VICTORIA UNIVERSITY OF WELLINGTON

Department of Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 471 5328
Fax: +64 4 495 5232
Internet: Tech.Reports@comp.vuw.ac.nz

Visualisation of processes in ART neural networks

Martin Elliffe

Technical Report CS-TR-93/2
February 1993

Abstract

Artificial Neural Networks (ANNs) are of current research interest in many institutions worldwide [8] [12]. Adaptive Resonance Theory (ART) forms the basis of a family of like-titled ANNs, of particular current research interest at the Computer Science Department of Victoria University of Wellington [5] [4].

A problem common to many ANN paradigms is the difficulty users experience in understanding the processes by which the networks operate. Scientific visualisation techniques have been applied [11] [3] to produce visualisation tools which alleviate this problem.

This report discusses the inapplicability of current ANN visualisation techniques for use with ART. Seven new applicable techniques are presented.

The design of the irresistibly-titled SOTA (State Of The ART) as a single tool combining the new techniques is also presented, providing visualisation facilities to assist users in their understanding of the internal processes of ART ANNs.

Publishing Information

This report was originally prepared as part of the B.Sc.(Hons) Degree in Computer Science under the supervision of Guszti Bartfai.

Author Information

Martin Elliffe can be contacted through Guszti Bartfai (e-mail: guszti@comp.vuw.ac.nz).

Contents

1 Introduction **2**
- 1.1 Visualisation . . . 2
- 1.2 Neural Networks . . . 3
  - 1.2.1 Biological Neural Networks . . . 3
  - 1.2.2 Artificial Neural Networks . . . 3

2 ART **5**
- 2.1 Basic Concepts . . . 5
- 2.2 Network Topology . . . 5
- 2.3 Connection Weights . . . 6
- 2.4 Additional Constructs . . . 7
  - 2.4.1 The 2/3 Rule . . . 7
  - 2.4.2 Vigilance . . . 8
  - 2.4.3 The Reset Wave . . . 8
  - 2.4.4 Uncommitted Templates . . . 8
  - 2.4.5 Learning Speed . . . 9

3 ANN Visualisation Techniques **10**
- 3.1 Hinton diagrams . . . 10
- 3.2 Bond diagrams . . . 10
- 3.3 Hyperplane Diagrams . . . 10
- 3.4 Response-Function Plots . . . 11
- 3.5 Trajectory Diagrams . . . 11
- 3.6 Lascaux trees . . . 11
- 3.7 Summary . . . 12

4 Design Goals **13**
- 4.1 Network State . . . 13
- 4.2 Network State Modification . . . 14
- 4.3 Simulation/Visualisation Concurrency . . . 14

5 Visualisation Techniques **15**
- 5.1 Example ART Configuration . . . 15
- 5.2 N-feature Vectors . . . 16
  - 5.2.1 Choosing a Mapping . . . 16
  - 5.2.2 Template Dispersion Diagrams . . . 17
  - 5.2.3 Summary . . . 18
- 5.3 N-feature Sets . . . 19
  - 5.3.1 Quadrant Diagrams . . . 19
  - 5.3.2 Circle Diagrams . . . 21

5.3.3 Step Diagrams 22
5.3.4 Summary 23
5.4 Other Techniques 24
5.4.1 Vigilance Graphs 24
5.4.2 Input-Relative Diagrams 27

6 Design **30**
6.1 Architecture 30
6.1.1 Data Subsystem 30
6.1.2 Display Subsystem 31
6.1.3 Control Subsystem 32
6.2 Implementation Details 34

7 Future Directions **35**
7.1 Feature Ordering 35
7.2 Enhanced Template Analysis 35
7.3 Domain-Specific Representations 36
7.4 Vigilance Regions 36
7.5 SOTA Design Extensions 37
7.6 Extensions for ART-2 38

8 Conclusion **40**

A Algorithms **42**

Chapter 1

Introduction

Scientific visualisation techniques have been applied to the problem of representation of complex processes in many domains [10].

Artificial Neural Networks (ANNs) are typically complex dynamic systems, whose internal processes can be difficult to understand. Scientific visualisation techniques have been successfully applied to the ANN domain, resulting in the development of several representation techniques [3].

The central problem of ART representation is shown to be that of mapping high-dimensional binary (for the ART-1 ANN subclass) data to a lower-dimensional equivalent. This report discusses and rejects existing representation techniques for visualisation of these specialist data constructs. Seven new representation techniques which achieve appropriate mappings are presented.

The report also presents a high-level design for an interactive visualisation tool which incorporates all new representation techniques.

1.1 Visualisation

The term "scientific visualisation" encompasses a wide range of techniques based on the premise that it is easier to understand processes by examination of appropriate graphical images than numerical data alone [10].

The essential advantage of such an approach is that a single graphical object can display information pertaining to a combination of several quantitative data items at the same time. Visual attributes such as object size; shape; colour; orientation; velocity; and proximity to other objects (to name but a few), can each be associated with some attribute of the underlying data. Part of the appearance of an object can therefore change as the value of the associated data item changes. The "highly developed visual pattern-recognition abilities of humans" [3] can then assist in interpreting the actual behaviour of the underlying process.

The primary goal of a scientific visualisation system design is always to clarify understanding. An appropriate mapping between quantitative data items and visual attributes must be devised, based on considerable initial understanding of the underlying process. That is, for different processes, different aspects of the data available are important for different reasons. In one situation, focusing viewer attention on anomalies (or, conversely, regularities) of data may be important, whereas the relationship between (like) data items may be important in another. Displaying everything that happens may well be inappropriate in certain circumstances, just as watching operation at an intimate level is vital in others. Advanced understanding of the information available for display is required to select which items are relevant, just as restraint is required to avoid losing the message in the medium [9].

The anomaly of development of useful scientific visualisation systems is thus that in the areas where the techniques would be of most value (that is, where clarification is most needed), appropriate background understanding of the process is least available. Development is therefore necessarily evolutionary, design improvements being made as understanding increases.

1.2 Neural Networks

1.2.1 Biological Neural Networks

Neural research indicates that the brain consists of a very large number of highly interconnected neurons. While each neuron is incredibly complex at the molecular level, it is believed that much of the power of the brain as a whole is derived from the neurons' large-scale interaction. Assimilation of memories appear to be achieved by alteration of the strengths of connections between neurons, rather than change within the neurons themselves. Neurons work slowly by comparison with contemporary electronic devices, their responses in the order of milliseconds, suggesting that the relatively complex functions a brain can perform are achieved in a relatively small number of time steps.

Further, the brain allows memories to be addressable by content; the frequent death of individual neurons does not drastically affect performance of the brain as a whole; and humans can perform complex recognition functions very swiftly.

These observations combine to suggest that the brain performs computations in a vastly different manner from that used by contemporary electronic computers. A different model is required to discover more about how the brain works, let alone how to construct an artificial device able to even approximate some of its advanced abilities.

1.2.2 Artificial Neural Networks

ANNs are an attempt to provide computational models consistent with the observations of section 1.2.1 above.

Many different ANN paradigms have been developed over the last fifty years. All are based on an interconnected network of relatively simple computational nodes. Neither the topology of the network as a whole nor the (parametric) function performed by any node typically change during the lifetime of any network instance, the only variable being the strength of the connections between nodes (which determine how sensitive one node is to the output of another).

Thus, from some (usually random within some predefined range) starting state, any instance of a network "learns" its connection strengths in response to its experiences.

Each ANN paradigm can therefore be characterised by :

- the topologies supported;
- the function computed by each node; and
- the "learning" mechanism (ie the manner in which node interconnection strengths are modified to reflect experience).

Node Functionality

Each node in an ANN computes a function based on its inputs. Three steps are usually performed :

- input - the net input to every node is calculated by multiplying the input received on each connection by the strength of that connection, then summing all such products (a weighted sum);
- activation - the internal activation state of every node is calculated by some model-specific function of the net input; and
- output - the level of output from every node is calculated by a model-specific function of the activation state.

Learning Mechanisms

Many different learning schemes have been used, with many different functional expectations. One important distinction is the difference between supervised and unsupervised learning mechanisms.

Supervised learning requires an omniscient external entity to provide feedback regarding the accuracy of the network's output given some input. Supervised ANNs therefore adjust their weights to attempt to produce the desired result (minimise their inaccuracy) the next time that input is supplied. The most widely used example of a supervised ANN paradigm is back-propagation [8].

Unsupervised learning, on the other hand, requires no such external omniscience. Instead, the learning mechanism organises the ANN so that some paradigm-appropriate response is prompted by the input alone.

The functions performed by supervised and unsupervised ANNs are very different. The supervised variants use external feedback to try to steer their operation toward simulation of an arbitrary function, while unsupervised networks perform a specific function based only on the current input and their experience to date (for example, Kohonen networks, a common type of unsupervised ANN, arrange their nodes in a quantised version of the input domain's probability density function).

Chapter 2

ART

ART [2] [1] provides the basis for a class of self-organising unsupervised competitive learning ANNs. Considerable effort has been expended throughout the development of the theory to maintain its physiological plausibility, discussion of which is beyond the scope of this report. Very complex models may result from implementation of this theoretical construct, the focus of this report being the simplified binary version - ART-1. The term "ART" will hereinafter be used to refer to ART-1.

2.1 Basic Concepts

An ART network classifies input patterns of binary digits, each bit being thought of as indicating the presence (1) or absence (0) of a pattern feature. The network achieves this task by learning pattern templates against which the input is compared, this "closeness of fit" then compared against a user-defined level of vigilance. If the match complies within vigilance, the template may be updated to reflect the additional information now known about this class, and the network settles into a steady state known as "resonance". If the match is not close enough, either comparison against another template will be attempted, or the network will adapt by constructing an entirely new category with a template corresponding exactly to the current input.

2.2 Network Topology

The fundamental ART network architecture (see figure 2.1) consists of two fully interconnected layers of nodes, termed F1 and F2.

F1 contains N nodes, each of which receives input from one input connection. Each F1 node therefore represents one pattern feature. An "active" F1 node indicates the presence of the relevant feature, and produces an output of 1 to all connected nodes in F2.

F2 contains M nodes, each of which represent one pattern template. The pattern template any node represents is stored in the weights of connections from that node down to the nodes of layer F1. If the pattern template represented by the F2 node includes the pattern feature represented by an F1 node, the downward connection between these nodes will have a weight of 1. If not, the connection weight will be 0.

An important feature of F2 is that it is "competitive", in that nodes compete with each other to determine their resulting activity. While some variants of the ART model may allow a pattern of activity across the nodes of F2, this report discusses the more common case where a maximum of one node in the layer can be active at one time. The single active ("winning") F2 node is selected by having a larger net input than any other. The size of the

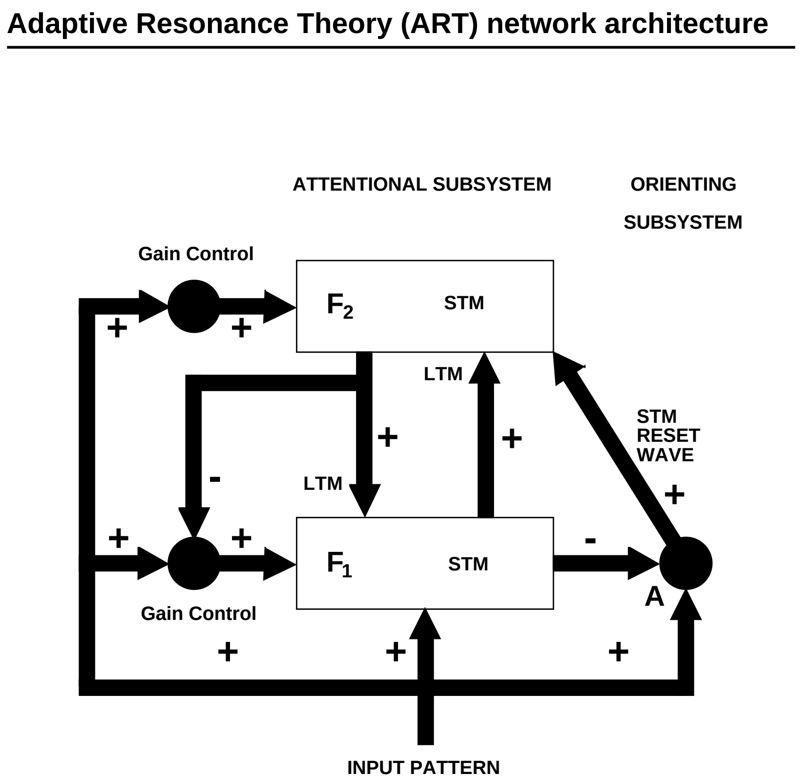

Figure 2.1: ART network architecture

net input to any F2 node is given by the sum of the bottom-up weights on all connections between this F2 node and the active nodes of F1 (ie its a weighted sum). Note that if two or more F2 nodes have the same net input, some arbitrary scheme of discrimination is used to select the winner.

