ARTMAP: Supervised Real-Time Learning and Classification of Nonstationary Data by a Self-Organizing Neural Network

GAIL A. CARPENTER, STEPHEN GROSSBERG, AND JOHN H. REYNOLDS

Center for Adaptive Systems and Department of Cognitive and Neural Systems

(Received 28 November 1990; revised and accepted 13 February 1991)

Abstract—This article introduces a new neural network architecture, called ARTMAP, that autonomously learns to classify arbitrarily many, arbitrarily ordered vectors into recognition categories based on predictive success. This supervised learning system is built up from a pair of Adaptive Resonance Theory modules (ART, and ART,) that are capable of self-organizing stable recognition categories in response to arbitrary sequences of input patterns. During training trials, the ART, module receives a stream \( \{a^n\} \) of input patterns, and ART, receives a stream \( \{b^n\} \) of input patterns, where \( b^n \) is the correct prediction given \( a^n \). These ART modules are linked by an associative learning network and an internal controller that ensures autonomous system operation in real time. During test trials, the remaining patterns \( a^n' \) are presented without \( b^n \), and their predictions at ART, are compared with \( b^n \). Tested on a benchmark machine learning database in both on-line and off-line simulations, the ARTMAP system learns orders of magnitude more quickly, efficiently, and accurately than alternative algorithms, and achieves 100% accuracy after training on less than half the input patterns in the database. It achieves these properties by using an internal controller that jointly maximizes predictive generalization and minimizes predictive error by linking predictive success to category size on a trial-by-trial basis, using only local operations. This computation increases the vigilance parameter \( \rho \) of ART, by the minimal amount needed to correct a predictive error at ART, and \( \rho \) calibrates the minimum confidence that ART, must have in a category, or hypothesis, activated by an input \( a^n \) in order for ART, to accept that category, rather than search for a better one through an automatically controlled process of hypothesis testing. Parameter \( \rho \) is compared with the degree of match between \( a^n \) and the top-down learned expectation, or prototype, that is read-out subsequent to activation of an ART, category. Search occurs if the degree of match is less than \( \rho \). ARTMAP is hereby a type of self-organizing expert system that calibrates the selectivity of its hypotheses based upon predictive success. As a result, rare but important events can be quickly and sharply distinguished even if they are similar to frequent events with different consequences. Between input trials \( \rho \) relaxes to a baseline vigilance \( \bar{\rho} \). When \( \bar{\rho} \) is large, the system runs in a conservative mode, wherein predictions are made only if the system is confident of the outcome. Very few false-alarm errors then occur at any stage of learning, yet the system reaches asymptote with no loss of speed. Because ARTMAP learning is self-stabilizing, it can continue learning one or more databases, without degrading its corpus of memories, until its full memory capacity is utilized.

Keywords—ARTMAP, Adaptive resonance theory, Supervised learning, Self-organization, Prediction, Expert system, Mushroom database, Machine learning.

1. INTRODUCTION: PREDICTIVE ART

As we move freely through the world, we can attend to both familiar and novel objects, and can rapidly learn to recognize, test hypotheses about, and learn to name novel objects without unsel ectively disrupting our memories of familiar objects. This article describes a new self-organizing neural network architecture—called a Predictive ART or ARTMAP architecture—that is capable of fast, yet stable, on-line recognition learning, hypothesis testing, and adaptive naming in response to an arbitrary stream of input patterns.
The possibility of stable learning in response to an arbitrary stream of inputs is required by an autonomous learning agent that needs to cope with unexpected events in an uncontrolled environment. One cannot restrict the agent’s ability to process input sequences if one cannot predict the environment in which the agent must successfully function. The ability of humans to vividly remember exciting adventure movies is a familiar example of fast learning in an unfamiliar environment.

1.1. Fast Learning About Rare Events

A successful autonomous agent must be able to learn about rare events that have important consequences, even if these rare events are similar to frequent events with very different consequences. Survival may hereby depend on fast learning in a nonstationary environment. Many learning schemes are, in contrast, slow learning models that average over individual event occurrences and are degraded by learning instabilities in a nonstationary environment (Carpenter & Grossberg, 1988; Grossberg, 1988a).

1.2. Many-to-One and One-to-Many Learning

An efficient recognition system needs to be capable of many-to-one learning. For example, each of the different exemplars of the font for a prescribed letter may generate a single compressed representation that serves as a visual recognition category. This exemplar-to-category transformation is a case of many-to-one learning. In addition, many different fonts, including lower case and upper case printed fonts and scripts of various kinds, can all lead to the same verbal name for the letter. This is a second sense in which learning may be many-to-one.

Learning may also be one-to-many, so that a single object can generate many different predictions or names. For example, upon looking at a banana, one may classify it as an oblong object, a fruit, a banana, a yellow banana, and so on. A flexible knowledge system may thus need to represent in its memory many predictions for each object, and to make the best prediction for each different context in which the object is embedded.

1.3. Control of Hypothesis Testing, Attention, and Learning by Predictive Success

Why does not an autonomous recognition system get trapped into learning only that interpretation of an object which is most salient given the system’s initial biases? One factor is the ability of that system to reorganize its recognition, hypothesis testing, and naming operations based upon its predictive success or failure. For example, a person may learn a visual recognition category based upon seeing bananas of various colors and associate that category with a certain taste. Due to the variability of color features compared with those of visual form, this learned recognition category may incorporate form features more strongly than color features. However, the color green may suddenly and unexpectedly become an important differential predictor of a banana’s taste.

The different taste of a green banana triggers hypothesis testing that shifts the focus of visual attention to give greater weight, or salience, to the banana’s color features without negating the importance of the other features that define a banana’s form. A new visual recognition category can hereby form for green bananas, and this category can be used to accurately predict the different taste of green bananas. The new, finer category can form, moreover, without recoding either the previously learned generic representation of bananas or their taste association.

Future representations may also form that incorporate new knowledge about bananas, without disrupting the representations that are used to predict their different tastes. In this way, predictive feedback provides one means whereby one-to-many recognition and prediction codes can form through time, by using hypothesis testing and attention shifts that support new recognition learning without forcing unselective forgetting of previous knowledge.

1.4. Adaptive Resonance Theory

The architecture described herein forms part of Adaptive Resonance Theory, or ART, which was introduced in 1976 (Grossberg, 1976a, 1976b) in order to analyze how brain networks can autonomously learn in real time about a changing world in a rapid but stable fashion. Since that time, ART has steadily developed as a physical theory to explain and predict ever larger data bases about cognitive information processing and its neural substrates (Grossberg, 1982a, 1987a, 1987b, 1988b). A parallel development has described a series of rigorously characterized neural architectures—called ART 1, ART 2, and ART 3—with increasingly powerful learning, pattern recognition, and hypothesis testing capabilities (Carpenter & Grossberg, 1987a, 1987b, 1988, 1990).

1.5. Self-Organizing Predictive Maps

The present class of architectures are called Predictive ART architectures because they incorporate ART modules into systems that can learn to predict a prescribed m-dimensional output vector \( b \) given a prescribed n-dimensional input vector \( a \) (Figure 1). The present example of Predictive ART is called ARTMAP because its transformation from vectors...
in $\mathbb{R}^n$ to vectors in $\mathbb{R}^m$ defines a map that is learned by example from the correlated pairs $\{a^{(p)}, b^{(p)}\}$ of sequentially presented vectors, $p = 1, 2, \ldots$ (Carpenter, 1989). For example, the vectors $a^{(p)}$ may encode visual representations of objects, and the vectors $b^{(p)}$ may encode their predictive consequences, such as different tastes in the banana example above. The degree of code compression in memory is an index of the system's ability to generalize from examples.

Figure 1 compares properties of the ARTMAP network with those of the Back Propagation network (Parker, 1982; Rumelhart & McClelland, 1986; Werbos, 1974, 1982). Both ARTMAP and Back Propagation are supervised learning systems. With supervised learning, an input vector $a^{(p)}$ is associated
with another input vector $\mathbf{b}^{(p)}$ on each training trial. On a test trial, a new input $\mathbf{a}$ is presented that has never been experienced before. This input predicts an output vector $\mathbf{b}$. System performance is evaluated by comparing $\mathbf{b}$ with the correct answer. This property of generalization is the system's ability to correctly predict correct answers to a test set of novel inputs $\mathbf{a}$.

1.6. Conjointly Maximizing Generalization and Minimizing Predictive Error

The ARTMAP system is designed to conjointly maximize generalization and minimize predictive error under fast learning conditions in real time in response to an arbitrary ordering of input patterns. Remarkably, the network can achieve 100% test set accuracy on the machine learning benchmark database described below. Each ARTMAP system learns to make accurate predictions quickly, in the sense of using relatively little computer time; efficiently, in the sense of using relatively few training trials; and flexibly, in the sense that its stable learning permits continuous new learning, on one or more databases, without eroding prior knowledge, until the full memory capacity of the network is exhausted. In an ARTMAP network, the memory capacity can be chosen arbitrarily large without sacrificing the stability of fast learning or accurate generalization.

1.7. Match Tracking of Predictive Confidence by Attentive Vigilance

An essential feature of the ARTMAP design is its ability to conjointly maximize generalization and minimize predictive error on a trial-by-trial basis using only local operations. It is this property which enables the system to learn rapidly about rare events that have important consequences even if they are very similar to frequent events with different consequences. This property builds upon a key design feature of all ART systems; namely, the existence of an orienting subsystem that responds to the unexpectedness, or novelty, of an input exemplar $\mathbf{a}$ by driving a hypothesis testing cycle, or parallel memory search, for a better, or totally new, recognition category for $\mathbf{a}$. Hypothesis testing is triggered by the orienting subsystem if $\mathbf{a}$ activates a recognition category that reads out a learned expectation, or prototype, which does not match $\mathbf{a}$ well enough. The degree of match provides an analog measure of the predictive confidence that the chosen recognition category represents $\mathbf{a}$, or of the novelty of $\mathbf{a}$ with respect to the hypothesis that is symbolically represented by the recognition category. This analog match value is computed at the orienting subsystem where it is compared with a dimensionless parameter that is called vigilance (Carpenter & Grossberg, 1987a, 1987b). A cycle of hypothesis testing is triggered if the degree of match is less than vigilance. Conjoint maximization of generalization and minimization of predictive error is achieved on a trial-by-trial basis by increasing the vigilance parameter in response to a predictive error on a training trial (Carpenter & Grossberg, 1987a). The minimum change is made that is consistent with correction of the error. In fact, the predictive error causes the vigilance to increase rapidly until it just exceeds the analog match value, in a process called match tracking.

Before each new input arrives, vigilance relaxes to a baseline vigilance value. Setting baseline vigilance to 0 maximizes code compression. The system accomplishes this by allowing an "educated guess" on every trial, even if the match between input and learned code is poor. Search ensues, and a new category is established, only if the prediction made in this forced-choice situation proves wrong. When predictive error carries a cost, however, baseline vigilance can be set at some higher value, thereby decreasing the "false alarm" rate. With positive baseline vigilance, the system responds "I don't know" to an input that fails to meet the minimum matching criterion. Predictive error rate can hereby be made very small, but with a reduction in code compression. Search ends when the internal control system (Figure 1) determines that a global consensus has been reached.

1.8. Self-Organizing Expert System

ARTMAP achieves its combination of desirable properties by acting as a type of self-organizing expert system. It incorporates the basic properties of all ART systems (Carpenter & Grossberg, 1988) to carry out autonomous hypothesis testing and parallel memory search for appropriate recognition codes. Hypothesis testing terminates in a sustained state of resonance that persists as long as an input remains approximately constant. The resonance generates a focus of attention that selects the bundle of critical features common to the bottom-up input and the top-down expectation, or prototype, that is read-out by the resonating recognition category. Learning of the critical feature pattern occurs in this resonant and attentive state, hence the term adaptive resonance.

1.9. 2/3 Rule Matching, Priming, Intentionality, and Logic

The resonant focus of attention is a consequence of a matching rule called the 2/3 Rule (Carpenter & Grossberg, 1987a). This rule clarifies how a bottom-up input pattern can supraliminally activate its feature detectors at the level $F_i$ of an ART network.
yet a top-down expectation can only subliminally sensitize, or prime, the level $F_1$. Supraliminal activation means that $F_1$ can automatically generate output signals that initiate further processing of the input. Subliminal activation means that $F_1$ cannot generate output signals, but its primed cells can more easily be activated by bottom-up inputs. For example, the verbal command “Look for the yellow banana” can prime visual feature detectors to respond more sensitively to visual inputs that represent a yellow banana, without forcing these cells to be fully activated, which would have caused a visual hallucination.

