of research into this problem (Carpenter and Grossberg 1987a; Grossberg 1987).

ART networks and algorithms maintain the plasticity required to learn new patterns, while preventing the modification of patterns that have been learned previously. This capability has stimulated a great deal of interest, but many people have found the theory difficult to understand. The mathematics behind ART are complicated, but the fundamental ideas and implementations are not. We concentrate here on the actual operation of ART; those who are more mathematically inclined will find an abundance of theory in the references. Our objective is to provide enough concrete information in algorithmic form so that the reader can understand the basic ideas and, perhaps, write computer simulations to explore the characteristics of this important network.

### ART ARCHITECTURE

Adaptive resonance theory is divided into two paradigms, each defined by the form of the input data and its processing. ART-1 is designed to accept only binary input vectors, whereas ART-2, a later development that generalizes ART-1, can classify both binary and continuous inputs. Only ART-1 is presented in this volume. The reader interested in ART-2 is referred to Carpenter and Grossberg (1987b) for a complete treatment of this significant development. For brevity, ART-1 is referred to as ART in the paragraphs that follow.

### An Overview of ART

The ART network is a vector classifier. It accepts an input vector and classifies it into one of a number of categories depending upon

which of a number of stored patterns it most resembles. Its classification decision is indicated by the single recognition layer that fires (see Figure 8-1). If the input vector does not match any stored pattern, a new category is created by storing a pattern that is like the input vector. Once a stored pattern is found that matches the input vector within a specified tolerance (the vigilance), that pattern is adjusted (trained) to make it still more like the input vector. No stored pattern is ever modified if it does not match the current input pattern within the vigilance tolerance. In this way, the stability-plasticity dilemma is resolved; new patterns from the environment can create additional classification categories, but a new input pattern cannot cause an existing memory to be changed unless the two match closely.

### A Simplified ART Architecture

Figure 8-1 shows a simplified ART network configuration drawn as five functional modules. It consists of two layers of neurons labeled "comparison" and "recognition." Gain 1, Gain 2, and Reset provide control functions needed for training and classification. Before proceeding to the network's overall function, it is neces-

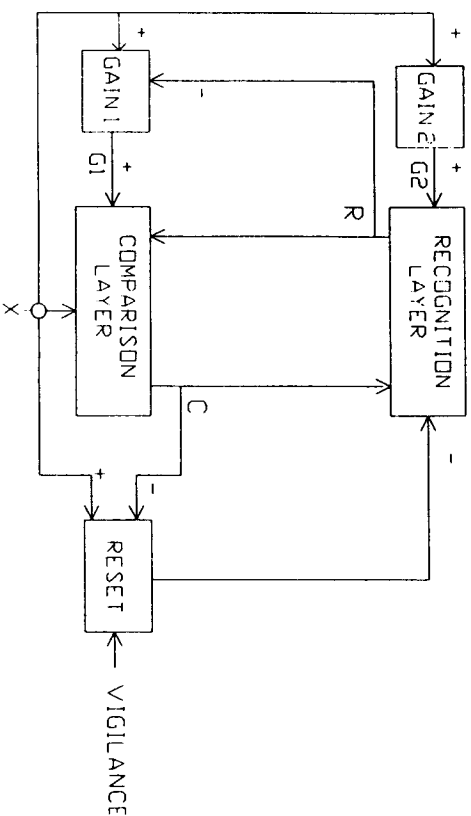


Figure 8-1. Simplified Adaptive Resonance Theory Network

sary to understand the internal operation of the modules; the discussion that follows describes each of them.

### Comparison Layer

The comparison layer receives the binary input vector  $X$  and initially passes it through unchanged to become the vector  $C$ . In a later phase, binary vector  $R$  is produced from the recognition layer, modifying  $C$  as described below.

Each neuron in the comparison layer (see Figure 8-2) receives three binary inputs (zero or one): (1) a component  $x_i$  from the input vector  $X$ ; (2) the feedback signal  $P_i$ , the weighted sum of the recognition layer outputs; and (3) an input from the gain signal Gain 1 (the same signal goes to all neurons in this layer).

To output a one, at least two of a neuron's three inputs must be one; otherwise, its output is zero. This implements the "two-thirds rule," described by Carpenter and Grossberg (1987b). Ini-