The values of N (defining the maximum number of features any pattern can contain, and thus the number of nodes in F1) and M (defining the maximum number of templates the network can contain, and thus the number of nodes in F2) are specified by initial configuration parameters. The symbols N and M will be used throughout this report in this fashion.

2.3 Connection Weights

Each node in F2 represents exactly one template. "Top-down" (from F2 to F1) connection weights store the detail of the template itself, with each weight either 1 (to each F1 node representing a feature that is present in the template) or 0. The activation of a node in F2 results in presentation of the template across F1 ("attentional priming" in Carpenter and Grossberg [2] terminology).

Top-down weights are modified by the equation

$$T_{ji}^{new} = \begin{cases} C_i & \text{if node j in F2 wins} \\ T_{ji}^{old} & \text{otherwise} \end{cases} \tag{2.1}$$

where T_{ji} is the top-down weight from the (winning) jth node of F2 to the ith node of F1, and C_i is the activation of the ith node of F1 (1 if the feature is present, 0 if not).

Top-down weights from inactive (all but the winner) F2 nodes are unchanged, with those from the winning node staying at one if the F1 node to which they are connected is active and either becoming or staying zero if the connected F1 node is inactive. The top-down weight corresponding to the winning F2 node thus takes on F1s current activation pattern, being the intersection of the old template for this F2 node and the current input.

"Bottom-up" weights (from F1 to F2) are slightly more complex, being a normalised representation of the templates each F2 node represents. That is, all bottom-up connections to an F2 node are either 0 (if that feature is not present in the template) or are equal to each other (if it is). Their exact value is given by

$$B_{ij} = \frac{L\, C_i}{L - 1 + \Sigma_{i=1}^{N} C_i} \tag{2.2}$$

where B_{ij} is the weight of the connection from the ith node of F1 to the (winning) jth node of F2, L is a constant (usually 2), and thus $\Sigma_{i=1}^{N} C_i$ is the total number of features present in F1.

The bottom-up weights define the strength of the connection between nodes in F1 and F2, and thus contribute directly to the pre-competition activation levels achieved across F2 (given some input pattern presentation). The level of activation achieved will be directly proportional to the number of features that are common between the node's template (thus giving non-zero bottom-up weights) and the current input pattern (giving non-zero F1 node output values), since the two values for each connection are multiplied during the weighted sum net input calculation. Note, however, that by use of a value greater than 1 for L, the size of individual bottom-up weights increase as an F2 node's template becomes shorter. Thus for any two nodes in F2 which intersect with the input pattern in the same number of features, the F2 node whose template is shorter than the other will gain a higher pre-competition activation.

The overall effect is that the bottom-up weights provide a dynamic search ordering on the template set, such that the order is adapted not only to reflect changes in weight values (the detail of the templates held), but also with the input pattern currently presented.

Carpenter and Grossberg [2] describe the bottom-up weights collectively as an "adaptive filter" (since the weights change as the network adapts to incorporate new experiences, and the F2 net input calculation filters out weights on connections from irrelevant F1 nodes).

2.4 Additional Constructs

ART requires several additional constructs to allow implementation.

2.4.1 The 2/3 Rule

F1 uses an additional construct to produce an appropriate output for the signal traffic it receives. When an input pattern is present, the activation across the nodes of F1 must echo that pattern, but when an input pattern is present **and** an F2 node is active, F1 must hold the intersection of the input pattern and the template F2 represents.

This is achieved by use of the 2/3 rule, which states that any node in F1 will only become active when 2 (or more) of its 3 connections are also active. That is, in addition to the input connection and the top-down connections (remembering that only one top-down connection to an F1 node can be active at any one time), a third connection (F1 gain control) is also used, active across all F1 nodes whenever any input pattern is present. Thus F1 will echo any input pattern, since one of the required signals is available whenever any input pattern is present, and the second will be whenever the relevant feature is present in the pattern. Finally, F1 gain control is disabled whenever one of the F2 nodes is active, thereby allowing the activation of nodes in F1 to be determined by signals (filtered by the top-down weights from F2) and from the input when both are available. The intersection between a template and input pattern is thus calculated.

2.4.2 Vigilance

Vigilance is the mechanism by which a template and input pattern are compared for "closeness of fit". An initial configuration parameter, V, in the range of 0 through 1 inclusive, defines the network's level of vigilance.

When an F2 node has been selected as the winner, thereby activating nodes in the F1 layer corresponding to the intersection of the F2 node's template and the current input pattern, ART then divides the number of F1 nodes active by the number of features in the input to calculate a similarity metric. If this calculated value is less than V, a reset wave will be invoked. Otherwise, no reset will be invoked, any necessary adjustments to the winning F2 nodes top-down and bottom-up weights will be made, and resonance achieved.

Note that similarity calculation is based on how many features present in the input pattern are also present in the template. Any feature present in the template but not in the input pattern is irrelevant (only affecting likelihood of the corresponding F2 node becoming the winner, since bottom-up weights are inversely proportional to the number of features present in the template).

The term "vigilance region" is used throughout the rest of this report to refer to the region of input space each template covers (since the size of this region varies with the size of V).

2.4.3 The Reset Wave

The reset wave is invoked when the input pattern and the template represented by the winning F2 node fail to match closely enough to exceed vigilance.

The mechanism used is based on gated-dipole fields, such that any active node in F2 is temporarily disabled, thus allowing activation of the F2 node with the next highest net input.

The parts of an ART network used to implement the comparison of a calculated similarity metric against V and activate a reset wave are known as the "orienting subsystem", the balance of the network being the "attentional subsystem".

2.4.4 Uncommitted Templates

Uncommitted templates are represented by F2 nodes which have never been activated (ie never won). Before any input patterns have been presented, all templates are uncommitted.

From the discussion in section 2.3 above, it can be seen that a template can only become committed when the F2 node by which it is represented becomes active (ie its net input is higher than that of any other node in F2).

Further, once the F2 node of an uncommitted node becomes active, the template itself can only be modified by removing one or more of its component features (since top-down weights can only be changed from 1 to 0).

Thus, the top-down weights of all uncommitted templates must be 1, since otherwise input patterns could be presented which the network could not handle. Since all templates are initially uncommitted, all top-down weights must be initialised to 1.

The bottom-up weights, on the other hand, must be greater than 0 (so that the associated F2 node can at some stage become active) yet small enough so that they do not interfere with the operation of committed templates. That is, an F2 node representing an uncommitted template must never become active when a committed template exists such that activation of the committed templates representative F2 node would allow the network to achieve resonance.

Such a value for the bottom-up weights of F2 nodes representing uncommitted templates (and therefore the initialisation value of all bottom-up weights) is given by the range

$$0 < B_{ij}(initial) < \frac{L}{L - 1 + N} \tag{2.3}$$

Moreover, such initialisation can mean that an uncommitted template will precede some (or all) committed templates in the search order for an input pattern. A new template can therefore be committed without specifically looking at some (or any) of the stored templates, since the orientation-proximity function of the bottom-up weights has determined that an uncommitted template is closer to the input pattern than *any* committed template.

2.4.5 Learning Speed

The ART network operates as a continuous process. Activity occurs only when an input pattern is present, be that activity the calculation and possibly reconfiguration required to achieve resonance, or simply the resonant state itself.

ART learning equations therefore have two forms, being

- fast learning (as outlined in this document); and
- slow learning (where weights are adjusted toward the values achieved by fast learning, with their exact values being dependent on the length of time the input pattern has been presented).

This project assumes use of the fast learning variant. The implications of the assumption are somewhat irrelevant in a practical sense (since neither ART simulator available uses slow learning anyway), but a slightly restricted conceptual model of ART network operation is made permissible.

That is, under fast learning, the activity invoked by presentation of an input pattern can be considered to have terminated once resonance is achieved. No further activity will occur until a different input pattern is presented. The design of SOTA is based on this assumption of discrete learning steps rather than a continuous process.

Chapter 3

ANN Visualisation Techniques

A comprehensive survey of existing ANN visualisation techniques has been collated by Craven and Shavlik [3]. This chapter discusses the applicability of each technique to the visualisation of the ART class of ANN.

3.1 Hinton diagrams

Hinton diagrams [8] are used to display the state of a wide variety of ANN paradigms. They represent internodal connection weights as squares of size proportional to the weights absolute value, with colour representing the weights sign (black for positive, white for negative).

They are widely used because, at base, all ANNs include these properties, and thus can be validly displayed. They don't mislead, but they can often be difficult to interpret. In an ART context, for example, with a small input pattern of 25 features, Hinton diagrams of the top-down and bottom-up weights would consist of 625 squares each. While all the data related to network state would be available, the representation provided is at a lower level than required.

3.2 Bond diagrams

Bond diagrams [11] are an enhancement to Hinton diagrams, representing connection weight by the length (absolute value) and colour (sign) of a symmetric radial "bond" stretched between circular node representations (network topology is therefore implicitly represented). Further, nodes are surrounded by a circle of radius proportional to the node bias (or threshold) value.

ART nodes do not have bias values. Their topology and connection weight are of interest, but bond diagrams offer relatively little of value above a Hinton diagram in this respect.

3.3 Hyperplane Diagrams

Hyperplane diagrams combine two or three dimensions of a networks input space, drawing "division surfaces" to represent the way that activation states of various nodes vary with the value of the input dimensions.

ARTs use of high dimensional spaces makes this technique essentially useless. To avoid representation of duplicate data points, a diagram has to display node activations for varying input values of the specified dimensions in a hyperplane determined by constant values for all the other dimensions (thus an n-plane where n is one greater than the number of dimensions displayed).

While this could be achieved, its meaning is minimal, essentially being a long-winded cut-down version of a Hinton diagram.

3.4 Response-Function Plots

A similar idea to hyperplane diagrams above, response-function plots differ most obviously by allowing the display of states of a node with a sigmoidal (S-shaped) rather than thresholding (step) activation function.

ART nodes have a different kind of activation function, F1 nodes being subject to the 2/3 rule, and F2 nodes being competitive (the highest value wins, suppressing all others).

Response-function plots are inappropriate for use with ART networks.

3.5 Trajectory Diagrams

Trajectory diagrams show the path a node follows through its weight space as the network learns, with the colour of the node representing the error attributed to that node.

ART networks have no concept of error, trajectory diagrams being designed with back-propagation in mind. The idea of tracing development through a multi-dimensional space *is* used in SOTA, but is of a hybrid form.

Back-propagation networks have weight values which are typically changed often and by small amounts. ART networks, on the other hand, have weights which are typically changed few times, and irreversibly. That is, ART templates really consist of the intersection of all the inputs which have achieved resonance with the corresponding F2 node activated. Thus, once a top-down (and therefore bottom-up) weight becomes zero, it stays zero. Bottom-up weights change more slowly, gradually increasing as the size of the template decreases. Tracing such trajectories may still be useful, but better (non-duplicating) techniques have been devised.