Carpenter and Grossberg (Grossberg, 1987a) have shown that the 2/3 Rule is realized by a kind of analog spatial logic. This logical operation computes the spatial intersection of bottom-up and top-down information. The spatial intersection is the focus of attention. It is of interest that subliminal top-down priming, which instantiates a type of “intentionality” in an ART system, implies a type of matching law, which instantiates a type of “logic.” Searle (1983) and others have criticized some Artificial Intelligence models because they sacrifice intentionality for logic. In ART, intentionality implies logic.

2. THE ARTMAP SYSTEM

The main elements of an ARTMAP system are shown in Figure 2. Two modules, ART$_a$ and ART$_b$, read vector inputs $a$ and $b$. If ART$_a$ and ART$_b$ were disconnected, each module would self-organize category groupings for the separate input sets. In the application described below, ART$_a$ and ART$_b$ are fast-learn ART 1 modules coding binary input vec-

FIGURE 2. Block diagram of an ARTMAP system. Modules ART$_a$ and ART$_b$ self-organize categories for vector sets $a$ and $b$. ART$_a$ and ART$_b$ are connected by an inter-ART module that consists of the Map Field and the control nodes called Map Field gain control and Map Field orienting subsystem. Inhibitory paths are denoted by a minus sign; other paths are excitatory.
tors. ART\(_a\) and ART\(_b\) are here connected by an inter-
ART module that in many ways resembles ART 1. This inter-ART module includes a Map Field that controls the learning of an associative map from ART\(_a\) recognition categories to ART\(_b\) recognition categories. This map does not directly associate exemplars \(a\) and \(b\), but rather associates the compressed and symbolic representations of families of exemplars \(a\) and \(b\). The Map Field also controls match tracking of the ART\(_a\) vigilance parameter. A mismatch at the Map Field between the ART\(_a\) category activated by an input \(a\) and the ART\(_b\) category activated by the input \(b\) increases ART\(_a\) vigilance by the minimum amount needed for the system to search for and, if necessary, learn a new ART\(_a\) catego-
ry whose prediction matches the ART\(_b\) category.

This inter-ART vigilance resetting signal is a form of “back propagation” of information, but one that differs from the back propagation that occurs in the Back Propagation network. For example, the search initiated by inter-ART reset can shift attention to a novel cluster of visual features that can be incor-
porated through learning into a new ART\(_a\) recogni-
tion category. This process is analogous to learning a category for “green bananas” based on “taste” feedback. However, these events do not “back
propagate” taste features into the visual representa-
tion of the bananas, as can occur using the Back Propagation network. Rather, match tracking reor-
ganizes the way in which visual features are grouped, attended, learned, and recognized for purposes of predicting an expected taste.

The following sections describe ARTMAP simu-
lations using a machine learning benchmark data-
base. The ARTMAP system is then described mathematically. The Appendix summarizes ART 1 and ARTMAP system equations for purposes of sim-
ulation, and outlines system responses to various input protocols.

3. ARTMAP SIMULATIONS: Distinguishing Edible and Poisonous Mushrooms

The ARTMAP system was tested on a benchmark machine learning database that partitions a set of vectors \(a\) into two classes. Each vector \(a\) characterizes observable features of a mushroom as a binary vector, and each mushroom is classified as edible or poisonous (Schlimmer, 1987a). The database represents the 11 species of genus *Agaricus* and the 12 species of the genus *Lepiota* described in “The Audubon Society Field Guide to North American Mushrooms” (Lincoff, 1981). These two genera constitute most of the mushrooms described in the “Field Guide” from the family *Agaricaceae* (order *Agari-
cales*, class *Hymenomycetes*, subdivision *Basidiomy-
cetes*, division *Eumycota*). All the mushrooms re-
presented in the database are similar to one another: “These mushrooms are placed in a single family on the basis of a correlation of characteristics that include microscopic and chemical features…” (Lincoff, 1981, p. 500). The “Field Guide” warns that poisonous and edible species can be difficult to distinguish on the basis of their observable features. For example, the poisonous species *Agaricus cali-
ifornicus* is described as a “dead ringer” (Lincoff, 1981, p. 504) for the Meadow Mushroom, *Agaricus campestris*, that “may be known better and gathered more than any other wild mushroom in North America” (Lincoff, 1981, p. 505). This database thus pro-
vides a test of how ARTMAP and other machine learning systems distinguish rare but important events from frequently occurring collections of sim-
ilar events that lead to different consequences.

The database of 8,124 exemplars describes each of 22 observable features of a mushroom, along with its classification as poisonous (48.2%) or edible (51.8%). The 8,124 “hypothetical examples” re-
present ranges of characteristics within each species; for example, both *Agaricus californicus* and *Agaricus campestris* are described as having a “white to brownish cap,” so in the database each species has corre-
sponding sets of exemplar vectors representing their range of cap colors. There are 126 different values of the 22 different observable features. A list of the observable features and their possible values is given in Table 1. For example, the observable feature of “cap-shape” has six possible values. Consequently, the vector inputs to ART\(_a\) are 126-element binary vectors, each vector having 22 1’s and 104 0’s, to denote the values of an exemplar’s 22 observable features. The ART\(_a\) input vectors are \((1, 0)\) for poi-
sonous exemplars and \((0, 1)\) for edible exemplars.

3.1. Performance

The ARTMAP system learned to classify test vectors rapidly and accurately, and system performance compares favorably with results of other machine learning algorithms applied to the same database. The STAGGER algorithm reached its maximum per-
fomance level of 95% accuracy after exposure to 1,000 training inputs (Schlimmer, 1987b). The HIL-
LARY algorithm achieved similar results (Iba, Wogulis, & Langley, 1988). The ARTMAP system consistently achieved over 99% accuracy with 1,000 exemplars, even counting “I don’t know” responses as errors. Accuracy of 95% was usually achieved with on-line training on 300–400 exemplars and with off-
line training on 100–200 exemplars. In this sense, ARTMAP was an order of magnitude more efficient than the alternative systems. In addition, with con-
tinued training, ARTMAP predictive accuracy al-
ways improved to 100%. These results are elaborated below.
Almost every ARTMAP simulation was completed in under 2 minutes on an IRIS 4D computer, with total time ranging from about 1 minute for small training sets to 2 minutes for large training sets. This is comparable to 2–5 minutes on a SUN 4 computer. Each timed simulation included a total of 8,124 training and test samples, run on a time-sharing system with nonoptimized code. Each 1–2 minute computation included data read-in and read-out, training, testing, and calculation of multiple simulation indices.

### 3.2. On-Line Learning

On-line learning imitates the conditions of a human or machine operating in a natural environment. An input $a$ arrives, possibly leading to a prediction. If made, the prediction may or may not be confirmed. Learning ensues, depending on the accuracy of the prediction. Information about past inputs is available only through the present state of the system. Simulations of on-line learning by the ARTMAP system use each sample pair $(a, b)$ as both a test item and a training item. Input $a$ first makes a prediction that is compared with $b$. Learning follows as dictated by the internal rules of the ARTMAP architecture.

Four types of on-line simulations were carried out, using two different baseline settings of the ART$_a$ vigilance parameter $\rho_a$: $\rho_a = 0$ (forced choice condition) and $\rho_a = 0.7$ (conservative condition); and using sample replacement or no sample replacement. With sample replacement, any one of the 8,124 input samples was selected at random for each input presentation. A given sample might thus be repeatedly encountered while others were still unused. With no sample replacement, a sample was removed from the input pool after it was first encountered. The replacement condition had the advantage that repeated encounters tended to boost predictive accuracy. The no-replacement condition had the advantage of having learned from a somewhat larger set of inputs at each point in the simulation. The replacement and no-replacement conditions had similar performance indices, all other things being equal. Each of the 4 conditions was run on 10 independent simulations.

With $\rho_a = 0$, the system made a prediction in response to every input. Setting $\rho_a = 0.7$ increased the number of “I don’t know” responses, increased the number of ART$_a$ categories, and decreased the rate of incorrect predictions to nearly 0%, even early in training. The $\rho_a = 0.7$ condition generally outperformed the $\rho_a = 0$ condition, even when incorrect

<table>
<thead>
<tr>
<th>Number</th>
<th>Feature</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cap-Shape</td>
<td>Bell, Conical, Convex, Flat, Knobbed, Sunken</td>
</tr>
<tr>
<td>2</td>
<td>Cap-Surface</td>
<td>Fibrous, Grooves, Scaly, Smooth</td>
</tr>
<tr>
<td>3</td>
<td>Cap-Color</td>
<td>Brown, Buff, Gray, Green, Pink, Purple, Red, White, Yellow, Cinnamon</td>
</tr>
<tr>
<td>4</td>
<td>Bruises</td>
<td>Bruises, No Bruises</td>
</tr>
<tr>
<td>5</td>
<td>Odor</td>
<td>None, Almond, Anise, Creosote, Fishy, Foul, Musty, Pungent, Spicy</td>
</tr>
<tr>
<td>6</td>
<td>Gill-Attachment</td>
<td>Attached, Descending, Free, Notched</td>
</tr>
<tr>
<td>7</td>
<td>Gill-Spacing</td>
<td>Close, Crowded, Distant</td>
</tr>
<tr>
<td>8</td>
<td>Gill-Size</td>
<td>Broad, Narrow</td>
</tr>
<tr>
<td>9</td>
<td>Gill-Color</td>
<td>Brown, Buff, Orange, Gray, Green, Pink, Purple, Red, White, Yellow, Chocolate, Black</td>
</tr>
<tr>
<td>10</td>
<td>Stalk-Shape</td>
<td>Enlarging, Tapering</td>
</tr>
<tr>
<td>11</td>
<td>Stalk-Root</td>
<td>Bulbous, Club, Cup, Equal, Rhizomorphs, Rooted, Missing</td>
</tr>
<tr>
<td>12</td>
<td>Stalk-Surface-Above-Ring</td>
<td>Fibrous, Silky, Scaly, Smooth</td>
</tr>
<tr>
<td>13</td>
<td>Stalk-Surface-Below-Ring</td>
<td>Fibrous, Silky, Scaly, Smooth</td>
</tr>
<tr>
<td>14</td>
<td>Stalk-Color-Above-Ring</td>
<td>Brown, Buff, Orange, Gray, Pink, Red, White, Yellow, Cinnamon</td>
</tr>
<tr>
<td>15</td>
<td>Stalk-Color-Below-Ring</td>
<td>Brown, Buff, Orange, Gray, Pink, Red, White, Yellow, Cinnamon</td>
</tr>
<tr>
<td>16</td>
<td>Veil-Type</td>
<td>Partial, Universal</td>
</tr>
<tr>
<td>17</td>
<td>Veil-Color</td>
<td>Brown, Orange, White, Yellow</td>
</tr>
<tr>
<td>18</td>
<td>Ring-Number</td>
<td>None, One, Two</td>
</tr>
<tr>
<td>19</td>
<td>Ring-Type</td>
<td>None, Cobwebby, Evanscent, Flaring, Large, Pendant, Sheathing, Zone</td>
</tr>
<tr>
<td>20</td>
<td>Spore-Print-Color</td>
<td>Brown, Buff, Orange, Green, Purple, White, Yellow, Chocolate, Black</td>
</tr>
<tr>
<td>21</td>
<td>Population</td>
<td>Abundant, Clustered, Numerous, Scattered, Several, Solitary</td>
</tr>
<tr>
<td>22</td>
<td>Habitat</td>
<td>Grasses, Leaves, Meadows, Paths, Urban, Waste, Woods</td>
</tr>
</tbody>
</table>
predictions and “I don’t know” responses were both counted as errors. The primary exception occurred very early in training, when a conservative system gives the large majority of its no-prediction responses.

Results are summarized in Table 2. Each entry gives the number of correct predictions over the previous 100 trials (input presentations), averaged over 10 simulations. For example, with \( \bar{p}_a = 0 \) in the no-replacement condition, the system made, on the average, 94.9 correct predictions and 5.1 incorrect predictions on trials 201–300. In all cases a 95% correct-prediction rate was achieved before trial 400. With \( \bar{p}_a = 0 \), a consistent correct-prediction rate of over 99% was achieved by trial 1,400, while with \( \bar{p}_a = 0.7 \) the 99% consistent correct-prediction rate was achieved earlier, by trial 800. Each simulation was continued for 8,100 trials. In all four cases, the minimum correct-prediction rate always exceeded 99.5% by trial 1,800 and always exceeded 99.8% by trial 2,800. In all cases, across the total of 40 simulations summarized in Table 2, 100% correct prediction was achieved on the last 1,300 trials of each run.