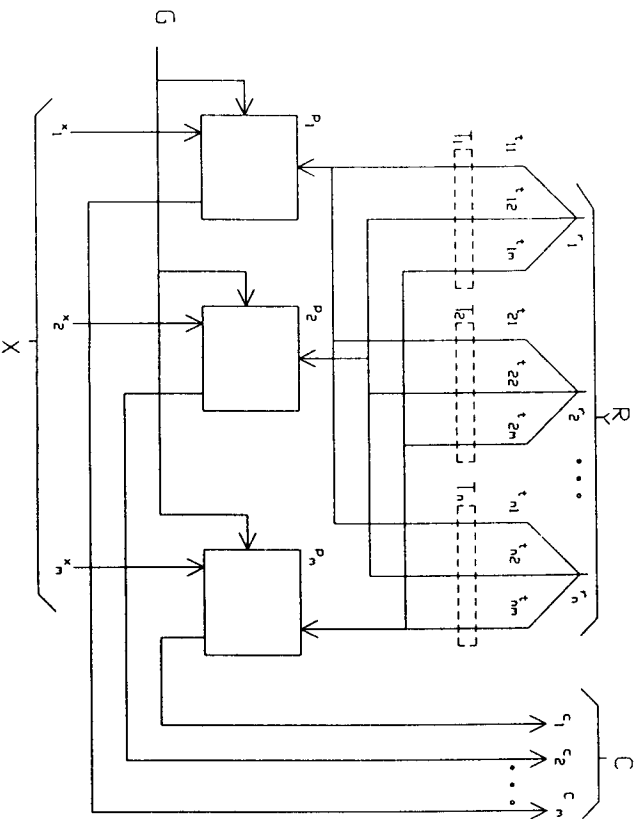


Figure 8-2. Simplified Comparison Layer

tially, gain signal Gain 1 is set to one, providing one of the needed inputs, and all components of the vector  $R$  are set to zero; hence, vector  $C$  starts out identical to the binary input vector  $X$ .

### Recognition Layer

The recognition layer serves to classify the input vector. Each recognition layer neuron has an associated weight vector  $B_j$ . Only the neuron with a weight vector best matching the input vector "fires"; all others are inhibited.

As illustrated in Figure 8-3, a neuron in the recognition layer responds maximally when the vector  $C$  from the comparison layer matches its set of weights; hence, these weights constitute a stored pattern or exemplar, an idealized example, for a category of input vectors. These weights are real numbers, not binary valued. A binary version of the same pattern is also stored in a corresponding set of weights in the comparison layer (see Figure 8-2); this set con-

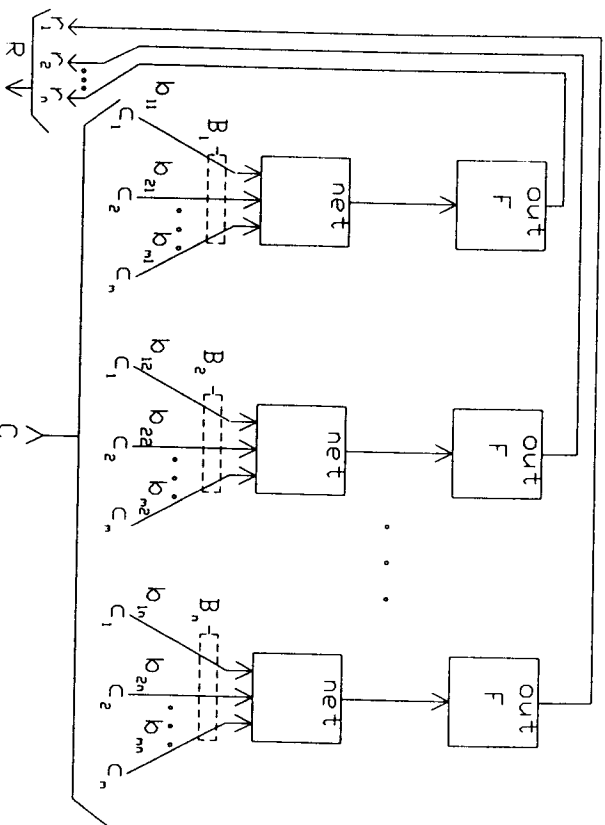


Figure 8-3. Simplified Recognition Layer

sists of those weights that connect to a specific recognition-layer neuron, one weight per comparison-layer neuron.

In operation, each recognition-layer neuron computes a dot product between its weights and the incoming vector  $C$ . The neuron that has weights most like the vector  $C$  will have the largest output, thereby winning the competition while inhibiting all other neurons in the layer.

As shown in Figure 8-4, the neurons in the recognition layer are interconnected by a lateral-inhibition network. In the simplest case (the only one considered in this volume), this ensures that only one neuron "fires" at a time (i.e., only the neuron with the highest activation level will output a one; all others will be zero). This competitive, winner-take-all response is achieved by connecting a negative weight  $I_{ij}$  from each neuron's output  $r_j$  to the input of the other neurons. Thus, if a neuron has a large output it inhibits all other neurons in the layer. Also, each neuron has a positive weight from its output to its own input. If a neuron's output is at a one level, this feedback tends to reinforce and sustain it.

#### Gain 2

G2, the output of Gain 2, is one if input vector  $X$  has any component that is one. More precisely, G2 is the logical "or" of the components of  $X$ .

