1994-08

A Collection of Art-Family Graphical Simulations

A COLLECTION OF ART-FAMILY GRAPHICAL SIMULATORS

David Pedini and Paolo Gaudiano

August 1994

Technical Report CAS/CNS–94–023

Boston University Center for Adaptive Systems and
Department of Cognitive and Neural Systems
111 Cummington Street
Boston, MA 02215

A collection of ART-family graphical simulators

David Pedini and Paolo Gaudiano
Boston University, Dept. of Cognitive and Neural Systems
111 Cummington Street, Boston, MA 02215

1 Introduction

The Adaptive Resonance Theory (ART) architecture, first proposed by (Grossberg, 1976b, 1976a), is a self-organizing neural network for stable pattern categorization in response to arbitrary input sequences. Since its original formulation, several versions of ART have been proposed, each designed to handle a particular task or input format. Recent ART architectures have been designed to work in a supervised fashion, offering a viable alternative to supervised neural networks such as backpropagation (Rumelhart, Hinton, & Williams, 1986). Perhaps the best-known variant of ART is ART2 (Carpenter & Grossberg, 1987b), an unsupervised neural network that handles analog inputs. We have developed a series of simulators for some of the ART-family neural architectures, namely, ART2 (Carpenter & Grossberg, 1987b), ART2-A (Carpenter, Grossberg, & Rosen, 1991b), Fuzzy ART (Carpenter, Grossberg, & Rosen, 1990), and Fuzzy ARTMAP (Carpenter, Grossberg, Markuzon, & Reynolds, 1992). This article briefly summarizes the history and functionality of ART and its variants, and then describes the software package, which is available in the public domain.

1.1 Background of Adaptive Resonance Theory

ART was initially developed by Grossberg in an attempt to understand how biological learning systems are capable of retaining plasticity throughout life, without compromising the stability of previously learned patterns. Somehow biological learning mechanisms must be able to buffer stored memories against transient changes, while retaining plasticity to learn novel events in the environment. This tradeoff between continued learning and buffering of old memories has been called the *stability-plasticity dilemma* (Grossberg, 1976a).

Consideration of the stability-plasticity dilemma led Grossberg (1976a) to develop ART. In an earlier article, Grossberg (1976b) had shown that neural networks that learn through competitive learning are unable to satisfy the stability-plasticity dilemma. In fact, most present-day neural network models require the existence of distinct training and performance phases. The price of imposing a limit on the training phase is that these networks cannot learn any patterns that are presented after the initial training phase. In order to solve this problem, somehow there must be mechanisms that tell neurons when they should learn new things, and when they should preserve existing memories.

Grossberg (1976a) proposed that while individual neurons cannot by themselves decide when to learn and when not to learn, it is possible to design a network of neurons that autonomously recognizes novel events that should be learned, while preventing recoding of previously stored memories. Specifically, Grossberg has shown that ART's network-level interactions enable neurons to resolve the stability-plasticity dilemma.

Figure 1**(a)** is a highly schematized description of ART, which includes a population of neurons receiving sensory inputs (labeled F1), a second population of neurons that organize the inputs into learned categories or prototypes (labeled F2), and an orienting subsystem (dashed lines). Incoming sensory signals give rise to a spatial distribution of activity over the input population F1. The F1 activity pattern is transmitted to the F2 layer via modifiable synaptic connections.