Note the relatively low correct-prediction rate for \( \bar{p}_a = 0.7 \) on the first 100 trials. In the conservative mode, a large number of inputs initially make no prediction. With \( \bar{p}_a = 0.7 \) an average total of only 2 incorrect predictions were made on each run of 8,100 trials. Note too that Table 2 underestimates prediction accuracy at any given time, since performance almost always improves during the 100 trials over which errors are tabulated.

### 3.3. Off-Line Learning

In off-line learning, a fixed training set is repeatedly presented to the system until 100% accuracy is achieved on that set. For training sets ranging in size from 1 to 4,000 samples, 100% accuracy was almost always achieved after one or two presentations of each training set. System performance was then measured on the test set, which consisted of all 8,124 samples not included in the training set. During testing no further learning occurred.

The role of repeated training set presentations was examined by comparing simulations that used the 100% training set accuracy criterion with simulations that used only a single presentation of each input during training. With only a few exceptions, performance was similar. In fact, for \( \bar{p}_a = 0.7 \), and for small training sets with \( \bar{p}_a = 0 \), 100% training-set accuracy was achieved with single input presentations, so results were identical. Performance differences were greatest for \( \bar{p}_a = 0 \) simulations with mid-sized training sets (60–500 samples) when 2–3 training set presentations tended to add a few more ART learned category nodes. Thus, even a single presentation of training-then-testing inputs, carried out on-

**TABLE 2**

<table>
<thead>
<tr>
<th>Trial</th>
<th>( \bar{p}_a = 0 ) No Replace</th>
<th>( \bar{p}_a = 0 ) Replace</th>
<th>( \bar{p}_a = 0.7 ) No Replace</th>
<th>( \bar{p}_a = 0.7 ) Replace</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>82.9</td>
<td>81.9</td>
<td>66.4</td>
<td>67.3</td>
</tr>
<tr>
<td>200</td>
<td>89.8</td>
<td>89.6</td>
<td>87.8</td>
<td>87.4</td>
</tr>
<tr>
<td>300</td>
<td>94.9</td>
<td>92.6</td>
<td>94.1</td>
<td>93.2</td>
</tr>
<tr>
<td>400</td>
<td>95.7</td>
<td>95.9</td>
<td>96.8</td>
<td>95.8</td>
</tr>
<tr>
<td>500</td>
<td>97.8</td>
<td>97.1</td>
<td>97.5</td>
<td>97.8</td>
</tr>
<tr>
<td>600</td>
<td>98.4</td>
<td>98.2</td>
<td>98.1</td>
<td>98.2</td>
</tr>
<tr>
<td>700</td>
<td>97.7</td>
<td>97.9</td>
<td>98.1</td>
<td>99.7</td>
</tr>
<tr>
<td>800</td>
<td>98.1</td>
<td>97.7</td>
<td>99.0</td>
<td>99.0</td>
</tr>
<tr>
<td>900</td>
<td>98.3</td>
<td>98.6</td>
<td>99.2</td>
<td>99.0</td>
</tr>
<tr>
<td>1000</td>
<td>98.9</td>
<td>98.5</td>
<td>99.4</td>
<td>99.0</td>
</tr>
<tr>
<td>1100</td>
<td>98.7</td>
<td>98.9</td>
<td>99.2</td>
<td>99.7</td>
</tr>
<tr>
<td>1200</td>
<td>99.6</td>
<td>99.1</td>
<td>99.5</td>
<td>99.5</td>
</tr>
<tr>
<td>1300</td>
<td>99.3</td>
<td>98.8</td>
<td>99.8</td>
<td>99.8</td>
</tr>
<tr>
<td>1400</td>
<td>99.7</td>
<td>99.4</td>
<td>99.5</td>
<td>99.6</td>
</tr>
<tr>
<td>1500</td>
<td>99.5</td>
<td>99.0</td>
<td>99.7</td>
<td>99.6</td>
</tr>
<tr>
<td>1600</td>
<td>99.4</td>
<td>99.6</td>
<td>99.7</td>
<td>99.8</td>
</tr>
<tr>
<td>1700</td>
<td>98.9</td>
<td>99.3</td>
<td>99.8</td>
<td>99.8</td>
</tr>
<tr>
<td>1800</td>
<td>99.5</td>
<td>99.2</td>
<td>99.8</td>
<td>99.9</td>
</tr>
<tr>
<td>1900</td>
<td>99.8</td>
<td>99.9</td>
<td>99.9</td>
<td>99.9</td>
</tr>
<tr>
<td>2000</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
</tr>
</tbody>
</table>
line, can be made to work almost as well as off-line training that uses repeated presentations of the training set. This is an important benefit of fast learning controlled by a match tracked search.

Under all training conditions, each of the 8,124 ART\(_c\) input vectors is a 126-dimensional binary vector with 22 positive entries. Simulation dynamics are illustrated by projecting these vectors onto the first two principal components of the data set (Kendall & Stuart, 1966). These two components represent 31% of the total variance of the data set.

Figure 3a shows the projections of all 3,916 exemplars representing poisonous mushrooms, and Figure 3b shows the 4,208 exemplars representing edible mushrooms. These figures show that, in these two dimensions, certain clusters are readily distinguishable, such as the clusters of poisonous samples on the top and left portions of Figure 3a. However, poisonous and edible samples are densely mixed near the positive x-axis.

3.4. Off-Line Forced-Choice Learning

The simulations summarized in Figure 4 and Table 3 illustrate off-line learning with $r = 0$. In this forced choice case, each ART\(_c\) input led to a prediction of poisonous or edible. The number of test set errors with small training sets was relatively large, due to the forced choice.

Figure 4 shows the evolution of test set errors as the training set is increased in size from 5 to 500. In Figure 4a, a set of 5 randomly chosen exemplars (3 poisonous, 2 edible) established 2 ART\(_c\) categories (1 poisonous, 1 edible) during training. For each of the 8,119 test set exemplars, the system was forced to choose between poisonous and edible, even if no category representation was a close match. The system made 73% correct predictions. Many of the errors were in the dense cluster of poisonous exemplars in the upper quarter of the graph (Figure 3a). By chance, this cluster was not represented in the 5-sample training set.

Table 3 summarizes the average results over 10 simulations at each size training set. For example, with very small, 5-sample training sets, the system established between 1 and 5 ART\(_c\) categories, and averaged 73.1% correct responses on the remaining 8,119 test patterns. Success rates ranged from chance (51.8%, 1 category) in one instance where all 5 training set exemplars happened to be edible, to surprisingly good (94.2%, 2 categories). The range of success rates for fast-learn training on very small training sets illustrates the statistical nature of the learning process. Intelligent sampling of the training set or, as here, good luck in the selection of representative samples, can dramatically alter early success rates. In addition, the evolution of internal category memory structure, represented by a set of ART\(_c\) category nodes and their top-down learned expectations, is influenced by the selection of early exemplars. Nevertheless, despite the individual nature of learning rates and internal representations, all the systems eventually converge to 100% accuracy on test set exemplars using only (approximately) 1/600 as many ART\(_c\) categories as there are inputs to classify.

Figure 4 and Table 3 summarize the rate at which learning converges to 100% accuracy. In Figure 4b, 25 exemplars were added to the 5 used for Figure 4a, and the resulting 30-sample training set was presented to a new ARTMAP system. The 25 additional training exemplars increased the number of ART\(_c\) categories to 3 and improved the test set correct-prediction rate to 92.3%. The addition of poisonous training exemplars in the upper quarter of the graph.

FIGURE 3. Mushroom observable feature data projected onto first 2 principal components. Each point represents a 126-dimensional ART\(_c\) input vector. Axes are scaled to run from -1 to +1. (a) 3,916 exemplars representing poisonous mushrooms (48.2%); (b) 4,208 exemplars representing edible mushrooms (51.8%).
FIGURE 4. Training sets of increasing size (left column) and test set exemplars that were incorrectly classified (right column), projected onto first two principal components. Baseline vigilance \( \rho_c \) equals 0. (a) With a 5-sample training set that established 2 ART\(_1\) categories, the test set of 8,119 inputs made 2,194 errors (27.0%). On 10 other 5-sample runs, the number of ART\(_1\) categories ranged from 1 to 5 and the error rate ranged from 5.8% to 48.2%, averaging 33.9%; (b) With a 30-sample training set that established 3 ART\(_1\) categories, the test set of 8,094 inputs made 624 errors (7.7%). On 10 other 30-sample training runs, the number of ART\(_1\) categories ranged from 4 to 6; and the error rate ranged from 6.7% to 25.1%, averaging 12.4%; (c) With a 125-sample training set that established 9 ART\(_1\) categories, the test set of 7,999 inputs made 288 errors (3.6%). On 10 other 125-training sample runs, the number of ART\(_1\) categories ranged from 5 to 14, and the error rate ranged from 1.2% to 8.5%, averaging 4.4%; (d) With a 500-sample training set that established 15 ART\(_1\) categories, the test set of 7,824 inputs made 168 errors (2.2%). On 10 other 500-training sample runs, the number of ART\(_1\) categories ranged from 9 to 22; and the error rate ranged from 0.7% to 3.1%, averaging 1.6%.
eliminated all errors there. However, errors persisted for exemplars near the positive x-axis. On 10 other simulations with 30-sample training sets, the correct prediction rate averaged 87.6% and ranged from 74.9% (4 categories) to 93.3% (6 categories).

The simulation that generated Figure 4c added 95 training samples to the 30 used for Figure 4b. The number of ARTc categories increased to 9 and the correct prediction rate increased to 96.4%. On 10 other simulations with 125 randomly chosen training exemplars, the correct-prediction rate averaged 95.6%, ranging from 91.5% (10 categories) to 98.8% (9 categories).

The simulation of Figure 4d added 375 samples to the set used in Figure 4c. This 500-sample training set increased the correct-prediction rate to 97.8% on the test set, establishing 15 categories. On 10 other runs, each with 500 randomly chosen training exemplars, the correct-prediction rate averaged 98.4%, ranging from 96.9% (14 categories) to 99.3% (9 categories). The low error rate of this latter 9-category simulation appears to reflect success of early sampling. On other runs, additional categories were added as errors in early category structures were detected.

With 1,000-sample training sets, 3 out of 10 simulations achieved 100% prediction accuracy on the 7,124-sample test set. With 2,000-sample training sets, 8 out of 10 simulations achieved 100% accuracy on the 6,124-sample test sets. With 4,000-sample training sets, all simulations achieved 100% accuracy on the 4,124-sample test sets. In all, 21 of the 30 simulations with training sets of 1,000, 2,000, and 4,000 samples achieved 100% accuracy on test sets. The number of categories established during these 21 simulations ranged from 10 to 22, again indicating the variety of paths leading to 100% correct prediction rate.

3.5. Off-Line Conservative Learning

As in the case of poisonous mushroom identification, it may be important for a system to be able to respond "I don't know" to a novel input, even if the total number of correct classifications thereby decreases early in learning. For higher values of the baseline vigilance $p_0$, the ARTMAP system creates more ARTc categories during learning and becomes less able to generalize from prior experience than when $p_0$ equals 0. During testing, a conservative coding system with $p_0 = 0.7$ makes no prediction in response to inputs that are too novel, and thus initially has a lower proportion of correct responses. However, the number of incorrect responses is always low with $p_0 = 0.7$, even with very few training samples, and the 99% correct-response rate is achieved for both forced choice ($p_0 = 0$) and conservative ($p_0 = 0.7$) systems with training sets smaller than 1,000 exemplars.