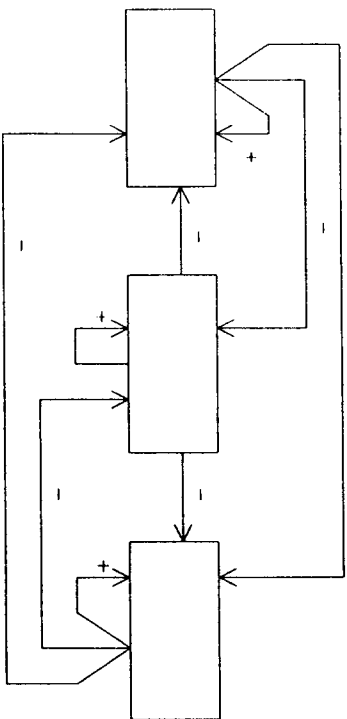


Figure 8-4. Lateral Inhibition-Recognition Layer

#### Gain 1

Like G2, the output of Gain 1 is one if any component of the binary input vector  $X$  is one; however, if any component of  $R$  is one, G1 is forced to zero. The table that follows shows this relationship:

"Or" of X Components	"Or" of R		G2
	0	1	
0	0	0	
1	0	1	
1	1	0	
0	1	0	

#### Reset

The reset module measures the similarity between vectors  $X$  and  $C$ . If they differ by more than the vigilance parameter, a reset signal is sent to disable the firing neuron in the recognition layer.

In operation, the reset module calculates similarity as the ratio of the number of ones in the vector  $C$  to the number of ones in the vector  $X$ . If this ratio is below the vigilance parameter level, the reset signal is issued.

### ART Classification Operation

The ART classification process consists of three major phases: recognition, comparison, and search.

#### The Recognition Phase

Initially, no input vector is applied; hence, all components of input vector  $X$  are zero. This sets G2 to zero, thereby disabling all recognition-layer neurons and causing their outputs to be zero. Because all recognition-layer neurons start out in the same state, all have an equal chance to win the subsequent competition.

The vector to be classified,  $X$ , is now applied. It must have one or more components that are one, thereby making both G1 and G2 equal to one. This "primes" all of the comparison-layer neurons, providing one of the two inputs required by the two-thirds rule, thereby allowing a neuron to fire if the corresponding component of the  $X$  input vector is one. Thus, during this phase, vector  $C$  is an exact duplicate of  $X$ .

Next, for each neuron in the recognition layer a dot product is formed between its associated weight vector  $B_j$  and the vector  $C$  (see Figure 8-4). The neuron with the largest dot product has weights that best match the input vector. It wins the competition and fires, inhibiting all other outputs from this layer. This makes a single component  $r_j$  of vector  $R$  (see Figure 8-1) equal to one, and all other components equal to zero.

To summarize, the ART network stores a set of patterns in the weights associated with the recognition-layer neurons, one for each classification category. The recognition-layer neuron with weights that best match the applied vector fires, its output becomes one, and all other outputs from this layer are forced to zero.

#### The Comparison Phase

The single neuron firing in the recognition layer passes a one back to the comparison layer on its output signal  $r_j$ . This single one may be visualized as fanning out, going through a separate binary weight  $t_{ji}$  to each neuron in the comparison layer, providing each with a signal  $p_i$ , which is equal to the value of  $t_{ji}$  (one or zero) (see Figure 8-5).

The initialization and training algorithms ensure that each weight vector  $T_j$  consists of binary valued weights; also, each weight vector  $B_j$  constitutes a scaled version of the corresponding weight vector  $T_j$ . This means that all components of  $P$ , the comparison-layer excitation vector, are also binary valued.

Since the vector  $R$  is no longer all zeros, Gain 1 is inhibited and its output set to zero. Thus, in accordance with the two-thirds rule, the only comparison-layer neurons that will fire are those that receive simultaneous ones from the input vector  $X$  and the vector  $P$ .

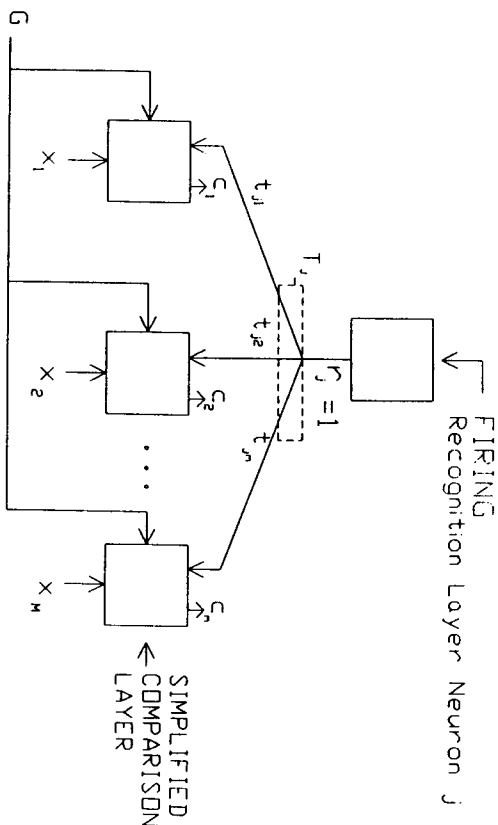


Figure 8-5. Signal Path for a Single-Firing Recognition-Layer Neuron

In other words, the top-down feedback from the recognition layer acts to force components of  $C$  to zero in cases in which the input does not match the stored pattern, that is, when  $X$  and  $P$  do not have coincident ones.