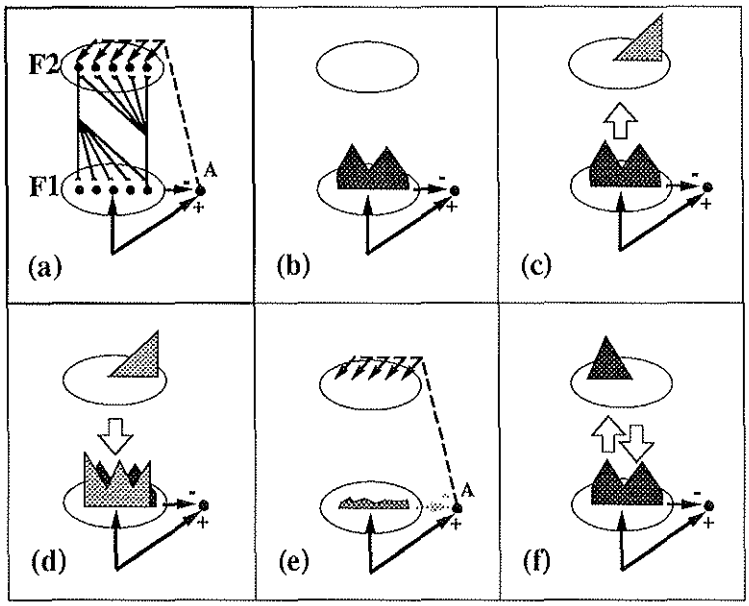

Figure 1: (a): Schematic representation of the ART architecture described in the text. Connections ending in semicircles represent modifiable synapses. Only some of the F1↔F2 connections are shown. (b)-(f): Diagram of the ART search cycle described in the text.

Competition between neurons in the F2 layer gives rise to a compact representation of the input pattern. This activation at F2 is referred to as a *category* because the same pattern of F2 activity can be elicited by a variety of similar F1 patterns. The active F2 neurons in turn project back to the F1 layer, also via a set of modifiable synapses. In so doing, the active F2 nodes learn the F1 pattern that activated them.

Suppose now that a new input pattern at F1 (dark shaded shape in Fig. 1**(b)**) erroneously activates an F2 category that was initially activated by a different input pattern (Fig. 1**(c)**). If learning were allowed to take place, the new input would wipe out the memory of the old input by changing the original learned meaning of the F2 category. Instead, the active F2 nodes immediately send a top-down signal to the F1 layer (Fig. 1**(d)**). The top-down signal is a copy of the pattern that the active F2 nodes "expect" to see at the F1 layer, and is thus referred to as *feedback expectation*. Because the input signals and feedback expectation differ, activation at the F1 population is temporarily suppressed. Temporary suppression of F1 activity disinhibits the orienting subsystem, which broadcasts an arousal signal (A) to all F2 nodes, indicating that a mismatch has occurred (Fig. 1**(e)**). The arousal signal is such that those F2 nodes that were most active (and thus responsible for the coding error) are suppressed, giving other F2 nodes (categories) the opportunity to become active. This process is a form of *hypothesis testing*, whereby the initial failure to categorize an input causes subsequent activation of different categories, or hypotheses about the meaning of the input. The process continues until a suitable category is activated, or else a new, uncommitted category is formed. The name *adaptive resonance* refers to the fact that learning only occurs when a good match causes a pattern to resonate between the F1 and F2 layers (Fig. 1**(f)**).

Several aspects make ART an attractive system for categorization problems. For one thing, if all categories in F2 have been committed to prior inputs, a new different input will not be able to be categorized, thus preserving older memories. In practical implementations, new nodes are added to F2 as needed, so that new inputs can always be coded with new categories. Another interesting

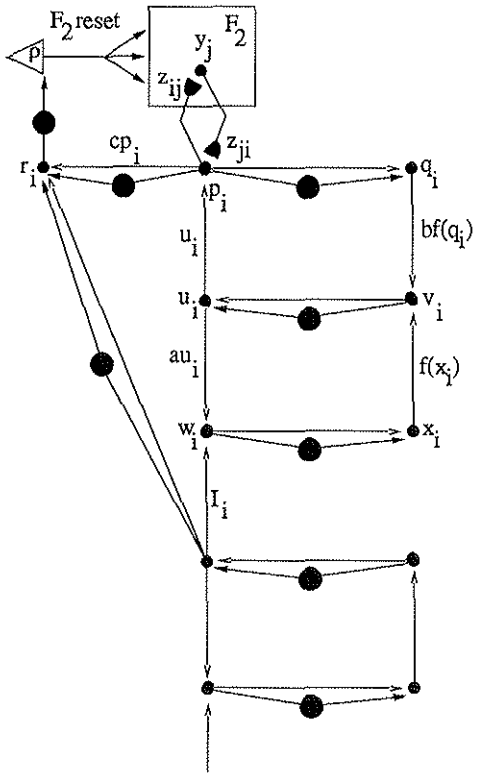

Figure 2: Diagram of one variant of the ART2 architecture proposed by Carpenter and Grossberg (1987b), adapted by permission. In this version of ART2, which is the one used in the simulator, the input is first normalized in the F0 layer (bottom loop), then transmitted to the F1 layer, which includes two normalization loops. Also, pattern match is calculated between the output of the F1 layer and the F0 layer.

