A DISTANCE BASED NEURAL NETWORK MODEL FOR SEQUENCE PROCESSING

by

DAVID CALVERT

A Thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 1998

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

This work describes the development of an artificial neural network architecture for sequence modeling and recall. The system described learns from an initial data set which it treats as an exemplar for all later comparisons. One of the principles of this work is to design a system that takes advantage of the architectural features common to neural networks. These features are many simple storage locations (weights) and a collection of simple processing elements. Although many methods currently exist for processing sequences, and the behavior of this network bears some similarity to these methods, it was always the intent to develop a system which operates in a unique manner. To that end, this system does not attempt to perform an existing sequence processing algorithm and simply cast the algorithm into a neural network. It instead tries to store a great deal of sequence information verbatim and then develop generalizations based upon this previously learned material.

The network developed uses two classification networks, either Fuzzy ART or Self Organizing Maps, to contrast enhance the input and output sequence elements. An internal sequencing network builds an association between these classifications during training. The sequence layer also generates a distance value which indicates the similarity between these stored patterns and testing sequences. The system is tested upon several data sets which have results generated using other neural network techniques. Data sets include the NETtalk word-phoneme matchings and the Santa Fe time series competition data. Results from the testing show that the network generally performs as well or better than many neural techniques. However, this technique is much faster than all other neural methods which it is compared against.

Several characteristics of the network are examined. These include the number of input presentations required for the network to focus on a solution and the use of the vigilance parameter in the classification networks as a distance measure.

Acknowledgements

I would like to thank my supervisors, Dr. Deborah Stacey and Dr. Mohamed Kamel, for their guidance throughout my graduate studies. Dr. James Linders, from the University of Guelph, also deserves thanks for his support throughout my graduate studies.

Contents

List of Tables

List of Figures

Chapter 1

Overview

1.1 Introduction

The majority of artificial neural network (ANN) architectures used today do not focus on temporal or sequential patterns. They tend to either associate an input pattern with an output pattern (*heteroassociative*) or cluster input patterns (*autoassociative*). Each input presented to the network is processed without regard for previous inputs. A network which models sequential patterns must be able to reference a context generated from the sum of past inputs in order to process the current input pattern. The network must process several elements of the sequence at one time or maintain a history (context) of previously presented patterns. Without such a mechanism the ANN is reduced to processing each pattern as a single event which may appear identical to many other events.

The problem to be addressed by this work is the modeling of temporal sequence patterns through the use of a neural network. Specifically, sequences of patterns are to be presented to the network which will build an internal representation of their structure. The system can then be used to recall this sequence structure so that it can be compared to a different sequence.

Many non-neural techniques exist which perform sequence modeling and prediction. One

of the most commonly applied techniques is the Autoregressive Moving Average (ARMA) method found in Time Series Analysis. This system uses a set of time-delayed sequence values which are filtered through a series of moving average and autocorrelation coefficients to model the characteristics of the training data. This linear method has a long history of successful application to sequence data and has spawned many derivative techniques. Some variations of the ARMA method address its inability to model non-linear systems. Markov Chains are a common stochastic technique which have been used successfully for many years. They operate by describing the state transitions of a system in terms of their probabilities of occurring. A prediction for the next state in a system is generated using its current state and the probabilities. Delay Coordinate Embedding is a lesser known technique which uses time-lagged vectors to represent the state-space of the system being modeled. During prediction, the input sequences are compared to the embedded vectors, a group of nearest neighbors to the current input is then determined, and a local linear model is used in to predict the next element from the neighbors.

The network developed in this work uses two clustering systems to categorize sequence information. Clustering is achieved in this work using both the Self Organizing Map and the Fuzzy ART network. An internal network is used to build associations between these two categorization systems. The associations are stored in a memory which maps an input element at a relative time step to the appropriate output element for the same time. Once these association have been stored they are used for modeling and prediction. The relative timing of input-output elements learned from the training set is used to identify the similarity to sequences presented during testing. A distance measure is developed for Fuzzy ART which allows for the relative similarity of clusters which are learned by the network to be determined. The distance between individual clusters is then used by the sequence processing network to create a set of distance values which represent the similarity between sequences of elements. The focus of this work is the development and analysis of this sequence processing network.