If there is a substantial mismatch between the  $X$  and  $P$  (few coincident ones), few neurons in the comparison layer will fire and  $C$  will contain many zeros, while  $X$  contains ones. This indicates that the pattern  $P$  being fed back is not the one sought and the neuron firing in the recognition layer should be inhibited. This inhibition is performed by the reset block in Figure 8-1, which compares the input vector  $X$  to the  $C$  vector and causes the reset signal to occur if their degree of similarity is less than the vigilance level. The effect of the reset is to force the output of the firing neuron in the recognition layer to zero, disabling it for the duration of the current classification.

#### The Search Phase

If there is no reset signal generated, the match is adequate and the

classification is finished. Otherwise, other stored patterns must be searched to seek a better match. In the latter case, the inhibition of the firing neuron in the recognition layer causes all components of the vector  $\mathbf{R}$  to return to zero.  $G1$  goes to one, and input pattern  $\mathbf{X}$  once again appears at  $C$ . As a result, a different neuron wins in the recognition layer and a different stored pattern  $\mathbf{P}$  is fed back to the comparison layer. If  $\mathbf{P}$  does not match  $\mathbf{X}$ , that firing recognition-layer neuron is also inhibited. This process repeats, neuron by neuron, until one of two events occurs:

1. A stored pattern is found that matches  $\mathbf{X}$  above the level of the vigilance parameter, that is,  $S > \rho$ . If this occurs, the network enters a training cycle that modifies the weights in both  $\mathbf{T}_j$  and  $\mathbf{B}_j$ , the weight vectors associated with the firing recognition layer neuron.
2. All stored patterns have been tried, found to mismatch the input vector, and all recognition-layer neurons are inhibited. If this is the case, a previously unallocated neuron in the recognition layer is assigned to this pattern and its weight vectors  $\mathbf{B}_j$  and  $\mathbf{T}_j$  are set to match the input pattern.

#### Performance Issues

The network described must perform a sequential search through all of its stored patterns. In an analog implementation, this will occur very rapidly; however, it can be a time-consuming process in a simulation on a conventional serial digital computer. If, however, the ART network is implemented with parallel processors, all dot products in the recognition layer can be performed simultaneously. In this case, the search will be very rapid.

The stabilization time required for the lateral-inhibition network can also be lengthy in a serial digital computer. For lateral inhibition to select a "winner," all neurons in the layer are involved in simultaneous computation and communication. This can require a substantial amount of computation before convergence occurs. A feedforward lateral-inhibition network as used in the neocognitron can substantially reduce this time (see Chapter 10).

## ART IMPLEMENTATION

### Overview

ART, as it is generally found in the literature, is something more than a philosophy, but much less concrete than a computer program. This has allowed a wide range of implementations that adhere to the spirit of ART, while they differ greatly in detail. The implementation that follows is based on Lippman (1987), with certain aspects changed for compatibility with Carpenter and Grossberg (1987a) and the conventions of this volume. This treatment is typical, but other successful implementations differ greatly.

### ART Operation

Considered in more detail, the operation of an ART system consists of five phases: initialization, recognition, comparison, search, and training.

#### Initialization

Before starting the network training process, all weight vectors  $\mathbf{B}_j$  and  $\mathbf{T}_j$  as well as the vigilance parameter  $\rho$  must be set to initial values.

The weights of the bottom-up vectors  $\mathbf{B}_j$  are all initialized to the same low value. According to Carpenter and Grossberg (1987a), this should be

$$b_{ij} < L/(L - 1 + m_j) \quad \text{for all } i, j \quad (8-1)$$

where

$$m_j = \text{the number of components in the input vector} \\ L = \text{a constant} > 1 \text{ (typically, } L = 2)$$

This value is critical; if it is too large the network can allocate all recognition-layer neurons to a single input vector.

The weights of the top-down vectors  $\mathbf{T}_j$  are all initialized to 1, so

$$f_j = 1 \quad \text{for all } j, t \quad (8-2)$$

This value is also critical; Carpenter and Grossberg (1987a) prove that top-down weights that are too small will result in no matches at the comparison layer and no training.

The vigilance parameter  $\rho$  is set in the range from 0 to 1, depending upon the degree of mismatch that is to be accepted between the stored pattern and the input vector. At a high value of  $\rho$ , the network makes fine distinctions. On the other hand, a low value causes the grouping of input patterns that may be only slightly similar. It may be desirable to change the vigilance during the training process, making only coarse distinctions at the start, and then gradually increasing the vigilance to produce accurate categorization at the end.