property of ART is its use of a *vigilance level*, which determines how similar the top-down and bottom-up patterns must be in order for resonance to occur: a high vigilance level means that patterns must be very similar in order to be coded by the same category; different vigilance levels can be used to obtain coarser or finer categorization of inputs. A third important property of ART is that the time it takes to access a stored category (or to determine that a new category is chosen) remains the same regardless of the number of learned patterns. In fact, under some general conditions it can be shown that the first categorization is the best possible match among previously coded categories, so that a reset caused by mismatch of this initial "hypothesis" will automatically lead to creation of a new category, as long as uncommitted F2 nodes are still available. These and other properties have made ART a useful building block for various applications. Several variants of the ART architecture, designed to handle specific classes of applications, are described in the next section.

1.2 Variations on the ART architecture

Carpenter and Grossberg (1987a) introduced ART1, an implementation of the ART architecture with binary input vectors, and proved a series of mathematical properties, including stability. The ART2 architecture (Carpenter & Grossberg, 1987b), shown in Fig. 2, extends ART to allow analog input vectors. Stability can no longer be proven, but in practice the system can be shown to be stable under most conditions. (Carpenter et al., 1991b) implemented a fast, algorithmic version of ART2, which they termed ART2-A. The ART2-A algorithm is equivalent to running ART in *fast*

learning mode, which means that all weights are allowed to reach equilibrium in response to each input pattern. The ART2-A algorithm offers a significant speed-up compared to the regular ART2 network, and comparable performance. Another version of ART, known as Fuzzy ART (Carpenter et al., 1990), utilizes fuzzy logic operators, and is able to handle binary or analog inputs.

Many of today's neural network applications utilize supervised networks. The use of supervised networks may or may not be biologically plausible, but it is sometimes the best solution for applied problem-solving. For instance, suppose one wants to categorize some inputs into specific classes. This can be done with a supervised network, in which case each input is paired with the desired category, as specified by the user. On the other hand, an unsupervised network such as ART may be able to form categories that more accurately reflect the nature of the input patterns, and it can handle continuous inputs without the need for a limited training phase. In order to take advantage of the advantages of both supervised and unsupervised learning, (Carpenter, Grossberg, & Reynolds, 1991a) developed ARTMAP, a neural network that combines two ART1 modules for supervised learning: one ART1 module is presented with the input, while the other ART1 module is given the desired output. The first ART1 module independently organizes its inputs into categories, but each category is then associated with the activation of the second ART1 module (the desired output). This scheme combines the advantages of supervised and unsupervised learning. Recent research efforts have shown that ARTMAP can achieve high performance level on several benchmarks, while being computationally very efficient. In order to allow supervised categorization of analog inputs, each ART1 module can be replaced by a Fuzzy ART module. Such an architecture has been presented by (Carpenter et al., 1992), and is known as Fuzzy ARTMAP.

2 Simulation software

The best way to learn how to use ART is probably to read some of the original articles and to write a program to implement the architecture. However, some readers have found the articles challenging for a variety of reasons: there are many articles on ART, and it is sometimes difficult to figure out where to start; the articles are typically not geared toward simple implementations and applications, but instead focus on theoretical properties of the architectures, and on their properties as models of biological and psychological processes.

A few years ago one of us circulated a simple C implementation for ART2, complete with a UNIX-style manual page, and sample input files. This simple package, which has a very crude text-driven menu and no graphical interface, has been circulated on most public domain sites related to neural networks, and the author still receives a steady stream of requests for updates, bug reports, etc. The popularity of that simple program led us to develop a set of graphical ART-family simulators that could be used by people interested in learning about ART. To this end, we have created a set of simulators for ART2, ART2-A, Fuzzy ART, and FuzzyARTMAP.

The graphical user interface (GUI) was designed to provide the user with a flexible, informative tool for analyzing all aspects of the ART architectures. Figure 3 illustrates the layout of the ART2 simulator. The GUI consists of several panels, described below.

2.1 The Main Menu

The Main Menu (upper left corner of Fig. 3) provides standard functions such as help, loading and saving ART2 objects, or changing global settings for the program.