Table 4 summarizes simulation results that repeat

### Table 4

<table>
<thead>
<tr>
<th>Training Set Size</th>
<th>Average % Correct (Test Set)</th>
<th>Average % Incorrect (Test Set)</th>
<th>Average % No-Response (Test Set)</th>
<th>Number of ARTc Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>25.6</td>
<td>0.6</td>
<td>73.8</td>
<td>2–3</td>
</tr>
<tr>
<td>5</td>
<td>41.1</td>
<td>0.4</td>
<td>58.5</td>
<td>3–5</td>
</tr>
<tr>
<td>15</td>
<td>57.6</td>
<td>1.1</td>
<td>41.3</td>
<td>6–10</td>
</tr>
<tr>
<td>30</td>
<td>62.3</td>
<td>0.9</td>
<td>36.8</td>
<td>14–18</td>
</tr>
<tr>
<td>60</td>
<td>78.5</td>
<td>0.8</td>
<td>20.8</td>
<td>21–27</td>
</tr>
<tr>
<td>125</td>
<td>83.1</td>
<td>0.7</td>
<td>16.1</td>
<td>33–37</td>
</tr>
<tr>
<td>250</td>
<td>92.7</td>
<td>0.3</td>
<td>7.0</td>
<td>42–51</td>
</tr>
<tr>
<td>500</td>
<td>97.7</td>
<td>0.1</td>
<td>2.1</td>
<td>48–64</td>
</tr>
<tr>
<td>1000</td>
<td>99.4</td>
<td>0.04</td>
<td>0.5</td>
<td>53–66</td>
</tr>
<tr>
<td>2000</td>
<td>100.0</td>
<td>0.00</td>
<td>0.05</td>
<td>54–69</td>
</tr>
<tr>
<td>4000</td>
<td>100.0</td>
<td>0.00</td>
<td>0.02</td>
<td>61–73</td>
</tr>
</tbody>
</table>
the conditions of Table 3 except that $\bar{p}_c = 0.7$. Here, a test input that does not make a 70% match with any learned expectation makes an "I don't know" prediction. Compared with the $\bar{p}_c = 0$ case of Table 3, Table 4 shows that larger training sets are required to achieve a correct prediction rate of over 95%. However, because of the option to make no prediction, the average test set error rate is almost always less than 1%, even when the training set is very small, and is less than .1% after only 500 training trials. Moreover, 100% accuracy is achieved using only (approximately) $1/130$ as many ART, categories as there are inputs to classify.

3.6. Category Structure

Each ARTMAP category code can be described as a set of ART, feature values on 1 to 22 observable features, chosen from 126 feature values, that are associated with the ART, identification as poisonous or edible. During learning, the number of feature values that characterize a given category is monotone decreasing, so that generalization within a given category tends to increase. The total number of classes can, however, also increase, which tends to decrease generalization. Increasing the number of training patterns hereby tends to increase the number of categories and decrease the number of critical feature values of each established category. The balance between these opposing tendencies leads to the final net level of generalization.

Table 5 illustrates the long term memory structure underlying the 125-sample forced-choice simulation shown in Figure 4c. Of the 9 categories established at the end of the training phase, 4 are identified as poisonous (P) and 5 are identified as edible (E). Each ART, category assigns a feature value to a subset of the 22 observable features. For example, Category 1 (poisonous) specifies values for 5 features, and leaves the remaining 17 features unspecified. The corresponding ART, weight vector has 5 ones and 121 zeros. Note that the features that characterize Category 5 (poisonous) form a subset of the features that characterize Category 6 (edible). Recall that this category structure gave 96.4% correct responses on the 7,999 test set samples, which are partitioned as shown in the last line of Table 5. When 100% accuracy is achieved, a few categories with a small number of specified features typically code large clusters, while a few categories with many specified features code small clusters of rare samples.

Table 6 illustrates the statistical nature of the coding process, which leads to a variety of category structures when fast learning is used. Test set prediction accuracy of the simulation that generated Table 6 was similar to that of Table 5, and each simulation

<table>
<thead>
<tr>
<th>#</th>
<th>Feature</th>
<th>1 = P</th>
<th>2 = E</th>
<th>3 = E</th>
<th>4 = E</th>
<th>5 = P</th>
<th>6 = E</th>
<th>7 = P</th>
<th>8 = P</th>
<th>9 = E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cap-Shape</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Cap-Surface</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Cap-Color</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Bruises?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Odor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Gill-Spacing</td>
<td>Close</td>
<td>Close</td>
<td>Close</td>
<td>Close</td>
<td>Close</td>
<td>Close</td>
<td>Close</td>
<td>Close</td>
<td>Close</td>
</tr>
<tr>
<td>8</td>
<td>Gill-Size</td>
<td>Broad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Gill-Color</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Stalk-Shape</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Stalk-Root</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Stalk-Surface-Above-Ring</td>
<td>Smooth</td>
<td>Smooth</td>
<td>Smooth</td>
<td>Smooth</td>
<td>Smooth</td>
<td>Smooth</td>
<td>Smooth</td>
<td>Smooth</td>
<td>Smooth</td>
</tr>
<tr>
<td>13</td>
<td>Stalk-Surface-Below-Ring</td>
<td>Smooth</td>
<td></td>
<td>White</td>
<td>White</td>
<td>White</td>
<td>Pink</td>
<td>White</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Stalk-Color-Above-Ring</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
</tr>
<tr>
<td>15</td>
<td>Stalk-Color-Below-Ring</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
</tr>
<tr>
<td>16</td>
<td>Veil-Type</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
</tr>
<tr>
<td>17</td>
<td>Veil-Color</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
</tr>
<tr>
<td>18</td>
<td>Ring-Number</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
</tr>
<tr>
<td>19</td>
<td>Ring-Type</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
<td>One</td>
</tr>
<tr>
<td>20</td>
<td>Spore-Print-Color</td>
<td>Partial</td>
<td>White</td>
<td>Partial</td>
<td>White</td>
<td>Partial</td>
<td>White</td>
<td>Partial</td>
<td>White</td>
<td>Partial</td>
</tr>
<tr>
<td>21</td>
<td>Population</td>
<td>Several</td>
<td>Several</td>
<td>Scattered</td>
<td>Several</td>
<td>Scattered</td>
<td>Several</td>
<td>Scattered</td>
<td>Several</td>
<td>Scattered</td>
</tr>
<tr>
<td>22</td>
<td>Habitat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

# Coded/Category: 2367 1257 387 1889 756 373 292 427 251
had a 125-sample training set. However, the simulation of Table 6 produced only 4 ART\textsubscript{a} categories, only one of which (Category 1) has the same long-term memory representation as Category 2 in Table 5. Note that, at this stage of coding, certain features are uninformative. For example, no values are specified for features 1, 2, 3, or 22 in Table 5 or Table 6; and feature 16 (veil-type) always has the value “partial.” However, performance is still only around 96%. As rare instances form small categories later in the coding process, some of these features may become critical in identifying exemplars of small categories.

We will now turn to a description of the components of the ARTMAP system.

4. ART MODULES ART\textsubscript{a} and ART\textsubscript{b}

Each ART module in Figures 1 and 2 establishes compressed recognition codes in response to sequences of input patterns \( \mathbf{a} \) and \( \mathbf{b} \). Associative learning at the Map Field links pairs of pattern classes via these compressed codes. One type of generalization follows immediately from this learning strategy: If one vector \( \mathbf{a} \) is associated with a vector \( \mathbf{b} \), then any other input that activates \( \mathbf{a} \)'s category node will predict the category of pattern \( \mathbf{b} \). Any ART module can be used to self-organize the ART\textsubscript{a} and ART\textsubscript{b} categories. In the application above, \( \mathbf{a} \) and \( \mathbf{b} \) are binary vectors, so ART\textsubscript{a} and ART\textsubscript{b} can be ART 1 modules. The main computations of an ART 1 module will here be outlined. A full definition of ART 1 modules, as systems of differential equations, along with an analysis of their network dynamics, can be found in Carpenter and Grossberg (1987a).

In an ART 1 module, an input pattern \( \mathbf{I} \) is represented in field \( F_1 \) and the recognition category for \( \mathbf{I} \) is represented in field \( F_2 \). We consider the case where the competitive field \( F_2 \) makes a choice and where the system is operating in a fast-learn mode, as defined below. An algorithm for simulations is given in the Appendix.

### 4.1. \( F_1 \) Activation

Figure 5 illustrates the main components of an ART 1 module. A field, \( F_1 \) of \( M \) nodes, with output vector \( \mathbf{x} = (x_1, \ldots, x_M) \), registers the \( F_0 \rightarrow F_1 \) input vector \( \mathbf{I} = (I_1, \ldots, I_M) \). Each \( F_1 \) node can receive input from 3 sources: the \( F_0 \rightarrow F_1 \) bottom-up input; nonspecific gain control signals; and top-down signals from the \( N \) nodes of \( F_2 \), via an \( F_2 \rightarrow F_1 \) adaptive filter. A node is said to be active if it generates an output signal equal to 1. Output from inactive nodes equals 0. In ART 1 an \( F_1 \) node is active if at least 2 of the 3 input signals are large. This rule for \( F_1 \) activation is called the 2/3 Rule. The 2/3 Rule is realized in its simplest, dimensionless form as follows:

<table>
<thead>
<tr>
<th>#</th>
<th>Feature</th>
<th>1 = E</th>
<th>2 = P</th>
<th>3 = P</th>
<th>4 = E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cap-Shape</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Cap-Surface</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Cap-Color</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Bruises?</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Odor</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Gill-Attachment</td>
<td>Free</td>
<td>Free</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Gill-Spacing</td>
<td>Close</td>
<td>Close</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Gill-Size</td>
<td>Broad</td>
<td>Broad</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Gill-Color</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Stalk-Shape</td>
<td>Enlarging</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Stalk-Root</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Stalk-Surface-Above-Ring</td>
<td>Smooth</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Stalk-Surface-Below-Ring</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Stalk-Color-Above-Ring</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Stalk-Color-Below-Ring</td>
<td>White</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Veil-Type</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
</tr>
<tr>
<td>17</td>
<td>Veil-Color</td>
<td>White</td>
<td>White</td>
<td>White</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Ring-Number</td>
<td>One</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Ring-Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Spore-Print-Color</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Population</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Habitat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

# Coded/Category: 3099 1820 2197 883

**TABLE 6**
Critical Feature Values of the 4 Prototypes Learned in a 125-Sample Simulation With a Training Set Different From the One in Table 5. Prediction Accuracy Is Similar (96.0%), but the ART\textsubscript{a} Category Boundaries Are Different.
2/3 Rule matching

The $i$th $F_i$ node is active if its net input exceeds a fixed threshold. Specifically,

$$
x_i = \begin{cases} 
1 & \text{if } l_i + g_i + \sum_{j=1}^{n} y_j z_{ij} > 1 + z, \\
0 & \text{otherwise}, 
\end{cases} \tag{1}
$$

where term $l_i$ is the binary $F_0 \rightarrow F_i$ input, term $g_i$ is the binary nonspecific $F_i$ gain control signal, term $\sum_{j=1}^{n} y_j z_{ij}$ is the sum of $F_2 \rightarrow F_i$ signals $y_j$ via pathways with adaptive weights $z_{ij}$, and $z$ is a constant such that

$$0 < z < 1. \tag{2}$$

$F_i$ gain control

The $F_i$ gain control signal $g_i$ is defined by

$$g_i = \begin{cases} 
1 & \text{if } F_0 \text{ is active and } F_i \text{ is inactive} \\
0 & \text{otherwise}. 
\end{cases} \tag{3}
$$

Note that $F_2$ activity inhibits $F_i$ gain, as shown in Figure 5. These laws for $F_i$ activation imply that, if $F_2$ is inactive,

$$x_i = \begin{cases} 
1 & \text{if } l_i = 1 \\
0 & \text{otherwise}. 
\end{cases} \tag{4}
$$

If exactly one $F_2$ node $J$ is active, the sum $\sum_{j=1}^{n} y_j z_{ij}$ in eqn (1) reduces to the single term $z_{ij}$, so

$$x_i = \begin{cases} 
1 & \text{if } l_i = 1 \text{ and } z_{ij} > z \\
0 & \text{otherwise}. 
\end{cases} \tag{5}
$$

4.2. $F_2$ Choice

Let $T_j$ denote the total input from $F_i$ to the $j$th $F_2$ node, given by

$$T_j = \sum_{i=1}^{n} x_i z_{ij}. \tag{6}
$$

where the $Z_{ij}$ denote the $F_i \rightarrow F_2$ adaptive weights. If some $T_j > 0$, define the $F_2$ choice index $J$ by

$$J = \max\{T_j : j = 1 \ldots N\}. \tag{7}
$$

In the typical case, $J$ is uniquely defined. Then the $F_2$ output vector $y = (y_1, \ldots, y_N)$ obeys

$$y_j = \begin{cases} 
1 & \text{if } j = J \\
0 & \text{if } j \neq J. 
\end{cases} \tag{8}
$$

If two or more indices $j$ share maximal input, they equally share the total activity. This case is not considered here.