### Recognition

Application of an input vector  $\mathbf{X}$  initiates the recognition phase. Because initially there is no output from the recognition layer,  $G_1$  is set to 1 by the "or" of  $\mathbf{X}$ , providing all comparison-layer neurons with one of the two inputs needed for it to fire (as required by the two-thirds rule). As a result, any component of  $\mathbf{X}$  that is one provides the second input, thereby causing its associated comparison-layer neuron to fire and output a one. Thus, at this time, the vector  $\mathbf{C}$  will be identical to  $\mathbf{X}$ .

As discussed previously, recognition is performed as a dot product for each neuron in the recognition layer, and is expressed as follows:

$$NET_j = (\mathbf{B}_j \cdot \mathbf{C}) \quad (8-3)$$

where

$\mathbf{B}_j$  = the weight vector associated with recognition-layer neuron  $j$

$\mathbf{C}$  = the output vector of the comparison-layer neuron; at this time,  $\mathbf{C}$  is equal to  $\mathbf{X}$

$NET_j$  = the excitation of neuron  $j$  in the recognition layer

$F$  is the threshold function that follows:

$$OUT_j = 1 \text{ if } NET_j > T \\ 0 \text{ otherwise} \quad (8-4)$$

where  $T$  is a threshold.

Lateral inhibition is assumed to exist but is ignored here to simplify these equations. It ensures that only the recognition-layer neuron with the highest value for  $NET$  will have an output of one; all others will output zero. It is quite possible to devise systems in which more than one recognition-layer neuron fires at a time, but this is beyond the scope of this volume.

### Comparison

At this point, the feedback signal from the recognition layer causes  $G_1$  to go to zero; the two-thirds rule permits only those comparison-layer neurons to fire that have corresponding components of the vectors  $\mathbf{P}$  and  $\mathbf{X}$  both equal to one.

The reset block compares the vector  $\mathbf{C}$  to the input vector  $\mathbf{X}$ , producing a reset output whenever their similarity  $S$  is below the vigilance threshold. Computing this similarity is simplified by the fact that both vectors are binary (all elements are either one or zero). The procedure that follows computes the required measure of similarity.

1. Call  $D$  the number of 1s in the  $\mathbf{X}$  vector.
2. Call  $N$  the number of 1s in the  $\mathbf{C}$  vector.

Then compute the similarity  $S$  as follows:

$$S = N/D \quad (8-5)$$

For example, suppose that

$$\mathbf{X} = 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \quad \text{then } D = 5$$

$$\mathbf{C} = 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \quad \text{then } N = 4$$

$$S = N/D = 0.8$$

$S$  will vary from 1 (perfect match) to 0 (worst mismatch).

Note that the two-thirds rule makes  $\mathbf{C}$  the logical "and" of the

input vector  $\mathbf{X}$  with the vector  $\mathbf{P}$ . But  $\mathbf{P}$  is equal to  $\mathbf{T}_j$ , the weight vector from the winning neuron. Thus,  $D$  may be found as the number of 1s in the logical "and" of  $\mathbf{T}_j$  with  $\mathbf{X}$ .

#### Search

If the similarity  $S$  of the winning neuron is greater than the vigilance, no search is required. If, however, the network has been previously trained, application of an input vector that is not identical to any seen before may fire a recognition-layer neuron with a match below the vigilance level. Due to the training algorithm, it is possible that a different recognition-layer neuron will provide a better match, exceeding the vigilance level, even though the dot product between its weight vector and the input vector may be lower. An example of this situation is shown below.

If the similarity is below the vigilance level, the stored patterns must be searched, seeking one that matches the input vector more closely, or failing that, terminating on an uncommitted neuron that will then be trained. To initiate the search, the reset signal temporarily disables the firing neuron in the recognition layer for the duration of the search, G1 goes to one, and a different recognition-layer neuron wins the competition. Its pattern is then tested for similarity and the process repeats until either a recognition-layer neuron wins the competition with similarity greater than the vigilance (a successful search), or all committed recognition-layer neurons have been tried and disabled (unsuccessful search).

An unsuccessful search will automatically terminate on an uncommitted neuron, as its top-down weights are all ones, their initial values. Thus, the two-thirds rule will make the vector  $\mathbf{C}$  identical to  $\mathbf{X}$ , the similarity  $S$  will be one, and the vigilance will be satisfied.

#### Training

Training is the process in which a set of input vectors are presented sequentially to the input of the network, and the network weights are so adjusted that similar vectors activate the same recognition-layer neuron. Note that this is unsupervised training; there is no teacher and no target vector to indicate the desired response.

Carpenter and Grossberg (1987a) distinguish two kinds of train-

ing: slow and fast. In slow training, an input vector may be applied so briefly that network weights do not have enough time to reach their asymptotic values during a single presentation. Thus, the weights will be determined by the statistics of the input vectors rather than by the characteristics of any one. The differential equations of slow training describe the network dynamics during the training process.