Trajectory diagrams represent dimensions by radial projections - a mapping of points in high-dimensional space such that all the dimensional axes lie in one plane, radiating equiangularly from the origin. Thus, in spaces with more than two dimensions, diagram points may have non-unique mappings. The scale of this problem is reduced given the sparse population of this space by points available to ART, but the difficulty of interpreting radial projection diagrams in spaces with as few as 6 dimensions [3] prompt a desire to improve this method before real benefit can be gained.

3.6 Lascaux trees

Craven and Shavlik's technique [3], *Lascaux* Trees show how the presence or absence of various input features affect the activation of nodes further up a hierarchy.

Designed for back-propagation networks, inputs to any hidden or output node can be represented by lines whose width represents either the size of the weight of the connection (negative weights being drawn as dashed lines) or, according to a user option, the actual signal level passed (optionally embellished by a node activation function graph). The nodes themselves may then have a putative concept label, and display activation and error meters.

An input pattern representation can thus be given, thence showing its effect on the remainder of the network.

Again, this technique is back-propagation-specific, but does include some ideas which are of value. Attaching meaningful labels, providing these do not accidentally mislead (which may well, in my opinion, be the case in higher layers of a back-propagation network), would assist the user to understand the ART representation of the original input domain.

Further, the concept of "chunking" inputs which have some inherent relationship may be worth pursuing (see chapter 7), but is of a different form to that available under back propagation. In a multi-layered back-propagation network, it is possible that nodes in higher levels genuinely take on some conceptual meaning, their activation indicating that the input pattern included some appropriate distinguishing characteristic(s). (As I alluded in the previous paragraph, caution should always be used when interpreting the semantics of higher-layer nodes in back-propagation networks. A genuine understanding of their function is not always straightforward, and the function itself can be very different given different training runs.) In ART networks, no intermediate level nodes exist, but the idea still has merit as a kind of template content analysis tool. That is, it may well assist the user to understand more about the characteristics of the input domain if higher-level occurrence commonality can be detected and displayed (ie if two features occur together in all templates, they should probably be combined at the input level). Chunking is further discussed in section 7.2.

An organisational-chart (ie line-connected boxes) format representing the relationship between F2 nodes and input features may also be useful, though SOTA includes several other alternatives by which this information can be displayed.

3.7 Summary

None of the above techniques is of particular relevance to the visualisation of ART networks. Hinton diagrams do have some value, but the large number of diagrams that would need to be shown, and the difficulty of interpreting their content (the information they present is at a very low level) are genuine problems. Many of the techniques discussed were developed due to user dissatisfaction with Hinton diagrams as a mechanism for visualising the processes occurring in back-propagation networks. In much the same manner, customised techniques for ART networks are now required.

Chapter 4

Design Goals

The original motivation behind the design of SOTA was to provide a tool to assist a user to understand the processes performed by any ANN simulator, though of particular relevance to ART. Such a broad ambition rapidly proved to be beyond the scope of an honours-level project, and was thus scaled down to focus on ART alone. This chapter details the manner in which this underlying motivation was decomposed into specific design goals.

The first step was to split the motivation into three parts. The user should be able to :

- view the state of the network at any stage;
- view the process by which any change in the state of the network occurred; and
- view the behaviour of the network either concurrently with its simulation, or after the event.

The following sections translate these goals into ART-specific equivalents. The term "network" is used to refer to the simulated ART network of interest to the user.

4.1 Network State

The state of an ART network is defined by its initial configuration parameters and the current values in its top-down (template) and bottom-up weight matrices, and thus an unambiguous representation of these values is sought. Specifically, the representation should provide the user with :

- initial configuration parameter values (N; M; and V);
- some idea of template dispersion over the input space. This includes :
  - display of any input pattern;
  - display of any or all templates (and thus some idea of how templates relate to each other);
  - display of the region of input space covered by each template (ie the input patterns with which, if presented, each template would achieve resonance); and
  - display of the input space region covered by the template set as a whole
- a representation of how "full" the network is (what proportion of template nodes are still uncommitted).

4.2 Network State Modification

The process by which an ART network changes state is determined by the state of the network at some time instant and the input pattern about to be presented. The actual steps followed in changing that state are to determine a vigilance-satisfying winning F2 node and, if necessary, to then update the associated top-down and bottom-up weights.

To understand this process, SOTA must therefore provide the user with a representation of :

- the initial (pre-presentation) network state;
- the input pattern presented;
- the manner in which the vigilance-satisfying F2 node is chosen; and
- the resulting resonant network state.

4.3 Simulation/Visualisation Concurrency

In order to provide the SOTA user with the ability to view the behaviour of a network either concurrently with or independently of its simulation, design goals were to :

- allow concurrent operation of a simulator and SOTA, data being passed from the simulator to SOTA as the simulated network changes state; and
- allow SOTA to store the data supplied by the simulator, and to later view this stored data as if it were being supplied directly from another simulation.

Two ART simulators were available for the project. Sandy Leadbeater's PlaNet-based ART simulator, and Jagath Samarabandu's "ART1".

Source code was available for both simulators, and therefore the design was free to define both the content and mechanism used for data transfer. Nevertheless, an additional goal was constructed, being that the communication between the simulator and SOTA should be as simple as possible.

Further, the design included the specific goal of restricting SOTAs functionality to visualisation alone. That is, some low-level simulation implementation details are ignored by SOTA, despite their potential use in some circumstances. For example, knowing the value of the L parameter and all bottom-up weights would allow SOTA to work out how the simulator would behave given any input. It was decided, however, that these calculations were the simulator's job alone, and thus L and the bottom-up weights are not passed to SOTA.

Finally, efficient operation was not a high priority. While some care has been taken to avoid flagrant inefficiencies, SOTAs design has not been optimised for performance.

Chapter 5

Visualisation Techniques

The development of visualisation techniques consisted of two distinct approaches.

First, the concept that each input pattern or template could be thought of as a vector in N-dimensional space (see section 5.2) held promise, the representation problem being refined to discovery of an appropriate generic mapping from N-space (ie Euclidean space of N dimensions) onto 2- or 3-space. Techniques to achieve such a mapping were developed, but were found to be either inappropriate, impractical to implement, or both. The template dispersion diagram has been included as an example of an implementable vector-oriented representation.

The second approach (section 5.3) was to think of input patterns and templates as sets of features. The major distinction between the set and vector concepts is that while a mapping between spaces of different dimensionalities is still required, the set representation explicitly shows which components were used to get to the final point where the vector representation shows the final point alone. This conceptual basis led directly to the development of the quadrant; circle; and step representation techniques, and thence indirectly to the development of vigilance graphs and input-relative diagrams.

The following sections detail the ART network configuration used as input for all example diagrams in this chapter, followed by representation development under the N-vector and N-set approaches.

5.1 Example ART Configuration

This chapter includes several example diagrams, all based on a common ART network configuration. This section describes the configuration used, originally introduced in [2].

The example application domain is character recognition, with 25 features available for pattern discrimination. These features are arranged in a 5 by 5 (row-major) matrix.

The input pattern space is restricted to the upper-case versions of the first twenty letters of the alphabet.

An ART network to classify patterns in this space is configured with 25 input features (N); an arbitrary (but greater than 5) number of template nodes (M); and a vigilance level (V) of 0.5. The full range of allowed input patterns are presented (in alphabetic order) twice, resulting in the development of five committed templates :

a	b	c	d	e
11110	10000	11111	10001	11111
10000	10000	00000	10000	10000
10000	10000	00100	10101	11111
10000	10000	00000	10000	00001
10000	10001	00100	10001	11111

as well as $M - 5$ uncommitted templates (all features present).

Note that the templates bear no meaningful relationship to their labels. Template "e" in fact exactly represents the input pattern for "S", while template "c" represents the common features of input patterns for "I", "J", "S" (on the first iteration), and "T".

5.2 N-feature Vectors

If each ART input feature is represented by a different spatial dimension, then any input pattern is represented by a vector in N-space.

Further, given ART-1, each feature is either on or off, and thus each dimensional component of the vector is either 1 or 0 respectively. Therefore, any ART-1 input pattern or template can be thought of as a vector to one of the vertices of a unit side length N-cube.

The problem of representing any aspect of the input space, including representation of any input pattern or template, thus becomes how to map the vertices of an N-cube to 2- or 3-space. An infinite number of unambiguous representations must be possible, since the number of points possibly represented is finite for any value of N.

5.2.1 Choosing a Mapping

Given any unambiguous mapping, every possible point of the input space can be represented. Thus, any combination of templates and/or input patterns can be displayed to the user, and the "template dispersion" design goals can be met (ie any input pattern(s) or template(s) can be displayed, and therefore any or all template vigilance regions can also be displayed).

Considerable thought was therefore given to the choice of an appropriate mapping. A simple acceptance criterion was developed, termed "locality". Locality is the property that the distance between points in the mapping space is proportional to the distance between the original points in N-space. Any mapping which exhibited locality would be accepted.

The first factor on which discrimination between mappings could be performed was the number of features "on" in any vector. The length of any vector is proportional to the number of features "on" (in fact, in Euclidean space, $\| \overline{v} \| = \sqrt{C_v}$, where $\overline{v}$ is some binary-valued feature vector and C_v the number of features "on" in $\overline{v}$). Any mapping which exhibited locality would therefore have to group vectors according to their length - all a common distance from the representation of the origin (ie on concentric circles from the central origin).

Further, within each group, each vector would need to be the same distance from every other vector with the same number of differences. For example, in 4-space, the vector (1000) is the same distance from vectors (0100) (0010) and (0001), each of which have exactly one feature "on" and exactly two differences from (1000). In some 2-space mapping, therefore, we would require 4 points to be found on a circle about some origin, such that any three are the same distance from the other one. This is impossible without ambiguity.

The next step was to weaken the mapping acceptance criteria, saying that "near enough is good enough". A diagrammatic technique was therefore developed, such that all vectors of the same length were mapped onto an arc all of whose points were the same distance from the origin, the vectors ordered within the group by rising binary representational value (ie (0010) would come before (0100) but after (0001)). The locality exhibited was only approximate, resulting in some difficulty interpreting the relationship between templates. Difficulties were experienced in aligning points on different arcs correctly (an even spacing was not always appropriate as dimensionality grew), and the concept of vigilance regions was weakened (since they could be non-contiguous).

A slightly different approach was therefore considered, being that the actual mapping would be determined by the vigilance region dispersal. That is, instead of making non-contiguous vigilance regions from "hard" mappings, the idea was to calculate the size of all vigilance regions first, then retain the contiguity of these regions when the actual mapping was devised (ie work out how many vectors of each length fall in the vigilance region of each template, draw an appropriately shaped region for that template, and use some mechanism [eg ordering of binary representation] to provide unique mappings of the points within each region). The problems with this idea are :

- that it is in general impossible to truly reflect the relationship between vigilance regions (since one may border on several others, and that's what an ideal representation should show); and
- that the number of calculations required is too large to be practical. Even the "toy" example network of section 5.1 has 25 dimensions, resulting in over 33 million data points to map. Many of these points would require region membership recalculation on any change to the template set.

A more constrained approach to vigilance representation is therefore required if the design is to be realistically implementable.

Vigilance regions are not the only aspect of an ART network for which representations are sought, however. Any mechanism which produced an unambiguous mapping could display some representation of the template set, possibly giving a viewer some means of assessing differences or similarities between specific templates. Thus, the template dispersion diagram was devised.

5.2.2 Template Dispersion Diagrams

Template dispersion diagrams are a vector-oriented display mechanism, providing a unique mapping from binary vectors in N-space to points in two dimensions. Rather than showing all the components used to construct a pattern representation, template dispersion diagrams calculate the mapping and draw a line segment to the end point alone.

Figure 5.1 shows an example diagram, with the angle (as a clockwise displacement from the "12 o'clock" vertical) of each template-representative line segment equivalent to the template's binary representation, and length proportional to the number of features each contains. The concentric circles are "contour circles", each representing some specific template length.

While pattern size is represented implicitly by each template's angular displacement, the resultant angle is very difficult to interpret meaningfully. Therefore, the explicit (length) representation of pattern size has also been included.