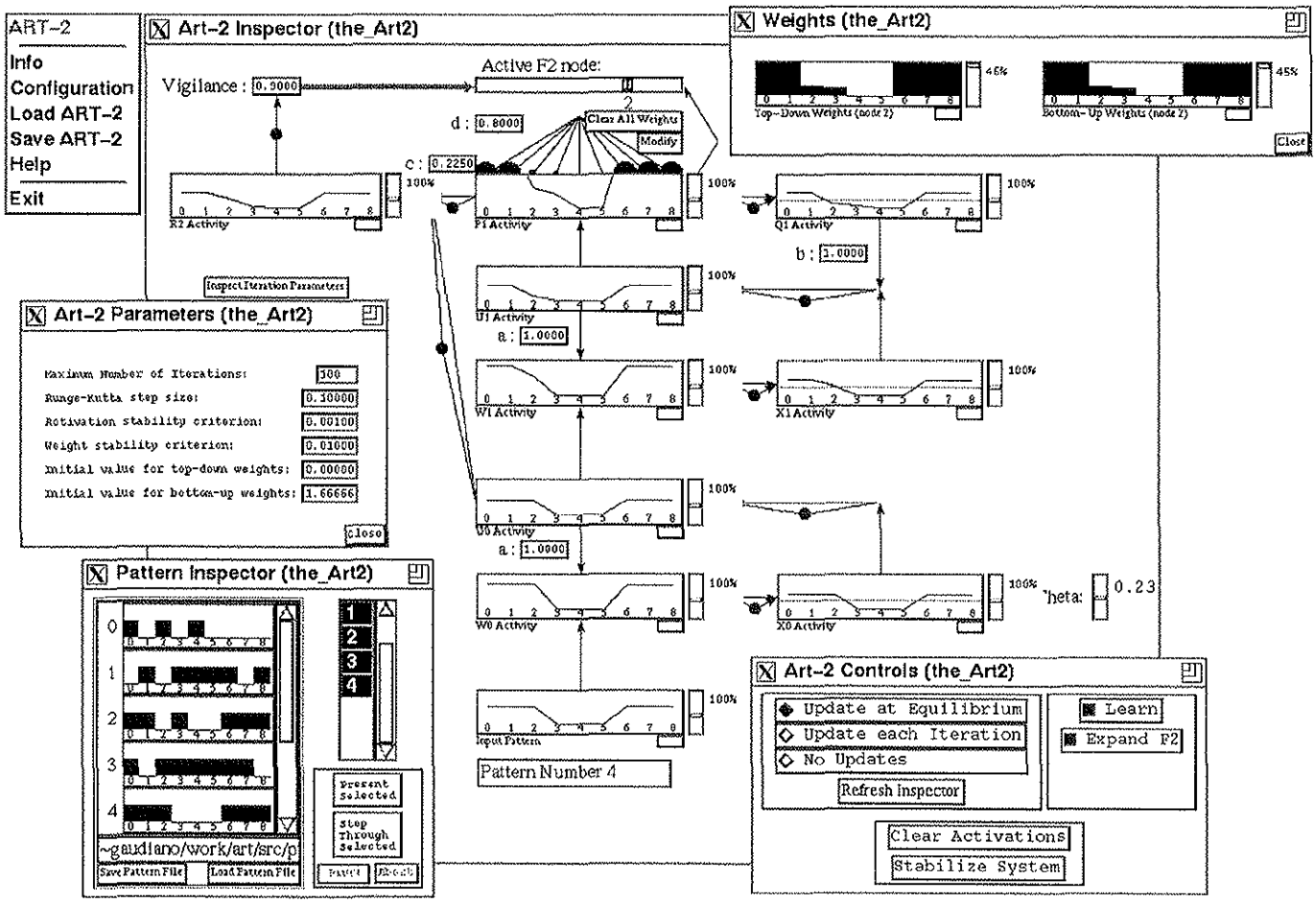

Figure 3: The graphical user interface for the ART2 program. The Main Menu in the upper left corner invokes additional panels for standard operations. The ART2 Inspector panel displays activation level of all populations in a typical ART2 architecture. Populations are organized in a position that mimics the typical ART2 layout, as proposed by Carpenter and Grossberg (1987b). The Pattern Inspector panel in the lower left corner allows loading and manipulation of input patterns. The Control panel in the lower right corner allows overall control of simulations and graphical interface. The Weights panel in the upper right corner (normally not visible) can be opened to allow inspection and modification of bottom-up and top-down LTM weighs. The Parameters panel (middle left) allows control of integration and other run-time parameters.