The sequence processing network is tested against several well known data sets, including

sets A and D from the Santa Fe time series set, the NETtalk data set, and several sets generated for this work. Timing and accuracy results are examined for the test cases. The system is demonstrated to be quite accurate in when used for modeling and prediction, has fast training and recall algorithms, and is stable when presented with novel data.

1.1.1 Motivations

The motivations for this work can be divided into two categories which describe the type of problem examined and the type of solution used to address this problem. The proposed problem is that of modeling temporal sequences. When compared to the existing body of neural network research, the amount of temporal processing research has been relatively small. There are many characteristics associated with artificial neural networks which are beneficial when applied to the learning of temporal patterns.

The technology used to address this problem is an artificial neural network which builds an internal representation for a given training sequence. A neural system was chosen because of its parallel search behavior and corresponding fast evaluation of its state. The ART and SOM architectures are chosen for their modular nature and well understood behavior and interface. In addition, its ability to dynamically create new categories and that they do not forget previously learned categories make the ART and SOM good choices for the systematic construction of modular neural networks [Bar95a].

Sequences processing appears in a wide variety of applications and has had many different technologies developed for use in modeling and prediction. Some of the more common applications include financial, biological, and environmental modeling and forecasting. Techniques which have been used for these purposes include statistical modeling, time series methods, state space reconstruction, and neural networks. The wide range of applications and the large number of existing techniques which have been developed for sequence processing illustrate the need for methods which can model and predict sequences given a sample of the data.

Most of the existing neural network techniques for sequence processing involve a recurrent gradient descent networks. There are several problems which arise from this architecture including the potential to become trapped in local solutions, difficulties when learning long sequences, and problems in analyzing and controlling the weights generated by the network. One of the objectives of this work is to develop a network which uses a different architecture which avoids some of these deficiencies.

1.1.2 Research Objectives

The objective of this work is to design and construct a neural network which can model sequential patterns. The resultant network models input sequences based upon a context derived from the input sequence itself. Inputs will be presented to the network as a sequence of unbuffered individual patterns. Each pattern represents one time step t of the temporal data. This forces the network to build a temporal representation of the input patterns without the benefit of multiple data elements presented to it simultaneously. The network must therefore maintain a memory of past events and uses that as a context against which to evaluate the later inputs.

Performance of the network when modeling sequences should be comparable to that of existing neural network techniques. The network is expected to model sequences with a high degree of accuracy. Due to this systems simple processing structure it should be considerably faster than other neural based methods. In terms of prediction the network is expected to forecast of a sequence with similar accuracy to other neural methods. Virtually all sequence processing techniques stipulate that the model generated is largely related to how well the training data represents the system being modeled. This technique also requires the data set to be representative of the system to be modeled or the resultant model will be incomplete.

1.1.3 Document Structure

This document is divided into five parts. Chapter one is the introduction to the problem. Chapter two examines some of the existing neural network models and looks at methods for temporal processing using neural networks. Chapter three describes the network which is developed to process temporal patterns. Testing data and results are included in chapter four and chapter five examines the characteristics of the new system and summarizes the research results.

Chapter 2

Literature Review

2.1 Neural Network Background

Neural networks are groups of simple processing elements (PEs) joined through a collection of weighted connections. This model is loosely based upon the neural structure found in a brain where neurons are connects to each other through synapses. In a brain the number of connections is vastly greater than those found in artificial networks. Artificial systems tend to be constructed in layers of processing elements and weights. Each layer of PEs completes the processing of its inputs before passing the output to the next layers input or to the system's output. The value calculated in each processing element is known as the activation value. The layer of weights between two sets of processing elements can be considered as a filter for the activations. Activations are often filtered through a layer of weights by calculating the product of the activation and the weight values. These values are then summed in the next layer or in the output of the network. In many networks models this sum is then passed through a thresholding function which introduces a non-linearity to the calculation and bounds the activation values. For a single processing element j the formula for its activation would be:

$$a_j = f(\sum_{i}^{n} w_{ij} a_i) \tag{2.1}$$

where a_i is the activation of all PEs attached to the input of node j, w_{ij} are all weighed connections between nodes i and j, n is the number of PEs connected to node j, and f is the non-linear function [Sim90]. Weights may be unidirectional, allowing values to be summed in one direction, or bidirectional, allowing activations to flow from either of the PEs to the other. In the case of bi-directional weights the activation process more resembles a system settling to an equilibrium state than the processing of layers of activation values and their directed influence on later layers.