This representation technique has three valuable properties :

- a single image gives a very clear indication of the status of the network. The number of committed templates can clearly be seen, with some minimal notion of similarity;
- the relative lengths of different templates is readily apparent, the figure 5.1 showing, for example, that template "e" has a very different length from the other committed templates;

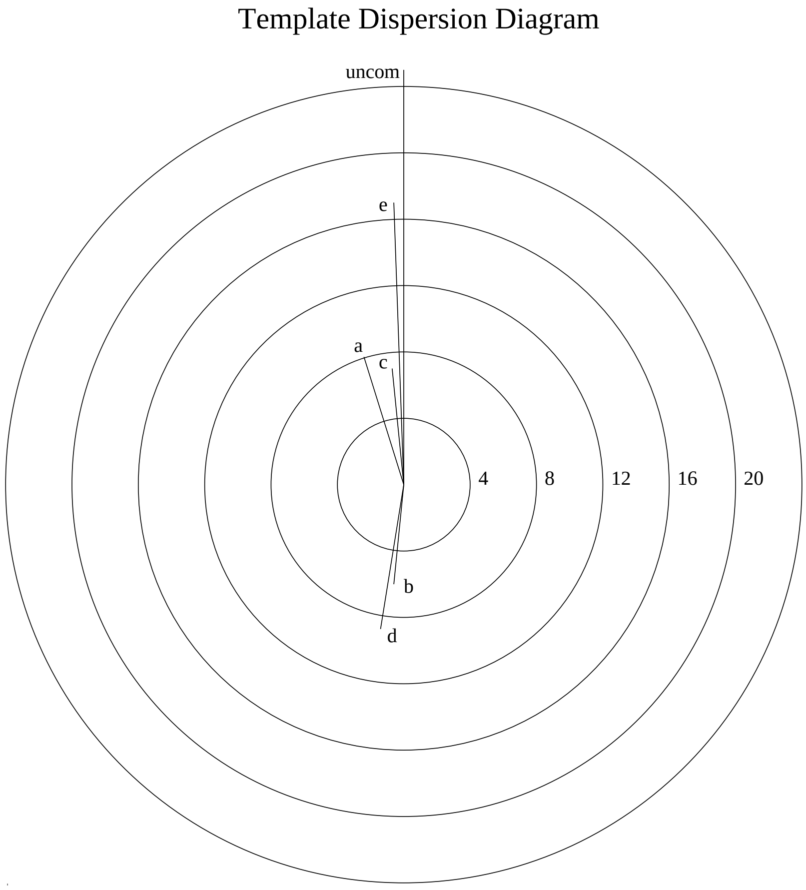

Figure 5.1: Template Dispersion Diagram

- a very clear representation of the general result of ART network learning is given. All templates begin as uncommitted nodes, with learning successively removing features to find the common features of the represented class. Template dispersion diagrams will show this process clearly, with template representations becoming shorter and rotating anti-clockwise as learning progresses.

The technique does have a serious flaw, however. The similarity between template representations is very dependent on feature order. Features whose position is early in the input pattern have much more effect on a template's representation than those whose position is late. Figure 5.1 shows close representation similarity between templates “a” and “c”, and between templates “b” and “d”. While, as we shall see in section 5.3, these particular groupings are relevant, such a result for the example diagram is largely an artifact of the similarity or otherwise of the first five template features only.

5.2.3 Summary

The interpretation of input patterns as N-vectors, while leading to the development of the potentially useful template dispersion diagram, was rejected as the generic basis for ART visualisation techniques.

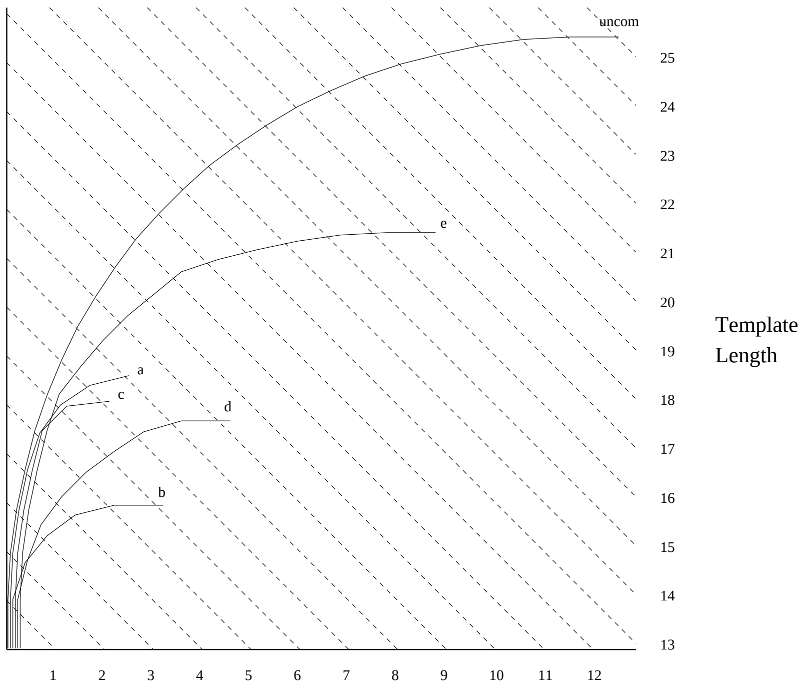

Figure 5.2: Quadrant Diagram

5.3 N-feature Sets

N-sets are an original alternative basis for representation design.

The major difference between vector and set conceptual bases is that the emphasis changes from the end-point that the vector defines to the components of which a set is composed. That is, provided a unique representation of each feature of a set is known, and is shown as part of the representation of the set as a whole, then unique set representations must be produced.

Such a simple concept has produced many alternative representation techniques, discussed by the remainder of this section.

5.3.1 Quadrant Diagrams

The first technique devised was based on a "quadrant". Each feature is uniquely represented by a clockwise angle from the upward vertical of between 0 and 90 degrees. Templates are drawn as a connected series of line segments, each segment representative of one feature of which the set is composed (ie pointing in the associated unique direction). Unique representations of any template or input pattern can therefore easily be constructed.

Figure 5.2 shows a quadrant diagram for the example network. Template representations are labeled near their termination, with uncommitted templates (all features present) forming an upper left bound to the space able to be occupied by committed template representations.

The dashed diagonal lines are contour lines, each of which represents a specific template length. Note that this length representation is slightly more complex than that used in the template dispersion diagram, since the visual clarity of contour lines (as opposed to circles or arcs) requires individual line segments to be of unequal length. Line segments which are close to either the vertical or horizontal are longer than those in-between. Elementary trigonometry allowed simple solution of this problem, a pseudo-code version of the algorithm used being included in figure A.1.

In order to assist visual discrimination between the template representations, each is displayed with a minor x-increment from its immediate predecessor, the uncommitted templates being drawn first. (This x-offset visual discrimination technique is also used in the circle and step diagrams of sections 5.3.2 and 5.3.3, and the vigilance graphs of section 5.4.1.) Note that this incremental offset causes the minor misalignment between line segment termination and the contour lines, and thus will not scale well. Large numbers of templates could display misleading length information, since the offset of later templates would become significant by comparison with the distance between contour lines. Use of colour, displacement along a z-axis, or some other visually discriminating technique (where supported by the display medium) would be better.

Quadrant diagrams have the following advantageous properties :

- the similarities and differences between templates can be seen more clearly than in the template dispersion diagram;
- template lengths can be clearly seen; and
- the learning process will be clearly visible.

Their disadvantages are that :

- while the influence of feature ordering is by no means as marked as in the template dispersion diagram, there remains a major distinction between the representations of features early in the pattern from those that occur late; and
- while each feature has a unique representation, it can be difficult to distinguish a feature from its neighbours. Individual feature discrimination is difficult;

The concept of feature ordering is worthy of additional discussion. The problem quadrant diagrams have with feature ordering is that the visual difference between feature representations varies with those features' relative position in the input pattern. At the extremes, the first feature is represented by a vertical line segment and the last by a horizontal. Thus, if we consider a domain of patterns each of which has only one feature "on", two of these patterns whose features are further away from each other (in the pattern) would appear to be *more different* from each other than two whose single features were adjacent. This runs contrary to ARTs interpretation that all features are of equal importance, their ordering being completely arbitrary.

Two potential solutions were devised, being circular diagrams (see section 5.3.2) and the possibility of alternative ordering.

Alternative ordering is based on the idea that the ability to convey that some feature A is more different from feature B than feature C may be valuable in some domains. Some spatial or geometric patterns may well naturally exhibit this property (ie no reordering may be required).

More generally, as an ART network learns there may well *develop* distinctions between features based on their relative discriminatory power. Once a number of templates have been learned, some features typically occur much more frequently than others. Features could therefore be ordered by the frequency of their occurrence to capture their relative discrimination.

The problem with this particular example of alternative feature ordering is that the representation sought would not (necessarily) be provided by the new ordering. That is, the ordering is intended to express the relative discriminatory power of each feature, but the mathematical foundation on which it is built is insufficient. Instead, the most discriminatory

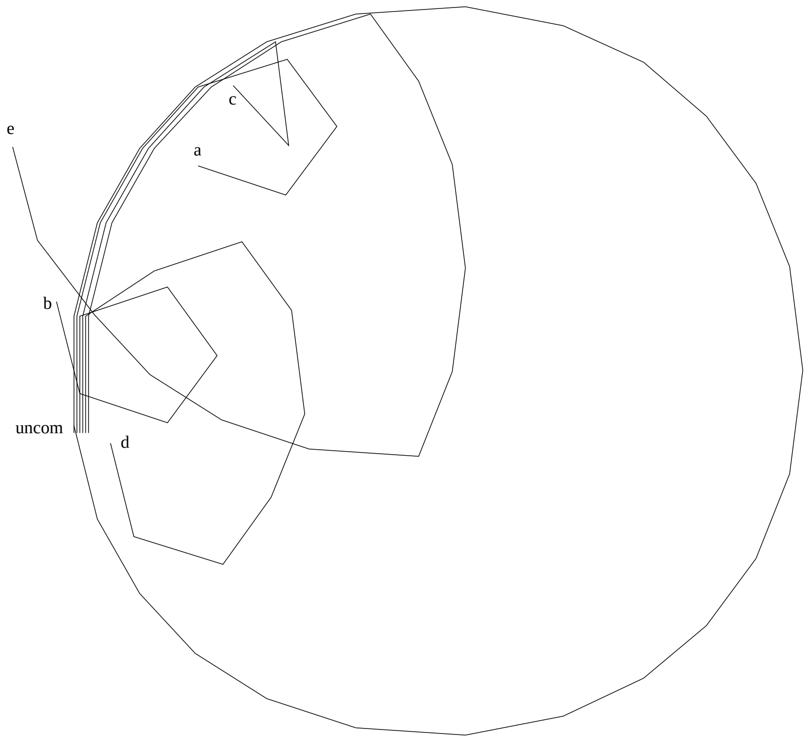

Figure 5.3: Circle Diagram

feature should be chosen to partition the domain, with subsequent discrimination evaluation based on the frequency of occurrence of features *in each partition*.

While the design of SOTA does allow arbitrary feature ordering, a limitation is that feature ordering must be expressed linearly. Thus, the concept of some non-linearity as required to truly represent relative disciminatory power cannot generally be provided - there is no mechanism to express different feature sub-ordering based on the presence or absence of an earlier feature.

A refinement following the introduction of arbitrary user-defined feature ordering was to allow ordering to be chosen at either a global or diagram-specific level. Thus, different orderings can be directly compared (by simultaneous display of diagrams identical except for the order of their features) while also allowing reconfiguration of all diagrams in a single step.

5.3.2 Circle Diagrams

Circle diagrams (see figure 5.3 are very similar to quadrant diagrams, the essential difference being that the angle of line segment orientation representative of each feature is here a fraction of 360 degrees. The rationale behind the creation of circle diagrams is to reduce the difference in representation of features at opposite ends of the pattern.