Fast training is a special case of slow training that applies if the input vectors are applied for a long enough period of time to allow the weights to approach their final values. In this case, the training formulas involve only algebraic equations. Also, top-down weights assume only binary values rather than the continuous range required in fast training. Only fast training is described in this volume; the interested reader can find an excellent treatment of the more general, slow-training case in Carpenter and Grossberg (1987a).

The training algorithm that follows is applied in both successful and unsuccessful searches.

Set the vector of bottom-up weights  $\mathbf{B}_j$  (associated with the firing recognition-layer neuron  $j$ ) to the normalized values of the vector  $\mathbf{C}$ . Carpenter and Grossberg (1987a) calculate these weights as follows:

$$b_{ij} = (L c_i) / \left( L - 1 + \sum_k c_k \right) \quad (8-6)$$

where

- $c_i$  = the  $i$ th component of the comparison-layer output vector
- $j$  = the number of the winning recognition-layer neuron
- $b_{ij}$  = the bottom-up weight in  $\mathbf{B}_j$ , connecting neuron  $i$  in the comparison layer to neuron  $j$  in the recognition layer
- $L$  = a constant  $> 1$  (typically 2)

Weights in the vector  $\mathbf{T}_j$  that are associated with the new stored pattern are adjusted so that they equal the corresponding binary values in the vector  $\mathbf{C}$ :

$$t_{ji} = c_i \quad \text{for all } i \quad (8-7)$$

where  $t_{ji}$  is the weight from the winning neuron  $j$  in the recognition layer neuron  $i$  in the comparison layer.

### AN ART TRAINING EXAMPLE

In outline, the network is trained by adjusting the top-down and bottom-up weights so that the application of an input pattern causes the network to activate the recognition-layer neuron associated with a similar stored pattern. Furthermore, training is accomplished in a fashion that does not destroy patterns that were learned previously, thereby preventing temporal instability. This task is controlled by the level of the vigilance parameter. A novel input pattern (one that the network has not seen before) will fail to match stored patterns within the tolerance imposed by the vigilance level, thereby causing a new stored pattern to be formed. An input pattern sufficiently like a stored pattern will not form a new exemplar; it will simply modify one that it resembles. Thus, with a suitable setting of the vigilance level, new input patterns already learned and temporal instability are avoided.

Figure 8-6 shows a typical ART training session. Letters are shown as patterns of small squares on an 8-by-8 grid. Each square

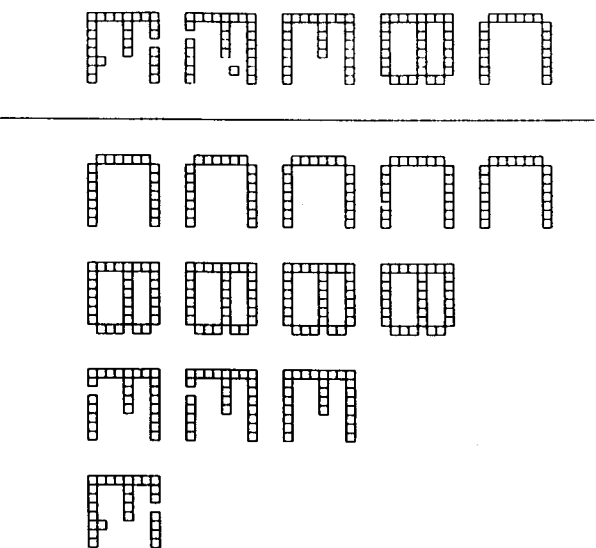


Figure 8-6. ART Training Session

on the left represents a component of the  $X$  vector with a value of one; all squares not shown are components with values of zero. Letters on the right represent the stored patterns; each is the set of the values of the components of a vector  $T_j$ .

First, the letter C is input to the newly initialized system. Because there is no stored pattern that matches it within the vigilance limit, the search phase fails; a new neuron is assigned in the recognition layer, and the weights  $T_j$  are set to equal the corresponding components of the input vector, with weights  $B_j$  becoming a scaled version.

Next, the letter B is presented. This also fails in the search phase and another new neuron is assigned. This is repeated for the letter E. Then, a slightly corrupted version of the letter E is presented to the network. It is close enough to the stored E to pass the vigilance test, so it is used to train the network. The missing pixel in the lower leg of the E produces a zero in the corresponding position of the vector C, causing the training algorithm to set that weight of the stored pattern to zero, thereby reproducing the break in the stored pattern. The extra isolated square does not corrupt the stored pattern, as there is no corresponding one introduced into it.

The fourth character is an E, with two different errors. This fails to match a stored pattern ( $S$  is less than the  $\rho$ ), so the search fails and a new neuron is assigned.

This example illustrates the importance of setting the vigilance parameter correctly. If the vigilance is too high, most patterns will fail to match those in storage and the network will create a new neuron for each of them. This results in poor generalization, as minor variations of the same pattern become separate categories. These categories proliferate, all available recognition-layer neurons are assigned, and the system's ability to incorporate new data halts. Conversely, if the vigilance is too low, totally different letters will be grouped together, distorting the stored pattern until it bears little resemblance to any of them.