2.2 The ART2 Inspector

The main panel contains small graphic displays of the activation of the various neural populations, arranged in a structure similar to the ART2 structure presented by Carpenter and Grossberg (1987b) and reproduced in Fig. 2 above. The user can actually modify individual activations with a mouse click on the activation graph. In addition, each population includes (1) a scale slider that rescales the graph when activity patterns become too small or too large; and (2) a small button to zero the activity of all nodes for that population. Aside from the population graphs, the main panel allows the user to select a few of the key run-time parameters, such as the strength of certain connections, the thresholds within each ART2 layer, and the vigilance parameter.

2.3 The Pattern Inspector

The Pattern Inspector panel (lower left corner of Fig. 3) allows the user to load, manipulate, and save pattern files. In addition, the user can utilize a combination of keystrokes and mouse clicks to select patterns that will be presented to the ART2 network. Once selected, patterns can be presented one at a time or in a continuous sequence.

2.4 The ART2 Controls Panel

The Controls panel (lower right corner of Fig. 3) allows the user to control certain aspects of the global behavior of the simulator. For instance, learning can be turned on or off, the entire system can be cleared or brought to equilibrium, and the refresh frequency of the main panel can be controlled.

2.5 The Parameters Panel

The Parameters panel (middle left of Fig. 3) is activated by pressing the button labeled "Inspect Iteration Parameters", located on the left side of the Main panel. The Parameters panel allows the user to set internal parameters such as the integration step size, the maximum number of iterations before equilibration, initial values for weights, and stability criteria.

2.6 The Weights Panel

The Weights panel (upper right corner of Fig. 3) is activated by clicking the "Modify" button near the top of the main panel. The top-down weights are normally shown as semicircular synapses connecting the F2 population to the P1 population. However, the user must activate the Weights panel in order to view both sets of weights, and to modify them.

The ART2 simulator includes other features that are intended to make the software easy to use. The other ART-family simulators (ART2-A, FuzzyART, FuzzyARTMAP) are designed in the same way as the ART2 simulator described here.

3 Technical Notes and Availability

The ART2 engine is written in C++ and the interface is written in C++ and Tcl/Tk to run under the X window system. This program has been compiled using the GNU C compiler (gcc) version 2.4.0 and libg++ version 2.3.1. It also requires at least Tcl version 7.3 and Tk version 3.6. Current

and past versions of gcc and libg++ may be found at prep.ai.mit.edu, and versions of Tcl/Tk may be found at sprite.berkeley.edu.

The ART2 engine has been written to be independent from the graphical interface. An object library suitable for linking into other applications can be created when compiling the software. An example program is provided with the distribution to illustrate how this library can be used.

The simulators have run successfully on Sun workstations under SunOS 4.1.3 and newer SunOS versions, and on NeXT workstations under NEXTSTEP 3.0.

The software is available by anonymous ftp to *cns.bu.edu*, in the directory *pub/ART*. The file *ART.tar.gz* is a gzip'd tar file of the entire source distribution. Compiled binaries for SUN and SGI workstations are also available in the subdirectory *bin*. Aside form the source code, the full distribution includes UNIX-style man pages, and sample input files, including the input file used in the original ART2 article (Carpenter & Grossberg, 1987b).

References

Carpenter, G. A., & Grossberg, S. (1987a). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing, 37,* 54–115.

Carpenter, G. A., & Grossberg, S. (1987b). ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics, 26,* 4919–4930.

Carpenter, G. A., Grossberg, S., Markuzon, N., & Reynolds, J. (1992). Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks, 3,* 698–713.

Carpenter, G. A., Grossberg, S., & Reynolds, J. (1991a). ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks, 4,* 565–588.

Carpenter, G. A., Grossberg, S., & Rosen, D. B. (1991b). ART 2-A: An adaptive resonance algorithm for rapid category learning and recognition.. *Neural Networks, 4,* 493–504.

Carpenter, G. A., Grossberg, S., & Rosen, D. (1990). Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks, 4*(1), 759–771.

Grossberg, S. (1976a). Adaptive pattern classification and universal recoding II: Feedback, expectation, olfaction, and illusions.. *Biological Cybernetics, 23,* 187–202.

Grossberg, S. (1976b). Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors. *Biological Cybernetics, 23,* 121–134.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature, 323,* 533–536.