That is, the angle representing the N'th feature is very similar to that representing the first, which the quadrant diagram would have shown as being very different.

Such a change provides the circle diagram with a number of desirable properties. First, in similar fashion to the quadrant diagram :

- the similarity between templates is clearly visible (for example, note the similarity between templates "a" and "c"); and
- the learning process will be obvious, with templates constricting as features are removed.

However, additional properties the circular diagram provides are :

- better ability to detect similarity between template subcomponents (note the similar hook shape toward the end of templates "a" and "b");
- individual features are more distinctive, and thus their presence in a number of templates is easier to detect; and
- "runs" of adjacent features are easier to detect, with "arcs" appearing where adjacent features are present, and sharp changes in direction indicating that several adjacent features are absent. The representation of template "e" clearly shows the presence and absence of several feature runs. Feature order is obviously critical to the likelihood and significance of runs.

The main disadvantages of the technique are that :

- template length information is obscured;
- the difference between representations is only reduced, not eliminated. The representation of the first feature and the $N/2$'th feature are now the extremes; and
- the visual distinctiveness of features is purely a side-effect of their number. An input pattern of 100 features would cause the same distinction problems for interpretation of a circle diagram that 25 features causes for the quadrant. It's better, but not arbitrarily so.

Note that the uncommitted template representation (the complete "circle") no longer delimits the diagram space able to be occupied, instead providing a general indication only. Its presence, however, is surprisingly useful for visual comparison of feature representations.

The algorithm used to calculate the co-ordinates of each component of each template representation is included in pseudo-code form in figure A.2.

5.3.3 Step Diagrams

Step diagrams are the final variant in this series, being a similar representation to both the quadrant and circle diagrams, but exhibiting an individual set of desirable and undesirable properties.

The major difference between the step technique and the others previously presented is that it has an explicit representation of absent features - template dispersion, quadrant, and circle diagrams implicitly represent feature absence by showing nothing at all.

Step diagrams, as in figure 5.4, are drawn in a two-dimensional graph, pattern features spaced across the x-axis. Features present in a pattern are represented by a slanted line segment above the appropriate x-axis position, while absent features are represented by horizontal lines.

The advantages the technique provides are that :

- template lengths are explicitly displayed (by the height the representation finally attains);
- the presence or absence of particular features in any or all templates is clearly represented. Figure 5.4 clearly shows that all templates include the first feature, while none include features 7,8,9,10,17,18, or 19;
- feature runs are clearly displayed; and

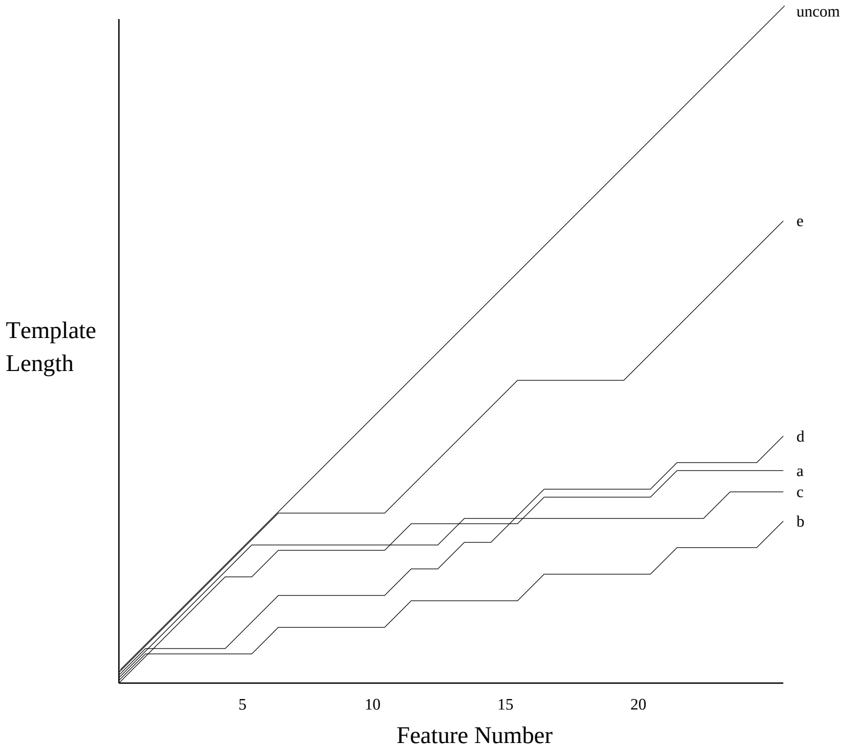

Figure 5.4: Step Diagram

- the effect of the learning process will again be obvious.

The major disadvantage of the technique is its relatively poor template comparison ability. Similarities are not nearly as clearly defined as in the circle diagram, either between templates as a whole or their subcomponents.

Figure A.3 includes a pseudo-code version of the very simple algorithm used to calculate co-ordinates for the step diagram.

5.3.4 Summary

Quadrant; circle; and step diagrams, presented here in the order of their creation, share a common N-set conceptual basis. When implemented in SOTA, these techniques portray one or more patterns (be they templates or actual input patterns) containing features ordered as the user sees fit.

Each has unique advantages and disadvantages, underlying the importance that the SOTA design places on the ability to simultaneously display multiple diagrams.

An enhancement to these techniques is to allow any diagram to show not the current state of the network, instead showing the history by which that state was developed. SOTA thus allows a user to follow the development of a template by displaying new template states and not deleting previous representations. The effect of the learning process is therefore directly depicted.

Psuedo-code versions of the algorithms used to calculate co-ordinates for each representation technique are included in appendix A. These algorithms are translated from the "C" programs which were used in the construction of the example diagrams.

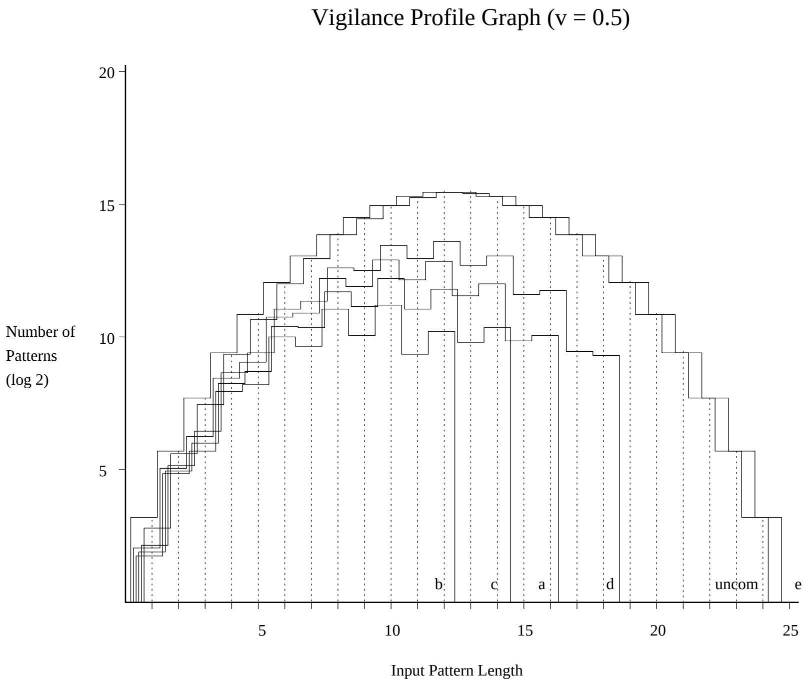

Figure 5.5: Vigilance Profile Graph

5.4 Other Techniques

A variety of additional representation techniques have also been developed. All example diagrams represent the ART network configuration as defined in section 5.3 above.

5.4.1 Vigilance Graphs

As discussed in section 5.2.1, it is impractical to implement true vigilance region representation by exhaustive calculation.

Two vigilance representation techniques have been developed, however, each of which operates on the assumption that all vigilance regions are discrete (ie each region is calculated on the assumption that its template is the only template in the network). This assumption is, in general, invalid, but useful representations can nevertheless be produced given arbitrarily large networks. The important advantage the assumption provides is that it reduces the calculation problem from finding which *particular* input patterns fall into a particular vigilance region, to providing a general profile of a region's shape across different input pattern lengths.

Considerable thought was given to further improving vigilance representation. Additional deliverable techniques were not developed, but the potential basis of a future approach is discussed in section 7.

Vigilance Profile Graphs

Vigilance profile graphs (see figure 5.5) are shown as bar graphs with a (discrete-valued) x-axis of input pattern length and a y-axis of the base 2 logarithm of the number of input patterns able to be accepted. (Note that the number of input patterns a template can accept can be very large - maximally $2^N - 1$ (33554431 for the example configuration) - and thus base 2 logarithms are preferred over raw values for the y-axis.)

The graph should be interpreted as showing the number of input patterns of each possible length that would be accepted by each template if that template were considered in isolation from all others.

Thus, while vigilance profile graphs do *not* necessarily show the exact shape of vigilance regions (since interaction is ignored), nor how different templates are arranged relative to each other (which vigilance regions border on or overlap which others), they do show the general (maximal) shape of those regions for networks of arbitrary size.

Note that some insight into the operation of ART networks was gained while testing the algorithm on which this technique is based. Consider a network with a pattern size (N) of 4. The vigilance profile for templates of length 2 and V 0.6 gives a split of (2 1 2 0), meaning that 2 patterns of length 1 will be accepted; 1 of length 2; 2 of length 3; and none of length 4. Superficial intuition would suggest that such a split is incorrect, since the general shape of such a profile is as for a binary distribution (small at the extremes and large in the middle).

However, such an intuition is wrong in this case, as can be seen by considering example template (1100) which would accept (1000); (0100); (1100); (1110); and (1101). The similarity metric of the templates of length 3 is 2/3 and thus exceed V at 0.6, but any other patterns of length 2 could only match on one feature, and thus have a lower similarity metric (they need a vigilance level of 0.5 or lower for acceptance).

Larger examples of this effect can be seen in figure 5.5, where significantly fewer templates of odd length are accepted than those of a shorter even length. A specific example is that the vigilance region of template "b" (which includes 6 features) will accept 73644 templates of length 10, and 27132 templates of length 12, yet only 11628 of length 11.

For a pattern of length 10 to exceed a vigilance level of 0.5, at least 5 features must be in both the pattern and the template. Thus, the number of "present feature" combinations available is the number of ways 5 items can be chosen from 6 (henceforth denoted in binomial coefficient form $\begin{pmatrix} k \\ p \end{pmatrix}$, where p is the size of the pattern to be chosen from k possible features, and thus

$$\begin{pmatrix} k \\ p \end{pmatrix} = \frac{k!}{p!\,(k-p)!}$$

), which is 6. 5 more features are required to bring the template length to 10 (10 - 5 = 5), and since these can't include any of the template features (25 - 6 = 19), the number of possibilities is $\begin{pmatrix} 19 \\ 5 \end{pmatrix}$ which is 11628. Since any combination of present features can be combined with any combination of "others", 6 and 11628 are multiplied to give 69768.

A similar calculation is performed for the case where all 6 template features are present, this single possibility being multiplied by $\begin{pmatrix} 19 \\ 4 \end{pmatrix}$ to give 3876. The two products are then added together to give 73644.

A pattern of length 11, on the other hand, only has one possible combination of present features $\begin{pmatrix} 6 \\ 6 \end{pmatrix}$, which can be multiplied by $\begin{pmatrix} 19 \\ 5 \end{pmatrix}$ to give 11628. This total is exceeded by a pattern of length 12, since while there is still only one combination of present features available, this can be combined with a larger number of other combinations $\begin{pmatrix} 19 \\ 6 \end{pmatrix}$ which is 27132.