Unfortunately, there is no theory to guide the setting of the vigilance parameter; one must first decide what degree of difference between patterns will constitute a different category. The boundaries between categories are often "fuzzy" and *a priori* decisions on a large set of input examples may be prohibitively difficult.

Carpenter and Grossberg (1987a) propose a feedback process to adjust the vigilance, whereby incorrect categorization results in "punishment" from an outside agency that acts to raise the vigilance. Such a system requires a standard to determine if the classification was incorrect.

### CHARACTERISTICS OF ART

The ART system has a number of important characteristics that are not obvious. The formulas and algorithms may seem arbitrary, whereas in fact, they have been carefully chosen to satisfy theorem regarding system performance. This section discusses some of the implications of the ART algorithms, thereby showing the reasoning behind the design of the initialization and training formulas.

#### Top-Down Weight Initialization

From the earlier training example it may be seen that the two-thirds rule makes vector C the "and" between the input vector X, and the winning stored vector T<sub>j</sub>. That is, only if corresponding components of each are one will that component of C be one. After training, these components of T<sub>j</sub> remain one; all others are forced to zero.

This explains why the top-down weights must be initialized to ones. If they were initialized to zeros, all components of vector C would be zero regardless of the input vector components, and the training algorithm would prevent the weights from being anything but zero.

Training may be viewed as a process of "pruning" components of the stored vectors that do not match the input vectors. This process is irreversible; that is, once a top-down weight has been set to zero, the training algorithm can never restore it to a one.

This characteristic has important implications for the learning process. Suppose that a group of closely related vectors should be classified into the same category, indicated by their firing the same recognition-layer neuron. If they are presented sequentially to the

network, the first will be assigned a recognition-layer neuron; its weights will be trained to match the input vector. Training with the rest of the vectors will set the weights of the stored vector to zero in all positions where they coincide with zeros from any of these input vectors. Thus, the stored vector comes to represent the logical intersection of all of the training vectors and may be thought of as encoding the essential features of a category of input vectors. A new vector consisting only of these essential features will be assigned to this category; thus, the network correctly recognizes a pattern it has never seen before, an ability reminiscent of human abstraction.

#### Bottom-Up Weight Adjustments

The weight adjustment formula (Equation 8-6, repeated here for reference) is central to the operation of the ART system.

$$b_{ij} = (L c_j) / (L - 1 + \sum_k c_k) \quad (8-6)$$

The summation in the denominator represents the number of ones in the output of the comparison layer. As such, this number may be thought of as the "size" of this vector. With this interpretation, large C vectors produce smaller weight values for  $b_{ij}$  than do small C vectors. This "self-scaling" property makes it possible to separate two vectors when one is a subset of another; that is, its ones are in some but not all of the positions of the other.

To demonstrate the problem that results if the scaling shown in Equation 8-6 is not used, suppose that the network has been trained on the two input vectors that follow, with a recognition-layer neuron assigned to each.

$$X_1 = 1 \ 0 \ 0 \ 0 \ 0$$

$$X_2 = 1 \ 1 \ 1 \ 0 \ 0$$

Note that X<sub>1</sub> is a subset of X<sub>2</sub>. Without the scaling property, bottom-up weights would be trained to the same values for each pattern. If this value were chosen to be 1.0, the weight patterns that follow would result.

$$\mathbf{T}_1 = \mathbf{B}_1 = 1 \ 0 \ 0 \ 0 \ 0$$

$$\mathbf{T}_2 = \mathbf{B}_2 = 1 \ 1 \ 1 \ 0 \ 0$$

If  $\mathbf{X}_1$  is applied once more, both recognition-layer neurons receive the same activation; hence, likely as not, neuron 2, the wrong one, will win the competition.

In addition to making an incorrect classification, training can be destroyed. Because  $\mathbf{T}_2$  feeds down 1 1 1 0 0, only the first 1 is matched by the input vector,  $\mathbf{C}$  becomes 1 0 0 0 0, vigilance is satisfied, and training sets the second and third 1s of  $\mathbf{T}_2$  and  $\mathbf{B}_2$  to 0, destroying the trained pattern.

Scaling the bottom-up weights according to Equation 8-6 prevents this undesirable behavior. Suppose, for example, that Equation 8-6 is used with  $L = 2$ , thereby producing the formula that follows:

$$b_{ij} = (2 c_j) / \left( 1 + \sum_k c_k \right)$$

Bottom-up weights will now train to the values

$$\mathbf{B}_1 = 1 \ 0 \ 0 \ 0 \ 0$$

$$\mathbf{B}_2 = 1/2 \ 1/2 \ 1/2 \ 0 \ 0$$

Applying  $\mathbf{X}_1$  produces an excitation of 1.0 on recognition-layer neuron 1, but only 1/2 for neuron 2; thus, neuron 1 (correctly) wins the competition. Similarly, applying  $\mathbf{X}_2$  produces excitation levels of 1.0 for neuron 1, but 3/2 for neuron 2, again selecting the correct winner.