ART 1

![ART 1 schematic diagram](image)

**FIGURE 5.** ART 1 schematic diagram (Carpenter & Grossberg, 1987a). The binary vector $I$ forms the bottom-up input to the field $F_2$ whose activity vector is denoted $x$. The competitive field $F_1$ is designed to make a choice. Adaptive pathways lead from each $F_i$ node to all $F_2$ nodes, and from each $F_2$ node to all $F_i$ nodes. Reset occurs when the match between $x$ and $I$ fails to meet the criterion established by the vigilance parameter $\rho$. All paths are excitatory unless marked with a minus sign.
4.3. Learning Laws

In fast-learn ART 1, adaptive weights reach their new asymptote on each input presentation. The learning laws, as well as the rules for choice and search, are conveniently described using the following notation. If \( \mathbf{a} \) is a binary \( M \)-vector, define the norm of \( \mathbf{a} \) by

\[
|\mathbf{a}| = \sum_{i=1}^{M} a_i.
\]

If \( \mathbf{a} \) and \( \mathbf{b} \) are two binary vectors, define a third binary vector \( \mathbf{a} \cap \mathbf{b} \) by

\[
(a \cap b)_i = 1 \leftrightarrow a_i = 1 \text{ and } b_i = 1.
\]

Finally, let \( \mathbf{a} \) be a subset of \( \mathbf{b} \) (\( \mathbf{a} \subseteq \mathbf{b} \)) iff \( \mathbf{a} \cap \mathbf{b} = \mathbf{a} \).

All ART 1 learning is gated by \( F_2 \) activity; that is, the adaptive weights \( z_{ji} \) and \( z_{ij} \) can change only when the \( J \)th \( F_2 \) node is active. Then both \( F_2 \to F_1 \) and \( F_1 \to F_2 \) weights are functions of the \( F_1 \) vector \( \mathbf{x} \), as follows:

**Top-down learning**

Top-down \( F_2 \to F_1 \) weights in active paths learn \( \mathbf{x} \); that is, when the \( J \)th \( F_2 \) node is active

\[
z_{ji} \to x_i.
\]

All other \( z_{ji} \) remain unchanged. Stated as a differential equation, this learning rule is

\[
dt z_{ji} = y_j(x_i - z_{ji}).
\]

In eqn (12), learning by \( z_{ji} \) is gated by \( y_j \). When the \( y_j \) gate opens—that is, when \( y_j > 0 \)—then learning begins and \( z_{ji} \) is attracted to \( x_i \). In vector terms, if \( y_j > 0 \), then \( \mathbf{z}_j = (z_{j1}, z_{j2}, \ldots, z_{jM}) \) approaches \( \mathbf{x} \). Such a law is therefore sometimes called learning by gated steepest descent. It is also called the outstar learning rule, and was introduced into the neural modelling literature in 1969 (Grossberg, 1969).

Initially all \( z_{ji} \) are maximal:

\[
z_{ji}(0) = 1.
\]

Thus with fast learning, the top-down weight vector \( \mathbf{z}_j \) is a binary vector at the start and end of each input presentation. By eqns (4), (5), (10), (11), and (13), the \( F_1 \) activity vector can be described as

\[
\mathbf{x} = \begin{cases} 
\mathbf{I} & \text{if } F_1 \text{ is inactive} \\
\mathbf{I} \cap \mathbf{z}_j & \text{if } \text{the } J \text{th } F_2 \text{ node is active.}
\end{cases}
\]

By eqns (5) and (12), when node \( J \) is active, learning causes

\[
z_j \to \mathbf{I} \cap \mathbf{z}_j^{(ad)},
\]

where \( \mathbf{z}_j^{(ad)} \) denotes \( \mathbf{z}_j \) at the start of the input presentation. By eqns (11) and (14), \( \mathbf{x} \) remains constant during learning, even though \( |\mathbf{z}_j| \) may decrease.

The first time an \( F_2 \) node \( J \) becomes active, it is said to be uncommitted. Then, by eqns (13)–(15),

\[
z_j \to \mathbf{I}
\]

during learning. Thereafter node \( J \) is said to be committed.

**Bottom-up learning**

In simulations it is convenient to assign initial values to the bottom-up \( F_1 \to F_2 \) adaptive weights \( z_{ij} \) in such a way that \( F_2 \) nodes first become active in the order \( j = 1, 2, \ldots \). This can be accomplished by letting

\[
Z_{ij}(0) = \alpha_i,
\]

where

\[
\alpha_1 > \alpha_2 > \ldots > \alpha_N.
\]

Like the top-down weight vector \( \mathbf{z}_j \), the bottom-up \( F_1 \to F_2 \) weight vector \( \mathbf{Z}_j = (Z_{j1}, Z_{j2}, \ldots, Z_{jM}) \) also becomes proportional to the \( F_2 \) output vector \( \mathbf{x} \) when the \( F_2 \) node \( J \) is active. In addition, however, the bottom-up weights are scaled inversely to \( |\mathbf{x}| \), so that

\[
Z_{ij} \to \frac{x_i}{\beta + |\mathbf{x}|},
\]

where \( \beta > 0 \). This \( F_1 \to F_2 \) learning law, called the Weber Law Rule (Carpenter & Grossberg, 1987a), realizes a type of competition among the weights \( \mathbf{z}_j \) adjacent to a given \( F_2 \) node \( J \). This competitive computation could alternatively be transferred to the \( F_1 \) field, as it is in ART 2 (Carpenter & Grossberg, 1987b). By eqns (14), (15), and (19), during learning

\[
\mathbf{Z}_j \to \frac{I \cap \mathbf{z}_j^{(ad)}}{\beta + |I \cap \mathbf{z}_j^{(ad)}|}.
\]

The \( \alpha_i \) initial values are required to be small enough so that an input \( \mathbf{I} \) that perfectly matches a previously learned vector \( \mathbf{Z}_j \) will select the \( F_2 \) node \( J \) rather than an uncommitted node. This is accomplished by assuming that

\[
0 < \alpha_i = Z_{ij}(0) < \frac{1}{\beta + |\mathbf{I}|}
\]

for all \( F_0 \to F_1 \) inputs \( \mathbf{I} \). When \( \mathbf{I} \) is first presented, \( \mathbf{x} = \mathbf{I} \), so by eqns (6), (15), (17), and (20), the \( F_1 \to F_2 \) input vector \( \mathbf{T} = (T_1, T_2, \ldots, T_N) \) is given by

\[
T_i = \sum_{j=1}^{N} t_{ij} = \begin{cases} 
|I| \alpha_i & \text{if } j \text{ is an uncommitted node} \\
|I \cap z_j|/(\beta + |z_j|) & \text{if } j \text{ is a committed node.}
\end{cases}
\]

In the simulations above, \( \beta \) is taken to be so small that, among committed nodes, \( T_j \) is determined by the size of \( |I \cap z_j| \) relative to \( |z_j| \). If \( \beta \) were large, \( T_j \) would depend primarily on \( |I \cap z_j| \). In addition, \( \alpha_i \)
values are taken to be so small that an uncommitted node will generate the maximum $T_j$ value in eqn (22) only if $|I \cap z_j| = 0$ for all committed nodes. Larger values of $\alpha$, and $\beta$ bias the system toward earlier selection of uncommitted nodes when only poor matches are to be found among the committed nodes. A more complete discussion of this aspect of ART 1 system design is given by Carpenter and Grossberg (1987a).

4.4. Hypothesis Testing, Confidence, Novelty, and Search

By eqns (7), (21), and (22), a committed $F_2$ node $J$ may be chosen even if the match between $I$ and $z_j$ is poor; the match need only be the best one available. If the match is too poor, then the ART 1 system can autonomously carry out hypothesis testing, or search, for a better $F_2$ recognition code. This search process is mediated by the orienting subsystem, which can reset $F_2$ nodes in response to poor matches at $F_1$ (Figure 5). The orienting subsystem is a type of novelty detector that measures system confidence. If the degree of match between bottom-up input $I$ and top-down weight vector $z_j$ is too poor, the system’s confidence in the recognition code labelled by $J$ is inadequate. Otherwise expressed, the input $I$ is too unexpected relative to the top-down vector $z_j$, which plays the role of a learned top-down expectation.

An unexpected input triggers a novelty burst at the orienting subsystem, which sends a nonspecific reset wave $r$ from the orienting subsystem to $F_2$. The reset wave endurably shuts off node $J$ so long as input $I$ remains on. With $J$ off and its top-down $F_2 \rightarrow F_1$ signals silent, $F_1$ can again instate vector $x = I \cap z_j$, which leads to selection of another $F_2$ node through the bottom-up $F_1 \rightarrow F_2$ adaptive filter. This hypothesis testing process leads to activation of a sequence of $F_2$ nodes until one is chosen whose vector of adaptive weights forms an adequate match with $I$, or until an uncommitted node is selected. The search takes place so rapidly that essentially no learning occurs on that time scale. Learned weights are hereby buffered against recoding by poorly matched inputs that activate unacceptable $F_2$ recognition codes. Thus, during search, previously learned weights actively control the search for a better recognition code without being changed by the signals that they process.

4.5. Vigilant Search and Resonant Learning

As noted above, the degree of match between bottom-up input $I$ and top-down expectation $z_j$ is evaluated at the orienting subsystem, which measures system confidence that category $J$ adequately represents input $I$. A reset wave is triggered only if this confidence measure falls below a dimensionless parameter $\rho$ that is called the vigilance parameter. The vigilance parameter calibrates the system’s sensitivity to disconfirmed expectations.

One of the main reasons for the successful classification of nonstationary data sequences by ARTMAP is its ability to recalibrate the vigilance parameter based on predictive success. How this works will be described below. For now, we characterize the ART 1 search process given a constant level of vigilance.

In fast-learn ART 1 with choice at $F_2$, the search process occurs as follows:

Step 1—Select one $F_2$ node $J$ that maximizes $T_j$ in eqn (22), and read-out its top-down weight vector $z_j$.

Step 2—With $J$ active, compare the $F_1$ output vector $x = I \cap z_j$ with the $F_1 \rightarrow F_2$ input vector $I$ at the orienting subsystem (Figure 5).

Step 3A—Suppose that $I \cap z_j$ fails to match $I$ at the level required by the vigilance criterion, i.e., that

$$|x| = |I \cap z_j| < \rho |I|.$$  

Then $F_2$ reset occurs: node $J$ is shut off for the duration of the input interval during which $I$ remains on. The index of the chosen $F_2$ node is reset to the value corresponding to the next highest $F_1 \rightarrow F_2$ input $T_j$. With the new node active, Steps 2 and 3A are repeated until the chosen node satisfies the resonance criterion in Step 3B. Note that reset never occurs if

$$\rho \leq 0.$$  

When eqn (24) holds, an ART system acts as if there were no orienting subsystem.

Step 3B—Suppose that $I \cap z_j$ meets the criterion for resonance; i.e., that

$$|x| = |I \cap z_j| \geq \rho |I|.$$  

Then the search ceases and the last chosen $F_2$ node $J$ remains active until input $I$ shuts off (or until $\rho$ increases). In this state, called resonance, both the $F_1 \rightarrow F_2$ and the $F_2 \rightarrow F_1$ adaptive weights approach new values if $I \cap z_j^{(old)} \neq z_j^{(old)}$. Note that resonance cannot occur if $\rho > 1$.

If $\rho = 0$, search ceases whenever $I \subseteq z_j$, as is the case if an uncommitted node $J$ is chosen. If vigilance is close to 1, then reset occurs if $F_1 \rightarrow F_2$ input alters the $F_1$ activity pattern at all; resonance requires that $I$ be a subset of $z_j$. If vigilance is near 0, reset never occurs. The top-down expectation $z_j$ of the first chosen $F_2$ node $J$ is then recoded from $z_j^{(old)}$ to $I \cap z_j^{(old)}$, even if $I$ and $z_j^{(old)}$ are very different vectors.
4.6. $F_2$ Gain Control

For simplicity, ART 1 is exposed to discrete presentation intervals during which an input is constant and after which $F_1$ and $F_2$ activities are set to zero. Discrete presentation intervals are implemented in ART 1 by means of the $F_1$ and $F_2$ gain control signals $g_1$ and $g_2$ (Figure 5). The $F_2$ gain signal $g_2$ is assumed, like $g_1$ in eqn (3), to be 0 if $F_0$ is inactive. Then, when $F_0$ becomes active, $g_2$ and $F_2$ signal thresholds are assumed to lie in a range where the $F_2$ node that receives the largest input signal can become active. When an ART 1 system is embedded in a hierarchy, $F_2$ may receive signals from sources other than $F_1$. This occurs in the ARTMAP system described below. In such a system, $F_2$ still makes a choice and gain signals from $F_0$ are still required to generate both $F_1$ and $F_2$ output signals. In the simulations, $F_2$ nodes that are reset during search remain off until the input shuts off. A real-time ART search mechanism that can cope with continuously fluctuating analog or binary inputs of variable duration, fast or slow learning, and compressed or distributed $F_2$ codes is described by Carpenter and Grossberg (1990).