Training occurs in ANNs through the adjustment of the weighted connection. Some explicit methods of training exist in which the weights are set initially and do not change there after. Another method of training involves the iterative adjustment of weights to reduce an error value. Training algorithms can be divided into two categories, supervised and unsupervised. Supervised training involves input by a teacher which identifies what is to be learned. Unsupervised training allows the network to draw its own conclusions regarding important features of the inputs based solely on the data itself.

A particularly successful ANN learning algorithm is Back-Propagation (BP) or generalized delta rule. A Back-Propagation network usually consists of two layers of weighted connections which join three layers of processing elements. An error term is calculated at the output of the network based upon the desired output pattern and the output which is generated by the network. This error term is used with a gradient descent method to adjust the weighted connections so that they can more accurately create the desired output. The training process is iterative and requires many presentations of the pattern pairs and many adjustments to the weighted connections before the network is capable of accurately reproducing the association between the input and output patterns.

2.2 ART Models

Adaptive Resonance Theory (ART) differs from gradient decent and energy minimization networks in its approach to adjusting weight values. ART clusters input patterns by ad-

justing a pair of linear filters. There are currently five models of ART network. ART 1, 2, 3, and Fuzzy ART are all autoassociative competitive pattern clustering networks. They are capable of grouping input patterns and of returning the exemplar pattern which they use to distinguish each cluster. ART 1 works only with binary input patterns, ART 2 works with continuous valued inputs, and ART 3 is an extension to the previous two networks which facilitates their use within a bidirectional hierarchy of networks. Fuzzy ART is an extension to ART 1 which allows it to categorize continuous valued patterns through the use of fuzzy logic. The fifth ART model, called ARTMAP, is constructed of two ART 1 systems and works as a heteroassociative pattern classifier for binary patterns. ARTMAP has also been constructed using fuzzy ART networks in order to create a heteroassociative network which works with continuously valued inputs.

2.3 ART Processing Summary and Architecture

All ART models have several structural features in common. These include a two layer architecture and several subsystems which control pattern matching. Each layer of an ART network is known as a field, and may consist of one or more layers of processing elements (nodes). One field is used to process input patterns (F1) and the second is for pattern classification (F2). The fields are joined by two unidirectional weight matrices (filters). The weights leading from the input to the classification field are the known as bottom-up weights, and those leading from the classification to the input field are top-down weights, (see figure 2.1). Layers within the fields are made up of processing elements which calculate an activation value based upon their inputs. Elements within the input field represent the input patterns, classification elements represent the groups into which similar input patterns are sorted.

Two subsystems operate in parallel with the two fields, these are the attentional gain control and orientation subsystems. The gain control subsystem is used to control which fields may become active. No field may produce an output unless it is receiving inputs from