### Bottom-Up Weight Initialization

Initializing the bottom-up weights to low values is essential to the correct functioning of the ART system. If they are too high, input vectors that have already been learned will activate an uncommitted recognition-layer neuron rather than the one that has been previously trained. The formula for bottom-up weight assignments, Equation 8-1, is repeated here for reference:

$$b_{ij} < L/(L - 1 + m) \quad \text{for all } i, j \quad (8-1)$$

Setting these weights to low values ensures that an uncommitted neuron will not "overpower" a trained recognition-layer neuron. Using our previous example with  $L = 2$  and  $m = 5$ ,  $b_{ij} < 1/3$ , so we arbitrarily set  $b_{ij} = 1/6$ . With these weights, applying a vector for which the network has been trained will cause the correctly trained recognition-layer neuron to win over an uncommitted neuron. For example, on an uncommitted neuron,  $\mathbf{X}_1$  would produce an excitation of 1/6, while  $\mathbf{X}_2$  would produce 1/2; both are below the excitation produced on the neuron for which they were trained.

### Searching

It may appear that direct access obviates the need for a search except when an uncommitted recognition-layer neuron is to be assigned. This is not the case; application of an input vector that is similar, but not identical, to one of the stored patterns may not on the first trial select a recognition-layer neuron such that the similarity  $S$  exceeds the vigilance  $\rho$ , even though another neuron will.

As in the preceding example, assume that the network has been trained on the two vectors that follow:

$$\mathbf{X}_1 = 1 \ 0 \ 0 \ 0 \ 0$$

$$\mathbf{X}_2 = 1 \ 1 \ 1 \ 0 \ 0$$

with bottom-up weight vectors trained as follows:

$$\mathbf{B}_1 = 1 \ 0 \ 0 \ 0 \ 0$$

$$\mathbf{B}_2 = 1/2 \ 1/2 \ 1/2 \ 0 \ 0$$

Now apply an input vector  $\mathbf{X}_3 = 1 \ 1 \ 0 \ 0 \ 0$ . In this case, the excitation to recognition-layer neuron 1 will be 1.0, while that of neuron 2 will be only 2/3. Neuron 1 will win (even though it is not the best match),  $\mathbf{C}$  will be set to 1 0 0 0 0, and the similarity  $S$  will be 1/2. If the vigilance is set at 3/4, neuron 1 will be disabled, and neuron 2 will now win the competition.  $\mathbf{C}$  will now become 1 1 0 0 0,  $S$  will be 1, the vigilance will be satisfied, and the search will stop.

## Theorems of ART

In Carpenter and Grossberg (1987a), several theorems are proven that show powerful characteristics to be inherent to the system. The four results that follow are among the most important:

1. After training has stabilized, application of one of the training vectors (or one with the essential features of the category) will activate the correct recognition-layer neuron without searching. This "direct-access" characteristic implies rapid access to previously learned patterns.
2. The search process is stable. After the winning recognition-layer neuron is chosen, the system will not switch from one neuron to another as a result of the top-down vector's modification of C, the output of the comparison layer; only reset can cause this change.
3. Similarly, training is stable. Training will not cause a switch from one recognition-layer neuron to another.
4. The training process terminates. Any sequence of arbitrary input vectors will produce a stable set of weights after a finite number of learning trials; no repetitive sequence of training vectors will cause ART's weights to cycle endlessly.

## DISCUSSION

ART is an interesting and important paradigm. It solves the stability-plasticity dilemma and performs well in other regards. The ART architecture was designed to be biologically plausible; that is, its mechanisms are intended to be consistent with those of the brain (as we understand them). It may fail, however, to simulate the distributed storage of internal representations, which many see as an important characteristic of the cerebral function. ART's exemplars represent "grandmother cells"; loss of one cell destroys an entire memory. In contrast, memories in the brain seem to be distributed over substantial regions; a recollection can often survive considerable physical damage without being lost entirely.

It seems logical to study architectures that do not violate our understanding of the brain's organization and function. The hu-

man brain constitutes an existence proof that a solution to the pattern-recognition problem is possible. It seems sensible to emulate this working system if we wish to duplicate its performance. However, a counterargument recounts the history of powered flight; man failed to get off the ground until he stopped trying to imitate the moving wings and feathers of the birds.

## References

- Carpenter, G., and Grossberg, S. 1986. Neural dynamics of category learning and recognition: Attention, memory consolidation, and amnesia. In *Brain Structure, Learning and Memory* (AAAS Symposium Series), eds. J. Davis, R. Newburgh, and E. Wegman.
- \_\_\_\_\_. 1987a. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing* 37:54-115.
- \_\_\_\_\_. 1987b. ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics* 26(23):4919-30.
- Grossberg, S. 1987. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science* 11:23-63.
- Lippman, R. P. 1987. An introduction to computing with neural nets. *IEEE Transactions on Acoustics, Speech and Signal Processing*, April, pp. 4-22.