5. THE MAP FIELD

A Map Field module links the $F_2$ fields of the ART$_a$ and ART$_b$ modules. Figure 6 illustrates the main components of the Map Field. We will describe one such system in the fast-learn mode with choice at the fields $F^a_2$ and $F^b_2$. As with the ART 1 and ART 2 architectures themselves (Carpenter & Grossberg, 1987a, 1987b), many variations of the network architecture lead to similar computations. In the ARTMAP hierarchy, ART$_a$, ART$_b$, and Map Field modules are all described in terms of ART 1 variables and parameters. Indices $a$ and $b$ identify terms in the ART$_a$ and ART$_b$ modules, while Map Field variables and parameters have no such index. Thus, for example, $\rho_a$, $\rho_b$, and $\rho$ denote the ART$_a$, ART$_b$, and Map Field vigilance parameters, respectively.

5.1. ART$_a$, ART$_b$, and Complement Coding

Both ART$_a$ and ART$_b$ are fast-learn ART 1 modules. With one optional addition, they duplicate the design described above. That addition, called complement coding (Carpenter, Grossberg, & Rosen, 1991), rep-
represents both the on-response to an input vector and the off-response to that vector. This ART coding strategy has been shown to play a useful role in searching for appropriate recognition codes in response to predictive feedback (Grossberg, 1982b, 1984). To represent such a code in its simplest form, let the input vector \( \mathbf{a} \) itself represent the on-response, and the complement of \( \mathbf{a} \), denoted by \( \mathbf{a}' \), represent the off-response, for each ART\(_n\) input vector \( \mathbf{a} \). If \( \mathbf{a} \) is the binary vector \((a_1, \ldots, a_M, 1)\), the input to ART\(_n\) is the \(2M+1\)-dimensional binary vector

\[
(a, a') = (a_1, \ldots, a_M, a_1, \ldots, a_M)
\]

where

\[
a_i' = 1 - a_i.
\]

The utility of complement coding for searching an ARTMAP system will be described below. Conditions will also be given where complement coding is not needed. In fact, complement coding was not needed for any of the simulations described above, and the ART\(_n\) input was simply the vector \( \mathbf{a} \).

In the discussion of the Map Field module below, \( F_2 \) nodes, indexed by \( j = 1 \ldots N_a \), have binary output signals \( y_j \), and \( F_1 \) nodes, indexed by \( k = 1 \ldots N_b \), have binary output signals \( x_k \). Correspondingly, the index of the active \( F_2 \) node is denoted by \( J \), and the index of the active \( F_1 \) node is denoted by \( K \). Because the Map Field is the interface where signals from \( F_1 \) and \( F_2 \) interact, it is denoted by \( F_{ab} \). The nodes of \( F_{ab} \) have the same index \( k = 1, 2, \ldots, N_b \) as the nodes of \( F_2 \) because there is a one-to-one correspondence between these sets of nodes. The output signals of \( F_{ab} \) nodes are denoted by \( x_k \).

### 5.2. 2/3 Rule Map Field Matching

Each node of \( F_{ab} \) can receive input from three sources: \( F_2 \), \( F_1 \), and a Map Field gain control \( G \). The \( F_{ab} \) output vector \( \mathbf{x} \) obeys the 2/3 Rule of ART 1; namely,

\[
x_k = \begin{cases} 
1 & \text{if } y_j^k + G + \sum_{j=1}^{N_a} y_j^k w_{jk} > 1 + \overline{w} \\
0 & \text{otherwise}
\end{cases}
\]

where term \( y_j^k \) is the \( F_2 \) output signal, term \( G \) is a binary gain control signal, term \( \sum_j y_j^k w_{jk} \) is the sum of \( F_2 \rightarrow F_{ab} \) signals \( y_j^k \) via pathways with adaptive weights \( w_{jk} \), and \( \overline{w} \) is a constant such that

\[
0 < \overline{w} < 1.
\]

Values of the gain control signal \( G \) and the \( F_2 \rightarrow F_{ab} \) weight vectors \( w_j = (w_{j1}, \ldots, w_{jN_b}) \), \( j = 1 \ldots N_a \), are specified below.

### 5.3. \( F_2 \rightarrow F_{ab} \) Gain Control

Comparison of eqns (1) and (28) indicates an analogy between fields \( F_2 \), \( F_{ab} \), and \( F_1 \) in a Map Field module and fields \( F_0 \), \( F_1 \), and \( F_2 \), respectively, in an ART 1 module. Differences between these modules include the bidirectional nonadaptive connections between \( F_2 \) and \( F_{ab} \) in the Map Field module (Figure 6) compared to the bidirectional adaptive connections between fields \( F_1 \) and \( F_2 \) in the ART 1 module (Figure 5). These different connectivity schemes require different rules for the gain control signals \( G \) and \( g \).

The Map Field gain control signal \( G \) obeys the equation

\[
G = \begin{cases} 
0 & \text{if } F_2 \text{ and } F_1 \text{ are both active} \\
1 & \text{otherwise}
\end{cases}
\]

Note that \( G \) is a persistently active, or tonic, signal that is turned off only when both ART\(_n\) and ART\(_n\) are active.

### 5.4. \( F_1 \rightarrow F_{ab} \) Initial Values

If an active \( F_1 \) node \( J \) has not yet learned a prediction, the ARTMAP system is designed so that \( J \) can learn to predict any ART\(_b\) pattern if one is active or becomes active while \( J \) is active. This design constraint is satisfied using the assumption, analogous to eqn (13), that

\[
w_{jk}(0) = 1
\]

for \( j = 1 \ldots N_a \) and \( k = 1 \ldots N_b \).

### 5.5. Map Field Activation

Rules governing \( G \) and \( w_{jk}(0) \) enable the following Map Field properties to obtain. If both ART\(_n\) and ART\(_b\) are active, then learning of ART\(_n\) \( \rightarrow \) ART\(_b\) associations can take place at \( F_{ab} \). If ART\(_n\) is active but ART\(_b\) is not, then any previously learned ART\(_n\) \( \rightarrow \) ART\(_b\) prediction is read out at \( F_{ab} \). If ART\(_n\) is active but ART\(_b\) is not, then the selected ART\(_b\) category is represented at \( F_{ab} \). If neither ART\(_n\) nor ART\(_b\) is active, then \( F_{ab} \) is not active. By eqns (28)–(31), the 2/3 Rule realizes these properties in the following four cases.

#### \( F_2 \) active and \( F_1 \) active

If both the \( F_2 \) category node \( J \) and the \( F_1 \) category node \( K \) are active, then \( G = 0 \) by eqn (30). Thus by eqn (28),

\[
x_k = \begin{cases} 
1 & \text{if } k = K \text{ and } w_{jk} > \overline{w} \\
0 & \text{otherwise}
\end{cases}
\]

All \( x_k = 0 \) for \( k \neq K \). Moreover, \( x_K = 1 \) only if an association has previously been learned in the pathway from node \( J \) to node \( K \), or if \( J \) has not yet learned to predict any ART\(_b\) category. If \( J \) predicts any category other than \( K \), then all \( x_k = 0 \).
If the $F^a_1$ node $J$ is active and $F^b_2$ is inactive, then $G = 1$. Thus

$$x_a = \begin{cases} 1 & \text{if } w_{jk} > \bar{w} \\ 0 & \text{otherwise.} \end{cases} \quad (33)$$

By eqns (31) and (33), if an input $a$ has activated node $J$ in $F^a_1$ but $F^b_2$ is not yet active, $J$ activates all nodes $k$ in $F^{ab}$ if $J$ has learned no predictions. If prior learning has occurred, all nodes $k$ are activated whose adaptive weights $w_{jk}$ are still large.

If the $F^b_2$ node $K$ is active and $F^b_2$ is inactive, then $G = 1$. Thus

$$x_b = \begin{cases} 1 & \text{if } k = K \\ 0 & \text{otherwise.} \end{cases} \quad (34)$$

In this case, the $F^{ab}$ output vector $x$ is the same as the $F^b_2$ output vector $y^b$.

If neither $F^a_1$ nor $F^b_2$ is active, the total input to each $F^{ab}$ node is $G = 1$, so all $x_k = 0$ by eqn (28).

5.6. $F^a_1 \rightarrow F^b_2$ Choice and Priming

If ART$_a$ receives an input $b$ while ART$_a$ has no input, then $F^a_1$ chooses the node $K$ with the largest $F^a_1 \rightarrow F^b_2$ input. Field $F^b_2$ then activates the $K$th $F^{ab}$ node, and $F^{ab} \rightarrow F^b_2$ feedback signals support the original $F^a_1 \rightarrow F^b_2$ choice. If ART$_a$ receives an input $a$ while ART$_a$ has no input, $F^b_2$ chooses a node $J$. If, due to prior learning, some $w_{jk}$ = 1 while all other $w_{jk} = 0$, we say that $a$ predicts the ART$_b$ category $K$, as $F^{ab}$ sends its signal vector $x$ to $F^b_2$. Field $F^b_2$ is hereby attentionally primed, or sensitized, but the field remains inactive so long as ART$_b$ has no input. If then an $F^a_1 \rightarrow F^b_2$ input $b$ arrives, $F^b_2$ process $x$ as soon as $F^b_2$ processes the input $b$, and $F^b_2$ chooses the primed node $K$. Field $F^b_2$ then receives $F^a_1 \rightarrow F^b_2$ expectation input $z_k^b$ as well as $F^b_2 \rightarrow F^b_2$ input $b$, leading either to match or reset.

5.7. $F^a_1 \rightarrow F^{ab}$ Learning Laws

The $F^a_1 \rightarrow F^{ab}$ adaptive weights $w_{jk}$ obey an outstar learning law similar to that governing the $F_2 \rightarrow F_1$ weights $z_{jk}$ in (12); namely,

$$\frac{d}{dt} w_{jk} = \gamma_f (x_k - w_{jk}). \quad (35)$$

According to (35), the $F^a_1 \rightarrow F^{ab}$ weight vector $w_j$ approaches the $F^{ab}$ activity vector $x$ if the $J$th $F^a_1$ node is active. Otherwise $w_j$ remains constant. If node $J$ has not yet learned to make a prediction, all weights $w_{jk}$ equal 1, by eqn (31). In this case, if ART$_a$ receives no input $b$, then all $x_k$ values equal 1 by eqn (33). Thus, by eqn (35), all $w_{jk}$ values remain equal to 1. As a result, category choices in $F^a_1$ do not alter the adaptive weights $w_{jk}$ until these choices are associated with category choices in $F^b_2$.

5.8. Map Field Reset and Match Tracking

The Map Field provides the control that allows the ARTMAP system to establish different categories for very similar ART$_a$ inputs that make different predictions, while also allowing very different ART$_a$ inputs to form categories that make the same prediction. In particular, the Map Field orienting subsystem becomes active only when ART$_b$ makes a prediction that is incompatible with the actual ART$_a$ input. This mismatch event activates the control strategy, called match tracking, that modulates the ART$_b$ vigilance parameter $p_b$ in such a way as to keep the system from making repeated errors. As illustrated in Figure 6, a mismatch at $F^{ab}$ while $F^b_2$ is active triggers an inter-ART reset signal $R$ to the ART$_a$ orienting subsystem. This occurs whenever

$$|x| < \rho |y|^a. \quad (36)$$

where $\rho$ denotes the Map Field vigilance parameter.