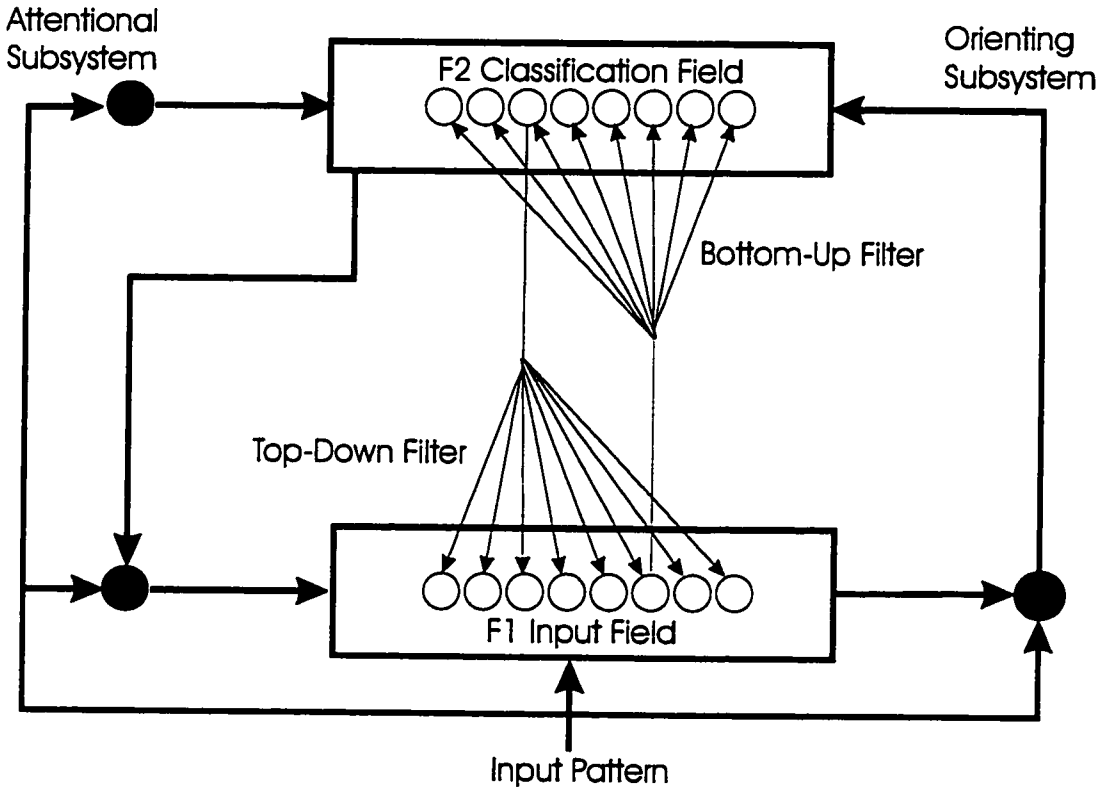


Figure 2.1: ART architecture. F1 and F2 fields are connected by top-down and bottom-up weight adaptive filters (only one connection is shown in each direction) Attentional subsystem allows activations to occur in fields only when appropriate, gain control inhibits F2 elements.

two other sources. Gain control creates a second input to activate fields when it is required and halts it not required. The purpose for this subsystem is not obvious unless the network is considered in the context of a larger system. In this case it is possible for activation to arise in the input layer if the classification layer is stimulated from above (outside the network, perhaps from other networks). This could lead to activation within the input field and learning when no input pattern is present. To avoid this, the gain control system allows learning only when an input pattern is present. The orienting subsystem controls which processing units in the classification layer may be active. If a poor classification is made then this subsystem "turns off" the processing element in the classification field (F2) allowing another element to become active and attempt to create a better classification, (see figure 2.2).

Processing begins when a pattern is presented to the network which creates an activation within the input field. Figures 2.3 and 2.4 display the different steps of pattern classification that take place in an ART network. The input field (F1) requires two input sources to create an activation pattern. The input pattern is one source, the second is the gain control subsystem which non-specifically excites all F1 nodes. This non-specific excitation allows the input pattern to create an activation in F1 without affecting the pattern which this activation forms. This activation pattern is filtered through the bottom-up weight layer. The product of the weighted connections and the input activations are summed in the processing elements of the classification layer. A non-specific activation signal, similar to that sent to the input field, is sent to the classification layer at this time so that the bottom-up pattern can cause activations to form. The classification processing elements compete amongst themselves using a center-on surround-off strategy. This causes the processing elements to simultaneously inhibit the other classification nodes while increasing their own activation values. Once a single element remains active it represents the group into which the input pattern has been classified. At this point the network will attempt to recreate the input pattern which caused the classification by processing the F2 activation through the top-down weights. If the pattern is correctly replicated then the network concludes that