Thus, the apparent anomaly is caused by a constraint imposed by the vigilance level. As the length of input patterns under consideration grows, so closer adherence to the template will be required if the input pattern is to exceed vigilance. Further, for those pattern lengths

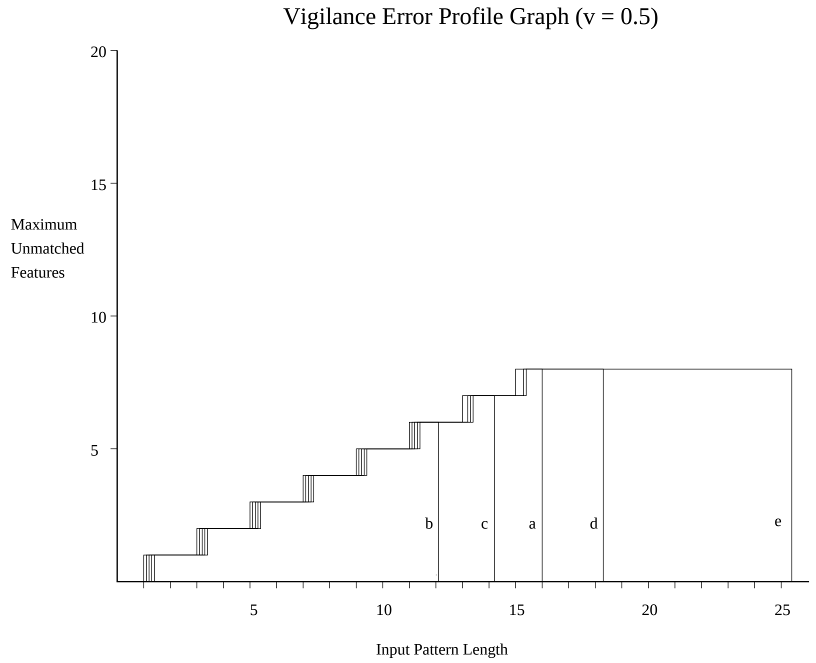

Figure 5.6: Vigilance Error Profile Graph

where an equal number of features in common with the template are required, the length which allows more combinations of "other" features will have the larger vigilance region.

A pseudo-code version of the algorithm used to calculate vigilance region sizes is included in figure A.4.

Vigilance Error Profile Graphs

Vigilance error profiles are a refinement to the vigilance profile representation technique.

Vigilance profiles are problematic in that a logarithmic scale is required for the y-axis of the diagram. The values shown do correspond to real input pattern possibilities, but their individual importance can be reduced given their large discrepancy in scale.

Vigilance error profiles, on the other hand, delimit the vigilance regions they represent in terms of the number of features input patterns of various lengths can contain which do not match features present in the relevant template (and are thus "erroneous"). Again, templates are represented by connected line segments across a bar graph with an x-axis representing input pattern length, but the y-axis now represents the number of erroneous features present in such an input pattern and can thus have a simple integer scale constrained by N.

Figure 5.6 shows an example vigilance error profile graph. Note that no representation of the uncommitted templates is visible, since uncommitted templates have all features on and thus no error (feature present in the input but not in the template) is possible.

The representation makes the implicit assumption that input patterns with fewer than the maximum number of errors will also fall within the same region. Note that this assumption is not entirely valid - there is a minimal error *requirement* for every input pattern which is longer than the corresponding template. For example, any pattern of length 7 must have at least one error when compared against a template of length 6. Such an enhancement could be made to this diagram, but only at the expense of visual clarity and a minor change to the semantics (the y-axis would represent both error bounds, not just its maximum value).

A disadvantage of the vigilance error profile representation is that it is only useful when

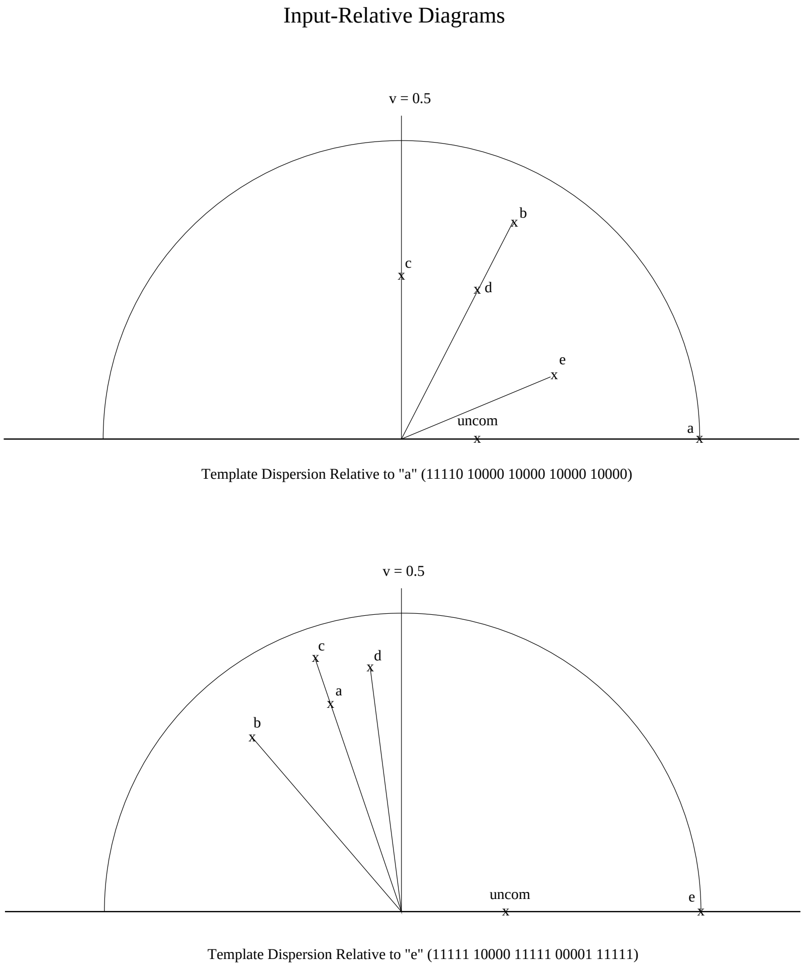

Figure 5.7: Input-Relative Diagrams

the vigilance regions really are discrete. That is, provided we could efficiently calculate the exact number of patterns which fell into each vigilance region, vigilance profiles could show the true state of each region (explicitly representing the number of patterns each region represents). Vigilance error profiles, on the other hand, only distinguish between features based on whether or not they are present in the template. Thus, while the technique gives a clearer result subject to the stated assumptions, there is no simple way by which it can be extended to represent overlapping regions (where one must know *which* features can be safely omitted from [or erroneously included in] particular patterns).

A pseudo-code version of the algorithm used to calculate vigilance error sizes is included in figure A.5.

5.4.2 Input-Relative Diagrams

Input-relative diagrams incorporate several different pieces of information in the same display. An input-relative diagram is empty when no input is present, thus mirroring the activation state of an ART network in the same situation.

The original motivation of this technique was to display the processes invoked by presentation of an input pattern. The technique developed succeeds in portraying the order in which winning F2 nodes are chosen (the bottom-up search path); the extent of the intersection between each template and the input pattern (the similarity metric); and compares each such intersection against vigilance.

The basic form of the diagram is the top half of a semi-circle, with an origin at its centre (see figure 5.7).

Within this framework, templates are represented by points inside the semi-circle. The distance of each point from the origin represents the activation at the corresponding node of F2, the semi-circle representing the position occupied by any template which exactly matched the input pattern. Thus, the order in which templates are activated (ie win the competition) is represented explicitly by how close template points are to the semi-circle.

Further, the angle of offset (clockwise from the horizontal) of a radial to any point represents the intersection of each template with the input pattern divided by the size of the input pattern (ie essentially the vigilance calculation). Thus the value of the vigilance parameter itself can be shown as a radial from the origin to the semicircle, no template left of which could achieve resonance with the input pattern.

Two example diagrams are given in figure 5.7. These diagrams are built from exactly the same ART network configuration, but represent the markedly different state achieved given the presence of different inputs. Vigilance, at 0.5, is represented by a vertical radial.

Note that the template which corresponds exactly to each input is at the right hand intersection of the activation-level semi-circle and the base. Since each is an exact match, the activation will be the maximum possible for that input pattern (thus the distance from the origin will be maximal) and the similarity metric is also maximal (thus the radial angle is 180 degrees).

The uncommitted template representations always have the maximal offset angle, since they will always match any input pattern absolutely. However, due to small initial bottom-up weights, their corresponding F2 nodes will only achieve small pre-competition activations, and thus the uncommitted nodes representation will be close to the origin.

The ability to produce this diagram requires knowledge of each F2 node's pre-competition activation level. These values could be calculated by the tool given knowledge of the L parameter (the value of L helps to determine the size of the bottom-up weights - see section 2.3), but lead SOTA toward actual simulation rather than observation. Thus, input-relative diagram production requires supply of actual F2 node pre-competition activation levels. This high data transfer requirement is the major disadvantage of the representation technique.

Further, the scale of the distance to the semicircle may be undefined. Since knowledge of L is specifically excluded from the visualisation tool, the actual maximal activation level cannot be arbitrarily calculated. As an alternative, SOTA can therefore use the maximal activation reached (perhaps incrementing this value slightly if the maximally active template is not an exact match) as the value corresponding to the maximal distance, and impose a linear scale in-between. Note that the scale is not of logical significance since all weights are relative to each other, but a well-chosen scale will enhance the user's ability to see that relativity.

The particularly valuable properties input-relative diagrams possess are that :-

- the user can easily see the effect of a different level of vigilance, since movement of the vigilance radial could allow inclusion of some template whose similarity metric was insufficient for resonance, or exclusion of some template whose similarity metric was sufficient; and
- the diagram explicitly portrays the mechanism by which ART networks select templates. This representation is invaluable given the overall goal of assisting users to understand the processes involved in ART networks.

Where the other representation techniques introduced in this chapter only convey information regarding the state of the network (the configuration of its templates, or an approximation of what this means in terms of classification of the input space), the input-relative diagram shows *why* the network behaves as it does. Given the original project goal, creation of this technique is the most valuable result.

Chapter 6

Design

The visualisation techniques discussed in chapter 5 have been combined into a high-level design for an interactive ART visualisation tool (happily named SOTA - State Of The ART).

This chapter presents that design.

6.1 Architecture

The overall function of the SOTA system is to provide dynamic display of various views of an ART simulation.

Three major functional components are required, being :

- the data (ANN) subsystem;
- the display subsystem; and
- the control (user) subsystem.

These components are discussed in turn in the following sections.

Figure 6.1 displays the basic SOTA architecture.

6.1.1 Data Subsystem

The data subsystem defines the mechanism (and format) by which data is received by SOTA.

Data is accepted in an arbitrary (typically file-based) format from an ART simulator, then translated to a standard format before being passed to SOTA for display.

Data translation is performed by an independent process, using code specific to the particular simulator in use. The standard format output from the translation process is fed through a stream-based transport mechanism to SOTA.

This structure allows simple use of playback files. Standard format data may be recorded into a playback file while a simulation is taking place. The file may later be "played back" by use of an appropriate translator whose function is, in this case, restricted to data delivery alone. Thus, visualisation may be performed independent of the existence, let alone concurrent execution, of an ART simulation.

The data susbsystem handles data of two basic kinds :

- configuration data :
  - basic parameters N, M, and V;
  - initial template configuration (if some training has already occurred);
  - simulator name;

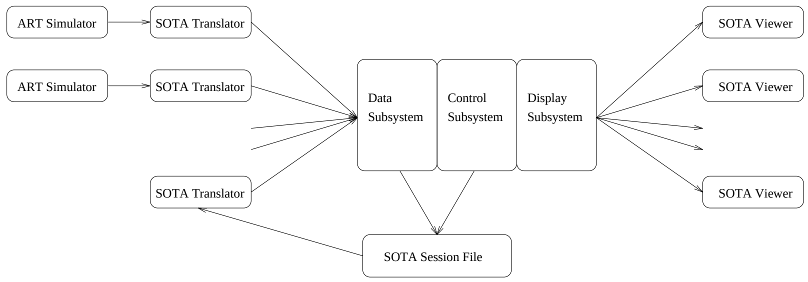

Figure 6.1: SOTA Architecture

- session name; and
- template/feature labels.