The entire cycle of $\rho_a$ adjustment proceeds as follows through time. At the start of each input presentation, $\rho_a$ equals a fixed baseline vigilance $\overline{\rho}$. When an input $a$ activates an $F^a_1$ category node $J$ and resonance is established,

$$|x| = |a \cap z_j| \geq \rho_a |a|, \quad (37)$$

as in eqn (25). An inter-ART reset signal is sent to ART$_a$ if the ART$_b$ category predicted by $a$ fails to match the active ART$_b$ category, by eqn (36). The inter-ART reset signal $R$ raises $\rho_a$ to a value that is just high enough to cause eqn (37) to fail, so that

$$\rho_a > \frac{|a \cap z_j|}{|a|}. \quad (38)$$

Node $J$ is therefore reset and an ART$_a$ search ensues. Match tracking continues until an active ART$_a$ category satisfies both the ART$_a$ matching criterion eqn (37) and the analogous Map Field matching criterion. Match tracking increases the ART$_b$ vigilance by the minimum amount needed to abort an incorrect ART$_a \rightarrow$ ART$_b$ prediction and to drive a search for a new ART$_a$ category that can establish a correct prediction. As shown by example below, match tracking allows $a$ to make a correct prediction on subsequent trials, without repeating the initial sequence of errors. Match tracking hereby jointly maximizes predictive generalization and minimizes
predictive error on a trial-by-trial basis, using only local computations.

5.9. Match Tracking Using VITE Dynamics

The operation of match tracking can be implemented in several different ways. One way is to use a variation on the Vector Integration to Endpoint, or VITE, circuit (Bullock & Grossberg, 1988) as follows. Let an ART\textsubscript{a} binary reset signal \( r_a \) (Figure 7) obey the equation

\[
\begin{align*}
\hat{r}_a &= \begin{cases} 1 & \text{if } \rho_a|a| - |x^a| > 0 \\ 0 & \text{otherwise.} \end{cases} \\
\end{align*}
\]

(39)
as in eqn (23). The complementary ART\textsubscript{a} resonance signal \( r^*_a = 1 - r_a \). Signal \( R \) equals 1 during inter-ART reset; that is, when inequality (36) holds. The size of the ART\textsubscript{a} vigilance parameter \( \rho_a \) is determined by the match tracking equation

\[
\frac{d}{dt} \rho_a = (\bar{\rho}_a - \rho_a) + y R r_a.
\]

(40)

where \( y > 1 \). During inter-ART reset, \( R = r^*_a = 1 \), causing \( \rho_a \) to increase until \( r_a = 0 \). Then \( \rho_a|a| > |x^a| \), as required for match tracking (38). When \( r_a = 0 \), \( \rho_a \) relaxes to \( \bar{\rho}_a \). This is assumed to occur at a rate slower than node activation, also called short-term memory (STM), and faster than learning, also called long-term memory (LTM). Such an intermediate rate is called medium-term memory (MTM) (Carpenter & Grossberg, 1990).

Comparing the match tracking circuit in Figure 7 to a VITE circuit, the inter-ART reset signal \( R \) is analogous to the VITE GO signal; total \( F^a \) output \( |x^a| \) is analogous to the Target Position Code (TPC); total \( F^b \) output, gated by \( \rho_a \), is analogous to the Present Position Command (PPC); and the quantity \( (\rho_a|a| - |x^a|) \) in (39) is analogous to the Difference Vector (DV). (See Bullock & Grossberg, 1988, Figure 17.)

An ART\textsubscript{a} search that is triggered by increasing \( \rho_a \) according to eqn (40) ceases if some active \( F^b \) node \( J \) satisfies

\[
|a \cap z^J| \geq \rho_a|a|.
\]

(41)

If no such node exists, \( F^b \) shuts down for the rest of the input presentation. In particular, if \( a \subseteq z^J \), match tracking makes \( \rho_a > 1 \), so \( a \) cannot activate another category in order to learn the new prediction. The following anomalous case can thus arise. Suppose that \( a = z^J \) but the ART\textsubscript{a} input \( b \) mismatches the ART\textsubscript{b} expectation \( z^k \) previously associated with \( J \). Then match tracking will prevent the recoding that would have associated \( a \) with \( b \). That is, the ARTMAP system with fast learning and choice will not learn the prediction of an exemplar that exactly matches a learned prototype when the new predic-

![FIGURE 7. Match tracking by a scalar VITE circuit. When \( r_a = 1 \), \( \rho_a \) rapidly increases until \( \rho_a|a| > |x^a| \). Once this occurs, \( r_a = 0 \) and \( r_c = 1 \), causing ART, reset. The inter-ART reset signal \( R \) plays a role analogous to the VITE model GO signal.](image-url)
tion contradicts the previous predictions of the exemplars that created the prototype. This situation does not arise when all ART\textsubscript{a} inputs \( a \) have the same number of 1's, as follows.

### 5.10. Equal-Norm Inputs and Search

Consider the case in which all ART\textsubscript{a} inputs have the same norm:

\[
|a| = \text{constant. (42)}
\]

When an ART\textsubscript{a} category node \( J \) becomes committed to input \( a \), then \( |z_J| = |a| \). Thereafter, by the 2/3 Rule (15), \( z_J \) can be recoded only by decreasing its number of 1 entries, and thus its norm. Once this occurs, no input \( a \) can ever be a subset of \( z_J \), by eqn (42). In particular, the situation described in the previous section cannot arise.

In the simulations reported in this article, all ART\textsubscript{a} inputs have norm 22. Equation (42) can also be satisfied by using complement coding, since \( |(a, a^c)| = M_x \). Preprocessing ART\textsubscript{a} inputs by complement coding thus ensures that the system will avoid the case where some input \( a \) is a proper subset of the active ART\textsubscript{a} prototype \( z_J \) and the learned prediction of category \( J \) mismatches the correct ART\textsubscript{b} pattern.

Finally, note that with ARTMAP fast learning and choice, an ART\textsubscript{a} category node \( J \) is permanently committed to the first ART\textsubscript{b} category node \( K \) to which it is associated. However, the set of input exemplars that access either category may change through time, as in the banana example described in the introduction.

### 5.11. Match Tracking Example

The role of match tracking is illustrated by the following example. The input pairs shown in Table 7 are presented in order \((a^{(1)}, b^{(1)}), (a^{(2)}, b^{(2)}), (a^{(3)}, b^{(3)})\). The problem solved by match tracking is created by vector \( a^{(2)} \) lying "between" \( a^{(1)} \) and \( a^{(3)} \), with \( a^{(1)} \subset a^{(2)} \subset a^{(3)} \), while \( a^{(1)} \) and \( a^{(3)} \) are mapped to the same ART\textsubscript{b} vector. Suppose that, instead of match tracking, the Map Field orienting subsystem merely activated the ART\textsubscript{a} reset system. Coding would then proceed as follows.

<p>| TABLE 7 |
|-------------------|-------------------|</p>
<table>
<thead>
<tr>
<th>Nested ART\textsubscript{a} Inputs and Their Associated ART\textsubscript{b} Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ART\textsubscript{a} inputs</td>
</tr>
<tr>
<td>( a^{(1)} ) (111000)</td>
</tr>
<tr>
<td>( a^{(2)} ) (111100)</td>
</tr>
<tr>
<td>( a^{(3)} ) (111110)</td>
</tr>
</tbody>
</table>

Choose \( \bar{\rho}_e = 0.6 \) and \( \rho_s > 0 \). Vectors \( a^{(1)} \) then \( b^{(4)} \) are presented, activate ART\textsubscript{a} and ART\textsubscript{b} categories \( J = 1 \) and \( K = 1 \), and the category \( J = 1 \) learns to predict category \( K = 1 \), thus associating \( a^{(1)} \) with \( b^{(4)} \). Next \( a^{(2)} \) then \( b^{(3)} \) are presented. Vector \( a^{(3)} \) first activates \( J = 1 \) without reset, since

\[
\frac{|a^{(2)} \cap z|}{|a^{(2)}|} = \frac{3}{4} \geq \rho_s = \bar{\rho}_e.
\]

However, node \( J = 1 \) predicts node \( K = 1 \). Since

\[
\frac{|b^{(3)} \cap z|}{|b^{(3)}|} = 0 < \rho_s,
\]

ART\textsubscript{b} search leads to activation of a different \( F_2^b \) node, \( K = 2 \). Because of the conflict between the prediction \((K = 1)\) made by the active \( F_2^b \) node and the currently active \( F_2^b \) node \((K = 2)\), the Map Field orienting subsystem resets \( F_2^b \), but without match tracking. Thereafter a new \( F_2^b \) node \((J = 2)\) learns to predict the correct \( F_2^b \) node \((K = 2)\), associating \( a^{(2)} \) with \( b^{(3)} \).

Vector \( a^{(3)} \) first activates \( J = 2 \) without ART\textsubscript{a} reset, thus predicting \( K = 2 \), with \( z_2 = b^{(3)} \). However, \( b^{(3)} \) mismatches \( z_2 \), leading to activation of the \( F_2^b \) node \( K = 1 \), since \( b^{(3)} = b^{(1)} \). Since the predicted node \((K = 2)\) then differs from the active node \((K = 1)\), the Map Field orienting subsystem again resets \( F_2^b \). At this point, still without match tracking, the \( F_2^b \) node \( J = 1 \) would become active, without subsequent ART\textsubscript{a} reset, since \( z_1 = a^{(1)} \) and

\[
\frac{|a^{(3)} \cap a^{(1)}|}{|a^{(3)}|} = \frac{3}{5} \geq \rho_s = \bar{\rho}_e.
\]

Since node \( J = 1 \) correctly predicts the active node \( K = 1 \), no further reset or new learning would occur. On subsequent prediction trials, vector \( a^{(3)} \) would once again activate \( J = 2 \) and then \( K = 2 \). When vector \( b^{(3)} \) is not presented, on a test trial, vector \( a^{(3)} \) would not have learned its correct prediction.

With match tracking, when \( a^{(3)} \) is presented, the Map Field orienting subsystem causes \( \rho_s \) to increase to a value slightly greater than \( |a^{(3)} \cap a^{(2)}||a^{(3)}|^{-1} = 0.8 \) while node \( J = 2 \) is active. Thus after node \( J = 2 \) is reset, node \( J = 1 \) will also be reset because

\[
\frac{|a^{(3)} \cap a^{(4)}|}{|a^{(3)}|} = 0.6 < 0.8 < \rho_s.
\]

The reset of node \( J = 1 \) permits \( a^{(3)} \) to choose an uncommitted \( F_2^b \) node \((J = 3)\) that is then associated with the active \( F_2^b \) node \((K = 1)\). Thereafter each ART\textsubscript{a} input predicts the correct ART\textsubscript{b} output without search or error.

### 5.12. Complement Coding Example

The utility of ART\textsubscript{a} complement coding is illustrated by the following example. Assume that the nested
input pairs in Table 7 are presented to an ARTMAP system in order \((a^{(3)}, b^{(3)}), (a^{(2)}, b^{(2)}), (a^{(1)}, b^{(1)})\), with match tracking but without complement coding. Choose \(\rho_e < 0.5\) and \(\rho_R > 0\).

Vectors \(a^{(3)}\) and \(b^{(3)}\) are presented and activate \(\text{ART}_a\) and \(\text{ART}_b\) categories \(J = 1\) and \(K = 1\). The system learns to predict \(b^{(3)}\) given \(a^{(3)}\) by associating the \(F_3^a\) node \(J = 1\) with the \(F_3^b\) node \(K = 1\). The system in order \((a^{(3)}, b^{(3)}), (a^{(2)}, b^{(2)}), (a^{(1)}, b^{(1)})\), with match tracking but without complement coding. Choose \(\rho_e < 0.5\) and \(\rho_R > 0\).

Vectors \(a^{(3)}\) and \(b^{(3)}\) are presented and activate \(\text{ART}_a\) and \(\text{ART}_b\) categories \(J = 1\) and \(K = 1\). The system learns to predict \(b^{(3)}\) given \(a^{(3)}\) by associating the \(F_3^a\) node \(J = 1\) with the \(F_3^b\) node \(K = 1\). Next \(a^{(2)}\) and \(b^{(2)}\) are presented. Vector \(a^{(2)}\) first activates \(J = 1\) without reset, since \(\rho_e > \rho_R\). However, node \(J = 1\) predicts node \(K = 1\). As in the previous example, after \(b^{(2)}\) is presented, the \(F_2^a\) node \(K = 2\) becomes active and leads to an inter-ART reset. Match tracking makes \(\rho_e > 1\), so \(F_2^a\) shuts down until the pair \((a^{(2)}, b^{(2)})\) shuts off. Pattern \(b^{(2)}\) is coded in \(\text{ART}_b\) as \(z_2^b\), but no learning occurs in the \(\text{ART}_a\) and \(\text{ART}_b\) modules.

The Appendix outlines ARTMAP system responses to various input situations, namely, combinations of: a without b, b without a, a then b, b then a, a making a prediction or making no prediction, and a’s prediction matching or mismatching b.