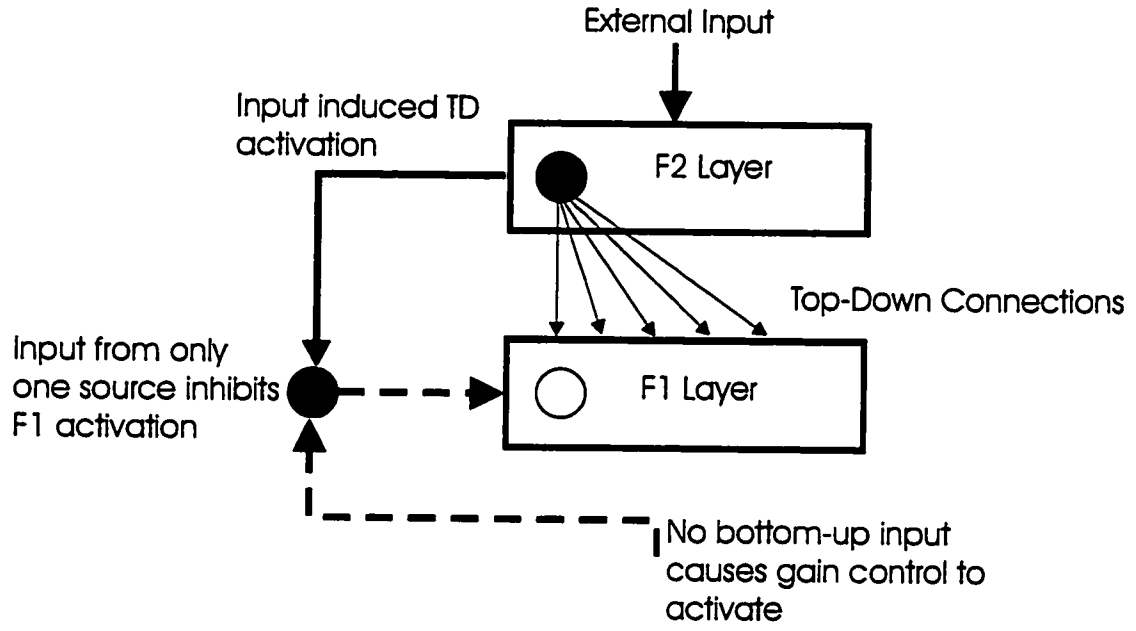


Figure 2.2: Preventing F1 activation without an input pattern present. External input to F2 causes activation to form and passes this onto F1. There is no activation sent to the attentional subsystem from the input pattern. The lack of a second attentional activation forces F1 to remain inactive and resist the top-down stimulation.

the input is correctly classified. To do this the classification field produces a new activation pattern across the input field by filtering the activation value through the top-down weight matrix. When this occurs, the gain control subsystem inhibits the non-specific pattern to the input field and maintains only two inputs, one from the input pattern and one from the classification layer filtered through the top-down weights. The similarity between this new pattern and the original input pattern is compared to determine if the classification layer has assigned the input pattern to the appropriate group. Similarity between the input and classification patterns is measured as a ratio of correctness and compared against a vigilance parameter. If the ratio is less than the vigilance then the pattern is suitably classified, if not then the pattern is misclassified. If the pattern is correctly classified then the top-down and bottom-up weight values are updated thereby training the network to more readily create the classification from the input pattern. If the patterns do not match then the orientation subsystem is activated. This causes activation within the selected classification element to be suppressed for an extended period of time. The input pattern is then re-applied to the input field. The activation pattern which this invokes across the classification layer can no longer effect the processing elements which the attentional subsystem is suppressing. Once an appropriate classification has been reached or the input pattern cannot be classified into any of the existing categories, and no new categories exist, then the orienting subsystem stops suppressing the classification elements. If all of the elements in the classification layer are suppressed by the orienting subsystem and no correct classification occurs, then that input pattern cannot be learned without using a network which has a larger number of classification elements.

2.3.1 Stability Plasticity Dilemma

The ability to learn new patterns until the capacity of the network has been reached introduces a problem known as the stability-plasticity dilemma. Simply stated, this is the ability for a network to remain adaptive (plastic) given unique inputs but remain stable in response to spurious inputs. A network which cannot remain plastic (too stable) will not

learn new patterns and one which is too plastic will overwrite existing pattern encodings.

ART networks are sensitive to *novelty* and are therefore capable of distinguishing between familiar or expected events and unfamiliar or unexpected events. The attentional and orienting subsystems control the effects which familiar and unfamiliar events have upon the network. Familiar events are processed by the attentional subsystem which constructs and adjusts the internal representation for the input patterns. The orienting subsystem is responsible for identifying if patterns are familiar to the network and for the creation of new categories for patterns which are unfamiliar.

2.3.2 The 2/3 Rule

The 2/3 rule is used as a barrier to prevent self activation in the ART networks. Input nodes can only become active if they receive stimulation from two of a possible three sources, (see figure 2.2). The three sources of input field activation are, the input pattern, the classification layer, and from the gain control in the attentional subsystem. If activation were allowed from only one source then the input field could be activated solely by top-down expectation values from the classification layer. This would cause an input pattern to register when one is not presented to the network. In response to this the new and false input would be filtered through the bottom-up weights and produce an activation in the classification layer which would cause learning to occur when no input is present. The attentional subsystem is responsible for implementing the 2/3 rule.