- simulation step data, consisting of :
  - an input vector;
  - the resultant F2 node pre-competition activation values;
  - a referent for the resonance-achieving F2 node; and
  - the new template thus learnt.

All configuration data except the N, M, and V parameters are optional.

Note that extra control information may be passed across this interface when replaying playback files to allow recreation of the same SOTA environment used during recording. Control information is considered to be step data in order that each transformation in SOTA's state is recreated between the same simulation steps as occurred in the original.

6.1.2 Display Subsystem

The display subsystem is built on the concept of "viewers". Each viewer provides a particular view of a network, displaying a particular type of diagram or selection of textual information.

Viewers are largely independent, though they do share global ART state information. Any change in that state, as communicated across the data subsystem, prompts an update event for all active viewers.

Viewer types are :

- template dispersion diagram;
- quadrant diagram;
- circle diagram;
- step diagram;
- vigilance profile graph;
- vigilance error profile graph;
- input-relative diagram; and
- textual information.

All but the last of these viewer types simply display their namesake diagram or graph as described in chapter 5. The exception, textual information, supplies a variety of miscellaneous information about the current simulation in a textual form. The content displayed is :

- the simulator name;
- the visualisation session name;
- the initial configuration parameters (N;M; and V);
- the proportion of F2 nodes which represent uncommitted templates;
- the number of input patterns presented;
- the input pattern most recently presented;
- the resultant F2 node activation sequence; and
- the resultant F2 winning node;

Several viewers of the same type can exists at the same time - SOTA itself places no limit on the number of viewers used. Multiple viewers of the same type need not produce identical displays, however, since each viewer can be configured in many different ways (as discussed in the following section).

Note that the quadrant; circle; step; and template dispersion diagrams are not restricted to displaying templates alone. Instead, these viewers also display a representation of the input pattern and highlight the resonant template before the new network state is displayed. Thus, the user can easily follow the input-oriented process by which the network changes state.

Template and feature labels, provided either as initial configuration data or entered dynamically, allow meaningful customisation of diagram component labelling.

6.1.3 Control Subsystem

The control subsystem defines the mechanism by which a user can control the functions performed by SOTA.

Three different levels of control are available, being

- global configuration;
- viewer creation; and
- viewer configuration.

Global configuration provides a mechanism whereby session-level variables may be assigned values. The variables accessible are :

- the simulator name;
- the visualisation session name;
- the filename into which this session will be recorded;
- labels to associate with each feature node;
- labels to associate with each template node;

- an individual feature node to highlight on all viewers;
- an individual template node to highlight on all viewers;
- the (default) order in which feature nodes are to be analysed by viewers;
- the (default) list of feature nodes to display on each viewer;
- the (default) list of template nodes to display on each viewer;
- a (default) viewer update flag (determining whether viewers will remove old representations or simply add to the diagram [ie provide development history]); and
- a global pause flag (whether or not the session is "live").

Viewer creation is invoked by simple selection of the desired viewer type from a menu.

Once a viewer is created several viewer-level configuration variables are available for dynamic display customisation :

- the order in which feature nodes are to be analysed;
- the list of feature nodes to display on this viewer;
- the list of template nodes to display on this viewer;
- an update flag (whether to insert or overwrite); and
- a pause flag (whether or not this viewer is "live").

Note that not all configuration variables are available to all viewers. Irrelevant variables (such as feature order for vigilance graphs) are simply omitted.

The configuration variables listed provide a simple mechanism for control of a wide range of possible representations. Note that a user can change many of these variables at either an individual viewer or global SOTA level. Several viewers can therefore be configured (and thus synchronised) in a single step, or remain completely independent.

A user can focus attention on subsets of the network state by selecting only some subset of the feature set and/or template set for display.

Quadrant, circle, step, and template dispersion diagrams have the ability to analyse features in a specific order, thus allowing a user to dynamically allocate relative feature "significance" (as discussed in section 5.3).

The update flag controls whether or not the drawing mode is to insert or overwrite. Overwrite mode is the default, where diagrams represent the current state of the network (old representations are deleted when new representations are added). Insert mode does not delete previous representations, thus providing a developmental trace. It is recommended that only one template is selected for display under insert mode.

The pause flag allows viewer "freezing". Visual comparisons between network states at different stages of development can therefore easily be achieved.

Viewers are destroyed either by a command issued locally to that viewer, or by global shutdown of SOTA.

It was decided not to allow a single SOTA session to include visualisation of several ART simulators simultaneously. Instead, it is entirely feasible that a user who wishes to compare the behaviour of simulators could run simultaneous SOTA sessions.

6.2 Implementation Details

A complete implementation of SOTA has not yet been developed.

A variety of "C" programs have been developed to validate the visualisation techniques presented, with the co-ordinates of all objects in all representation figures in this report (with the exceptions of the template dispersion and input-relative diagrams - I plead lack of development time) being generated by these programs. Appendix A includes psuedo-code listings of all algorithms used.

A prototype SOTA program has been written in John Ousterhout's TCL(6.4) language [6] and TK(2.2) [7] toolkit, providing proof of concept for creation; update; and deletion of multiple viewers. This program produces graphical display under MIT's X-windows(11r5) and SunOS(4.1).

No translation prototype has yet been developed, but few implementation difficulties are envisaged.

The general philosophy of implementation is to be as "graphical" as is practically possible. Menu-based selections, easily supported by TCL/TK, are intended to be used extensively.

At present SOTA is implemented in a single process. Each child window is updated by the central process itself, with communication achieved through use of global memory. Implementation experience may determine that separate processes (and therefore a more complex communication mechanism) be used.

A potential problem has been identified regarding diagram labelling. Several diagrams have representations of different templates which end at (or very close to) the same co-ordinates. For example, different templates of the same length will have identical representations in vigilance graphs. Mechanisms to ensure proximity of representations and their associated labels, yet which avoid overwriting, are required but have not yet been devised.

Chapter 7

Future Directions

While the original design goals for SOTA have been met, there are a number of areas in which enhancements could usefully be made.

7.1 Feature Ordering

As discussed in section 5.3.1, the order in which features are analysed can markedly affect the appearance of several diagrams. While ART treats all features identically, there may be some benefit a user can gain from imposing a specific order to features given knowledge of the application domain or the actual network configuration.

For example, the program used to generate the coordinates for quadrant; circle; and step diagrams also has the ability to change feature order so that those features which are used most frequently (across all templates) are represented before those which are less common. The design of SOTA extends this notion to allow the user to order features arbitrarily (at either a global or viewer-specific level) which, combined with the ability to exclude some features from representation altogether, creates a powerful customisation facility able to assist the knowledgeable user to focus their attention.

As mentioned, however, this ordering is linear, and is therefore unable to capture some representations (such as feature's relative discriminatory power) which may well be useful.

Enhancement of the representation techniques presented to incorporate arbitrary non-linear feature order is, unfortunately, not a simple extension. The set orientation on which many of the representations are based requires both that each feature have a unique representation (angles for the quadrant and circle diagrams, x-axis position for the step equivalent) and that this representation be consistent across all feature occurrences. To incorporate non-linearity, however, it must be possible to distinguish between occurrences of the same feature in different patterns (due to the presence or absence of other features in each pattern), and thus consistency of feature-level representation must be lost. Further, given that a unique representation would need to be computable for any feature in the presence or absence of any combination of other input features, the number of unique representations required would be exponentially related to N.

7.2 Enhanced Template Analysis

Another potential improvement is the inclusion of template analytical facilities. As discussed in section 3.6, the ability to automatically discover relationships between features (mutual occurrence or exclusion, for example) may well be useful. Based on this knowledge, a user could more easily translate template configuration into meaningful information for the application domain, whether it be to recognise that a feature is never used, or that two features are linked in some way (possibly representing the same concept, or that the underlying concepts are themselves linked).

Note that it is entirely possible for a feature to be present in one or more input patterns yet not be present in a template - absence from all templates does not necessarily imply total lack of influence. Care is needed in interpreting relationship information, especially given a low level of vigilance.

7.3 Domain-Specific Representations

A number of minor enhancements could well be made to simplify the user's translation process between SOTA representations and the actual application domain.

The first and most obvious enhancement is to use some kind of meaningful feature labelling. The domain used throughout this paper is not a good example, but one can easily imagine others (eg word recognition) where individual features can be meaningfully named.

Further, a representation of ART templates in a manner closely linked to the domain in which the network operates would be useful, both to assist the user in understanding the classification each template represents and to provide a more meaningful referent.

For example, the templates used to generate the example diagrams were introduced in section 5.3 as 5x5 matrices, rather than the 25 feature string which SOTA uses. Template "e" can therefore be seen to be an "S", not simply a binary string.

This and other similar "spatial" application domains should be readily representable, with the concept easily extended to others (eg ASCII characters, syllables). A generic method for performing this translation would therefore be very useful.

7.4 Vigilance Regions

The vigilance graphs introduced in section 5.4.1 both suffer from an inability to represent where regions overlap.

While the exact shape of every data point could be calculated, and thus the regions mapped exactly, there are two strong arguments against including such a facility.

First, the number of possible input patterns one has to consider may be very large (for example, there are more than 30 million where $N = 25$). Second, there is a danger that the visualisation tool would lose its identity, becoming an ART simulator in its own right. The tool should convey what the model does, not *do* what the model does.

One technique in particular has been considered for minimising these problems, in that an extension of the algorithm used to calculate (maximal) vigilance region sizes (see figure 5.5) could be extended to calculate interactions in a semi-generic fashion.

The basic concept relies on analysis of ART's arbitration mechanism. If an input pattern intersects with two templates in the same number of features, and provided L is greater than 1, then the shorter template will win this competition between them. It doesn't matter which of the common features is involved - we can calculate the outcome of the competition "generically".

Further, again given knowledge of L, the concept can be extended to allow discrimination between templates which intersect with patterns in an arbitrary (unequal) number of features, again regardless of which actual features are involved. As a trivial example, given an L parameter greater than 1, all patterns which intersect with pattern (111) in 2 places (thus giving a pre-competition activation of 2 * (2 / (2 - 1 + 3)) = 1) and also intersect with (101) in 1 place (1 * (2 / (2 - 1 + 2) = 2/3)) will actually belong in the first template's vigilance region. Since there are two such patterns, there is a reduction in the number of calculations required to calculate each vigilance region's true shape.

Thus, given that we can readily calculate the split of each template's maximal vigilance region by pattern length (as for the vigilance profile graph), and given that we have a manner of discriminating between patterns of arbitrary intersection with each, there must be the possibility for a generic calculation of the mutual influence of any two templates on each other's vigilance regions. The significant factor is that discriminatory calculations have been raised from operating on individual patterns to operating on groups of patterns with known partial components.

Finally, if a combined vigilance region could be derived during the comparison process, and given that this region could be used as part of the combination calculation, then the number of inter-template comparisons required may be reduced (every template may not need to be compared against every other).

Such a process is quite complex, however, and while some progress has been made on a preliminary analysis, much work is left to be done. In particular, I have not yet shown that such an approach would truly be more efficient than a simple exhaustive test. The advantage gained by performing group-based comparisons may be insufficient to offset the disadvantage of effectively reconsidering all patterns at each comparison. Further, it is not clear that it is trivial to store *which* of a template's vigilance region patterns have been removed by comparison with another template. Doubt is thus cast on the ability to carry the results of one comparison to the next.

Note also that the second criticism expressed at the start of this section (that visualisation tools should be for visualisation, not simulation) is somewhat unanswered, since the process discussed is really a group-oriented ART simulation, definitely requiring access to such internals as the L parameter (which SOTA specifically ignores).

7.5 SOTA Design Extensions

The design of the visualisation tool which combines these techniques (see chapter 6) can be improved in many ways.