**REFERENCES**


APPENDIX

A1. Simulation Algorithms

A1.1. ART 1 algorithm

Fast-learn ART 1 with binary $F_0 \rightarrow F_1$ input vector $I$ and choice at $F_1$ can be simulated by following the rules below. Fields $F_1$ and $F_2$ have $M$ nodes and field $F_2$ has $N$ nodes.

**Initial values**

Initially all $F_2$ nodes are said to be uncommitted. Weights $Z_{ij}$ in $F_2 \rightarrow F_3$ paths initially satisfy

$$Z_{ij}(0) = a_i, \tag{A1}$$

where $Z = (Z_{11}, \ldots, Z_{mM})$ denotes the bottom-up $F_2 \rightarrow F_1$ weight vector. Parameters $a_i$ are ordered according to

$$a_1 > a_2 > \ldots > a_m, \tag{A2}$$

where

$$0 < a_i < \frac{1}{(\beta + |I|)} \tag{A3}$$

for $\beta > 0$ and for any admissible $F_1 \rightarrow F_2$ input $I$. In the simulations in this article, $a_i$ and $\beta$ are small.

Weights $z_i$ in $F_2 \rightarrow F_3$; paths initially satisfy

$$z_i(0) = 1. \tag{A4}$$

The top-down, $F_3 \rightarrow F_2$ weight vector $(z_{1j}, \ldots, z_{mj})$ is denoted $z_j$.

**F$_1$ activation**

The binary $F_1$ output vector $x = (x_1, \ldots, x_M)$ is given by

$$x = \begin{cases} 1 & \text{if } F_1 \text{ is inactive} \\
1 \cap Z_j & \text{if the } Jth F_2 \text{ node is active}. \tag{A5} \end{cases}$$

**$F_2 \rightarrow F_3$ input**

The input $T_j$ from $F_2$ to the $Jth F_3$ node obeys

$$T_j = \begin{cases} |I| & \text{if } j \text{ is an uncommitted node index} \\
|I \cap Z_j(\beta + |I|) & \text{if } j \text{ is a committed node index} \tag{A6} \end{cases}$$

The set of committed $F_2$ nodes and update rules for vectors $z$ and $Z$ are defined iteratively below.

**$F_2$ choice**

If $F_2$ is active ($|I| > 0$), the initial choice at $F_2$ is one node with index $J$ satisfying

$$T_j = \max(T_j). \tag{A7}$$

If more than one node is maximal, one of these is chosen at random. After an input presentation on which node $J$ is chosen, $J$ becomes committed. The $F_2$ output vector is denoted by $y = (y_1, \ldots, y_N)$.

**Search and resonance**

ART 1 search ends upon activation of an $F_2$ category with index $j = J$ that has the largest $T_j$ value and that also satisfies the inequality

$$|I \cap Z_j| \geq \rho |I| \tag{A8}$$

where $\rho$ is the ART 1 vigilance parameter. If such a node $J$ exists, that node remains active, or in resonance, for the remainder of the input presentation. If no node satisfies (A8), $F_2$ remains inactive after search, until $I$ shuts off.

**Fast learning**

At the end of an input presentation the $F_2 \rightarrow F_3$ weight vector $Z_j$ satisfies

$$Z_{ij} = |I \cap Z_j^{(a)}| \tag{A9}$$

where $z_i^{(a)}$ denotes $z_i$ at the start of the current input presentation.

The $F_2 \rightarrow F_3$ weight vector $Z$ satisfies

$$Z_j = \frac{|I \cap Z_j^{(a)}}{\beta + |I \cap Z_j^{(a)}|}. \tag{A10}$$

A1.2. ARTMAP Algorithm

The ARTMAP system incorporates two ART modules and an inter-ART module linked by the following rules.

**ART$_1$ and ART$_2$**

ART$_1$ and ART$_2$ are fast-learn ART 1 modules. Inputs to ART$_1$ may, optionally, be in the complement code form. Embedded in an ARTMAP system, these modules operate as outlined above, with the following additions. First, the ART$_1$ vigilance parameter $\rho$ can increase during inter-ART reset according to the match tracking rule. Second, the Map Field $F^0$ can prime ART$_1$. That is, if $F^0$ sends nonuniform input to $F^2$ in the absence of an $F^3 \rightarrow F^2$ input $b$, then $F^2$ remains inactive. However, as soon as an input $b$ arrives, $F^2$ chooses the node $K$ receiving the largest $F^3 \rightarrow F^2$ input. Node $K$, in turn, sends to $F^1$ the top-down input $x^K$. Rules for match tracking and complement coding are specified below.

Let $x^k = (x_{1k} \ldots x_{nk})$ denote the $F^3$ output vector; let $y^k = (y_{1k} \ldots y_{nk})$ denote the $F^2$ output vector; and let $y^k = (y_{1k} \ldots y_{nk})$ denote the $F^1$ output vector. The Map Field $F^0$ has $N$ nodes and binary output vector $x$. Vectors $x^1, y^1, x^2, y^2, x$, and $y$ are set to 0 between input presentations.

**Map Field learning**

Weights $w_{ij}$, where $j = 1 \ldots N$, and $k = 1 \ldots N_m$, in $F^1 \rightarrow F^2$ paths initially satisfy

$$w_{ij}(0) = 1. \tag{A11}$$

Each vector $(w_{ij}, \ldots, w_{ik})$ is denoted $w_j$. During resonance with the ART$_1$ category $J$ active, $w_j \rightarrow x$. In fast learning, once $J$ learns to predict the ART$_2$ category $K$, that association is permanent, i.e., $w_{jk} = 1$ for all times.

**Fast Field activation**

The $F^0$ output vector $x$ obeys

$$x_{ij} = \begin{cases} y_{ij} & \text{if } F^0 \text{ is active and } F^1 \text{ is active} \\
y_{ij} & \text{if } F^0 \text{ is active and } F^1 \text{ is inactive} \\
0 & \text{if } F^0 \text{ is inactive and } F^1 \text{ is inactive}. \tag{A12} \end{cases}$$

**Match tracking**

At the start of each input presentation the ART$_1$ vigilance parameter $\rho$ equals a baseline vigilance $\rho_0$. The Map Field vigilance parameter is $\rho$. If

$$|x| < \rho |y^k|, \tag{A13}$$

then $\rho$ is increased until it is slightly larger than $|a \cap z^k|.|a|^{-1}$. Then

$$|x| = |a \cap z^k| < \rho |a|. \tag{A14}$$

where $a$ is the current ART$_1$ input vector and $J$ is the index of the active $F^2$ node. When this occurs, ART$_1$ search leads either to activation of a new $F^1$ node $J$ with

$$|x| = |a \cap z^k| \geq \rho |a| \tag{A15}$$

and

$$|x| = |y^k | = |y^k\cap w_j| \geq \rho |y^k|; \tag{A16}$$

or, if no such node exists, to the shut-down of $F^2$ for the remainder of the input presentation.

**Complement coding**

This optional feature arranges ART$_1$ inputs as vectors $(a, a') = (a_1, \ldots, a_n, a'_1, \ldots, a'_n)$. Then

$$a' = 1 - a_i. \tag{A18}$$

Complement coding may be useful if the following set of circumstances could arise: an ART$_1$ input vector $a$ activates an $F^2$ node $J$ previously associated with an $F^3$ node $K$; the current ART$_1$ input $b$ mismatches $z^k_j$; and $a$ is a subset of $z^k_j$. These circumstances never arise if all $|a| = \text{constant}$. For the simulations in this article, $|a| = 22$. With complement coding, $(a, a') = M_i$. This coding can allow independence of $b$ and $a$ and could appear alone, or one before the other.
Input a could make a prediction based on prior learning or make no prediction. If a does make a prediction, that prediction may be confirmed or disconfirmed by b. The system follows the rules outlined in the previous section assuming, as in the simulations, that all $|a|$ constant and that complement coding is not used. For each case, changing weight vectors $z_J$, $z_K$, and $w_k$ are listed. Weight vectors $Z_J$ and $Z_K$ change accordingly, by (A10). All other weights remain constant.

Case 1: a only, no prediction. Input a activates a matching $F_1$ node $J$, possibly following ART, search. All $F_1 \rightarrow F_a$ weights $w_{Ja} = 1$, so all $x_J = 1$. ART remains inactive. With learning $z_J \rightarrow z_a^{\text{add}} \cap a$.

Case 2: a only, with prediction. Input a activates a matching $F_1$ node $J$. Weight $w_{Ja} = 1$ while all other $w_{Ja} = 0$, and $x = w_J$. $F_1$ is primed, but remains inactive. With learning, $z_J \rightarrow z_a^{\text{add}} \cap a$.

Case 3: b only. Input b activates a matching $F_2$ node $K$, possibly following ART, search. At the Map Field, $x = y^*$. ART remains inactive. With learning, $z_K \rightarrow z_a^{\text{add}} \cap b$.

Case 4: a then b, no prediction. Input a activates a matching $F_2$ node $J$. All $x_J$ become 1 and ART is inactive, as in Case 1. Input b then activates a matching $F_2$ node $K$, as in Case 3. At the Map Field $x \rightarrow y^*$; that is, $x_J = 1$ and other $x_i = 0$. With learning $z_J \rightarrow z_a^{\text{add}} \cap a$, $z_K \rightarrow z_b^{\text{add}} \cap b$, and $w_K = y^*$; i.e., $J$ learns to predict $K$.

Case 5: a then b, with prediction confirmed. Input a activates a matching $F_1$ node $J$, which in turn activates a single Map Field node $K$ and primes $F_2$, as in Case 2. When input b arrives, the $K$th $F_1$ node becomes active and the prediction is confirmed; that is,

$$|b \cap z_K| \geq \rho_K |b|.$$  \hspace{1cm} (A19)

Note that $K$ may not be the $F_1$ node b would have selected without the $F_1 \rightarrow F_2$ prime. With learning, $x_J \rightarrow z_a^{\text{add}} \cap a$ and $z_K \rightarrow z_b^{\text{add}} \cap b$.

Case 6: a then b, prediction not confirmed. Input a activates a matching $F_2$ node, which in turn activates a single Map Field node and primes $F_2$, as in Case 5. When input b arrives, (A19) fails, leading to reset of the $F_2$ node via ART, reset. A new $F_2$ node $K$ that matches $b$ becomes active. The mismatch between the $F_2 \rightarrow F_a$ weight vector and the new $F_2$ vector $y^*$ sends Map Field activity $x$ to $0$, by (A12), leading to Map Field reset, by (A13). By match tracking, $\rho_K$ grows until (A14) holds. This triggers an ART, search that will continue until, for an active $F_2$ node $J$, $w_{Ja} = 1$, and (A15) holds. If such an $F_2$ node does become active, learning will follow, setting $z_J \rightarrow z_a^{\text{add}} \cap a$ and $z_K \rightarrow z_b^{\text{add}} \cap b$.

Case 7: b then a, no prediction. Input b activates a matching $F_2$ node $K$, then $x = y^*$, as in Case 3. Input a then activates a matching $F_3$ node $J$ with all $w_{Ja} = 1$. At the Map Field, $x$ remains equal to $y^*$. With learning, $x_J \rightarrow z_J^{\text{add}} \cap a$, $w_J \rightarrow y^*$, and $z_J \rightarrow z_a^{\text{add}} \cap b$.

Case 8: b then a, with prediction confirmed. Input b activates a matching $F_2$ node $K$, then $x = y^*$, as in Case 7. Input a then activates a matching $F_3$ node $J$ with $w_{Ja} = 1$ and all other $w_{Ja} = 0$. With learning, $z_J \rightarrow z_J^{\text{add}} \cap a$ and $z_K \rightarrow z_b^{\text{add}} \cap b$.

Case 9: b then a, prediction not confirmed. Input b activates a matching $F_2$ node $K$, then $x = y^*$ and input a activates a matching $F_3$ node, as in Case 8. However (A16) fails and $x \rightarrow 0$, leading to a Map Field reset. Match tracking resets $\rho_K$ as in Case 6, ART, search leads to activation of an $F_3$ node $J$ that either predicts $K$ or makes no prediction, or $F_3$ shuts down. With learning, $x_J \rightarrow z_J^{\text{add}} \cap a$. If $J$ exists, $x_J \rightarrow z_J^{\text{add}} \cap a$, and if $J$ initially makes no prediction, $w_J \rightarrow y^*$; i.e., $J$ learns to predict $K$. 

G. A. Carpenter, S. Grossberg, and J. H. Reynolds