First, the ability to select subsets of the feature and template sets for display can be usefully extended, the rationale being provision of some mechanism for diagram simplification where large networks are to be visualised. An automatic viewer creation and configuration process which uses knowledge of the current network state to produce a complementary set of appropriately configured viewers would save the user considerable setup time. Details of which viewers should be created in which circumstances have not yet been devised.

Second, SOTA includes the ability to change, in a single step, the configuration of either an individual viewer or all viewers. A logical extension is the creation of "viewer groups", where reconfiguration of groups of viewers may be achieved in a single step. Thus several groups of viewers which each focus attention on some aspect of the network configuration could be simply co-ordinated without effect on unrelated viewers.

Finally, no trace information is explicitly stored, and thus historical diagrams (where the update flag is set to insert mode) can only be generated from the present developmental stage of the network. Runtime recording of each developmental step would allow retrospective production of historical diagrams. Such an enhancement would also allow dynamic reconfiguration of insert mode diagrams, which is unavailable under the existing design.

As a side-issue, SOTA's design can easily be modified to incorporate visualisation of ART networks that use slow learning (see section 2.4.5. The major difference is that current state data packets would need to be sent from the simulator to SOTA at frequent intervals after an input pattern was presented. The design presented in chapter 6 assumes that only a single learning step (and thus a single current state packet) is required.

7.6 Extensions for ART-2

The representation techniques presented were developed specifically for ART-1, the binary version of the ART paradigm.

ART-2 is an alternative ART implementation, where input features may have real values. Thus, the basic mapping from an N-dimensional input space to a 2- or 3-dimensional representation space is much more complex than for ART-1 (see section 5) since the input space is no longer restricted to a finite number of data points. I will use the term "feature strength" to refer to the real value of any input.

All of the techniques so far presented in this report use only two dimensions. Three-dimensional representations would be more difficult to display (given the two-dimensional structure of both paper and screen), but would definitely not be impossible. Thus, the feature-oriented quadrant; circle; and step diagrams could include a "depth" dimension, with feature strength represented by values in the z-axis. Note, however, that visual discrimination between simultaneously displayed template representations is not always easy with these diagrams in only two dimensions, the use of a third dimension potentially causing additional visual interference.

An alternative approach is to represent feature strength by the length of the line segment used by the quadrant; circle, and step diagrams. While this would obscure template length information (in terms of the number of features present) in both the quadrant and step diagrams (with the circle having no length information anyway), the simple number of features included by any template holds much less relevance under ART-2 than ART-1. Instead, as is in fact the case for ART-1, the actual interest is in the total value of all present features, with relative strengths directly related to the feature's relative significance during the matching process.

Note that the step diagram could display ART-2 patterns particularly elegantly by retaining a regular discrete x-axis (each interval corresponding to a particular feature), increasing the gradient of line segments across that interval with the strength of the corresponding pattern feature.

Template dispersion diagrams could also undergo a similar transformation, with the length of each template-representative vector proportional to total feature strength. Any non-zero template feature strength would be equivalent to the feature being present in an ART-1 interpretation for angular offset calculation.

Vigilance graphs can be similarly modified. While these representations have an inherent assumption regarding the equality of features (the only feature-distinguishing characteristic being their presence in, or absence from, a template), this also translates to a real-valued ART implementation. That is, it remains true that the "closeness-of-fit" between template and input pattern determines the value to be compared against vigilance, the difference from ART-1 being that the contribution of each is not constant. It does not concern us which features are present in an input pattern such that it attains a total input value sufficient to exceed vigilance when matched against some template - it only concerns us *that* such a strength is achieved, with the knowledge that strength can only be achieved by some positive value for features in the input pattern which are also present in the template.

Thus, given transformation of the discrete integer values used by the axes of vigilance graphs into continuous real values, a conceptual shift to an understanding of pattern length in terms of overall pattern strength will result in ART-2 vigilance representations subject only to the same limitations as discussed for ART-1 (see section 5.4.1).

Details such as appropriate granularity for sample points along the x-axis, and the algorithm used to calculate the volume of some vigilance region of the input space (from the vigilance parameter and total template strength alone) have not been derived, but a solution is definitely considered feasible.

Note that mathematical description of vigilance regions as predictably-shaped N-dimensional volumes may result in the derivation of volume interaction formulae whereby the actual size of vigilance regions in a real-valued domain may be directly calculated.

Input-relative diagrams would function acceptably in an ART-2 environment largely unchanged. Actual pre-competition activations of nodes in F2 are transferred from the simulator directly, and thus relative distances of each template's representation from the centre of the semi-circle can be calculated in the same manner as for ART-1. Similarity metric calculation simulation can be performed by division of the strengths of template features (where these features are present in both the template and the input pattern) by the overall input pattern strength, and thus the angular displacement of each template's representation can also be easily calculated. This is in fact the same calculation, less some assumptions possible in a binary domain, as that performed to produce input-relative diagrams under ART-1.

Chapter 8

Conclusion

This paper presents seven new visualisation techniques for use in understanding the processes of ART ANNs, together with a high-level design for a tool combining these techniques.

Template dispersion diagrams are an N-vector approach used to provide a concise representation of overall network state.

Quadrant, circle, and step diagrams are all based on an original interpretation of multiple ANN nodes as an N-member set. This approach markedly distinguishes these techniques from the Euclidean N-space (N-vector) basis used by many existing ANN visualisation techniques.

Vigilance profile and vigilance error profile graphs provide representations of the maximal shape of the set of input patterns which would, on presentation, achieve resonance with each template. Study of the interaction between these vigilance regions is specifically excluded due largely to practical computational limitations.

Input-relative diagrams provide a representation of the state of an ART network given the presence of some input pattern. Comparative pre-competition template-node activation levels (the bottom-up weight search path) are explicitly represented, as is the relationship between the value of the input-relative similarity metric for each template and the vigilance parameter V.

These techniques are incorporated into a high-level design named SOTA, an interactive visualisation tool. Many instances of any diagram can be displayed at one time, and operation can occur either concurrently with a simulation session or from a stored activity file. SOTA allows significant user customisation of several of the diagrams displayed.

Acknowledgements

My general thanks to my family and friends for putting up with me during the development of both this report and the material it presents. Special thanks to Paul Hosking, Aaron Roydhouse, and Sandy Leadbeater for keeping me vaguely sane; Peter Andreae for the wisdom (if not the devastation) of his editorial scythe; and, above all, Guszti Bartfai for deriving sense from my ravings and providing continual enthusiasm, encouragement, and direction.

Bibliography

[1] G.A. Carpenter and S Grossberg. *Competitive Learning: From Interactive Activation to Adaptive Resonance*, chapter 5, pages 213–250. MIT Press, Cambridge, Ma, 1987.

[2] G.A. Carpenter and S. Grossberg. *A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine*, chapter 6, pages 251–315. MIT Press, Cambridge, Ma, 1987.

[3] M.W. Craven and J.W. Shavlik. Visualizing learning and computation in artificial neural networks. Technical report, University of Wisconsin, 1991.

[4] R. Edgar. Hardware design of the ART neural networks. VUW Diploma of Applied Science Project, 1992.

[5] A.R. Leadbeater. ART neural networks in visual pattern recognition. VUW BSc Honours Project, 1992.

[6] J.K. Ousterhout. TCL: An embeddable command language. In *Proceedings of the 1990 Winter USENIX Conference*, pages 133–146, 1990.

[7] J.K. Ousterhout. TK: An X11 toolkit based on the TCL language. In *Proceedings of the 1991 Winter USENIX Conference*, pages 105–115, 1991.

[8] D.E. Rumelhardt and J.L. McClelland, editors. *Parallel Distributed Processing; Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge, Ma, 1986.

[9] E.R. Tufte. *Envisioning Information.* Graphics Press, PO Box 430, Cheshire, Connecticut 06410, 1990.

[10] Various. Scientific visualization. *IBM Journal of Research and Development*, 35(1/2), January/March 1991.

[11] J. Wejchert and G. Tesauro. Visualizing processes in neural networks. *IBM Journal of Research and Development*, 35(1/2):244–253, January/March 1991.

[12] M Zeidenberg. *Neural Networks in Artificial Intelligence.* Ellis Horwood Ltd, 1990.

Appendix A

Algorithms

/* Quadrant - for input pattern length n, calculate the x and y offsets to be used should feature i be present. Note that all angles are measured in radians */

x[0] = 0
y[0] = 0

for $i = 1$ to $(n-1)$ step 1

  if $present(feature[i])$ then
    $\theta = (i * \pi/2)/(n-1)$
    $y[i] = y[i-1] + (1/(tan(\theta)+1))$
    $x[i] = x[i-1] + (1 - y[i])$

  else
    $x[i] = x[i-1]$
    $y[i] = y[i-1]$

  end if

end for

Figure A.1: Quadrant Diagram Offset Calculation

/* Circle- for input pattern length n, calculate the x and y offsets to be used should feature i be present */

x[0] = 0
y[0] = 0

if $odd(n)$ then
    $divisor = n$
else
    $divisor = n + 1$
end if

for $i = 1$ to $(n - 1)$ step 1

    if $present(feature[i])$ then
        $\theta = i * 2\pi/n$

        case

        $0 <= \theta <= \pi/2 :$
            $y[i] = y[i-1] + (1/(tan(\theta) + 1))$
            $x[i] = x[i-1] + (1 - y[i])$

        $\pi/2 < \theta <= \pi :$
            $\theta = \theta - (\pi/2)$
            $x[i] = x[i-1] + (1/(tan(\theta) + 1))$
            $y[i] = y[i-1] + (0 - (1 - x[i]))$

        $\pi < \theta <= 3\pi/2 :$
            $\theta = \theta - \pi$
            $y[i] = y[i-1] + (0 - (1/(tan(\theta) + 1)))$
            $x[i] = x[i-1] + (0 - (1 + y[i]))$

        $3\pi/2 < \theta <= 2\pi :$
            $\theta = \theta - (3\pi/2)$
            $x[i] = x[i-1] + (0 - (1/(tan(\theta) + 1)))$
            $y[i] = y[i-1] + (1 + x[i])$

        end case

    else
        x[i] = x[i - 1]
        y[i] = y[i - 1]

    end if

end for

Figure A.2: Circle Diagram Offset Calculation

/* Step - for input pattern length n, calculate the x and y offsets to be used should feature i be present */

x[0] = 0
y[0] = 0

for $i = 1$ to $(n-1)$ step 1

    if $present(feature[i])$ then
        $x[i] = x[i-1] + 1$
        $y[i] = y[i-1] + 1$

    else
        $x[i] = x[i-1] + 1$
        $y[i] = y[i-1]$

    end if

end for

Figure A.3: Step Diagram Offset Calculation

/* Vigilance Region Size - given vigilance v, calculate the *total* number of input patterns which would fall inside the vigilance region of a template of size n. Also calculate the *split* of that total by input pattern length. Note that cells of array *split* are initialised to 0. */

/* for every (real) subset of the template */
for $i = 1$ to n step 1

/* get the number of "present feature" combinations */

$$present = \begin{pmatrix} n \\ i \end{pmatrix}$$

$j = 0$ /* for every vigilance-satisfying length */
while $i \geq ((i + j) * v)$ /* get the number of "incidental" combinations */

$$others = \begin{pmatrix} k - n \\ j \end{pmatrix}$$

/* update the total combination counter */
$total = total + (present * others)$
/* and the relevant "split" counter */
$split[i + j] = split[i + j] + (present * others)$

$j = j + 1$

end while
end for

Figure A.4: Vigilance Profile (Region Size) Calculation

```
/* Vigilance Error Profile - given pattern length n; template length t; and vigilance v,
calculate the maximum number of erroneous error which such a template could tolerate
in an input pattern of length i */

for i = 1 to n step 1

    /* check to see if a match is possible */
    if (i > ⌈t/v⌉) then
        error = ⊥
    else
        /* how much error can be tolerated by such an input */
        error = i − ⌈i ∗ v⌉
        /* is this much error physically possible (given t) ? */
        if ((N − t) < error) then error = N − t
    end if

end for
```

Figure A.5: Vigilance Error Profile Calculation