2.3.3 Fast and Slow Learning

Slow and fast learning in ART use identical network architectures, only the update rule for the top-down and bottom-up weight matrices are modified. Slow learning adjusts the weight matrices iteratively in response to repeated input patterns so weights will eventually settle upon a correct representation. Fast learning allows the weights to reach their asymptotic value during the presentation of each input pattern.

Due to the immediate method of encoding involved with fast learning there are several features of the network which can be identified; learning self-stabilizes in a finite number of learning trials, after learning has stabilized all input patterns have either direct access to a classification node or cannot be learned, top-down templates either remain the same size or are reduced in size through presentation of patterns, and that categorization nodes may exist after training which are not accessed by any of the input patterns [HGH94].

2.4 ART 1

The following sections describe in detail the classification process which occurs when a pattern is presented to inputs of an ART 1 network. The different levels of activation which appear in the network throughout the classification are illustrated in figures 2.3 and 2.4. Note that variables using a subscript i represent activations in field one (F1) and variables with a subscript j represent activations in field two (F2). There are M processing elements in F1 and N elements in F2. Subscripts referring to weights are given with the source of the activation as the first subscript and the destination of the product of the activation and the weight value as the second subscript. ART 1 can process only binary input patterns, so all input patterns I must be vectors of 0 and 1.

2.4.1 Input Processing

The activation of each x_i PE in F1 is calculated using the equation:

$$\dot{x}_i = -x_i + (1 - Ax_i)(I_i + DV_i + BG) - (B + Cx_i) \tag{2.2}$$

where A, B, C, and D are constants all greater or equal to zero and $max(D, 1) < B < D+1$. The V_i term is net input value from F2 to F1 unit i and is calculated using:

$$V_i = \sum_{j=1}^{N} u_j z_{ji} \tag{2.3}$$

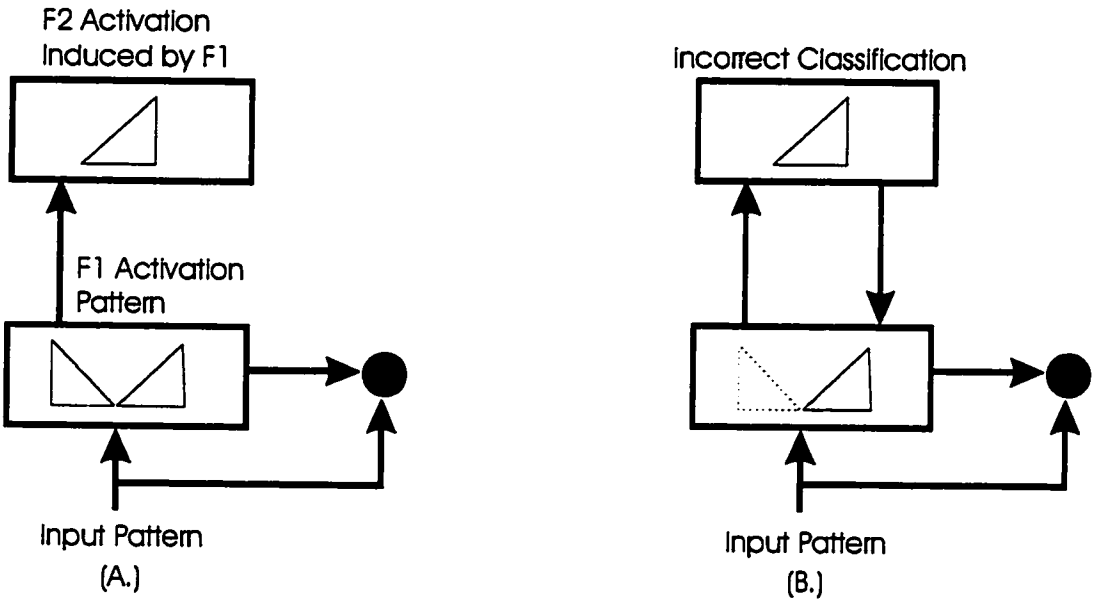


Figure 2.3: Activation patterns in ART 1. (a.) The input pattern forms a pattern across F1 which is categorized by F2 and activates the orienting subsystem. (b.) The activated node in F2 causes a pattern to form over F1 which represents the exemplar which is associated with that node.

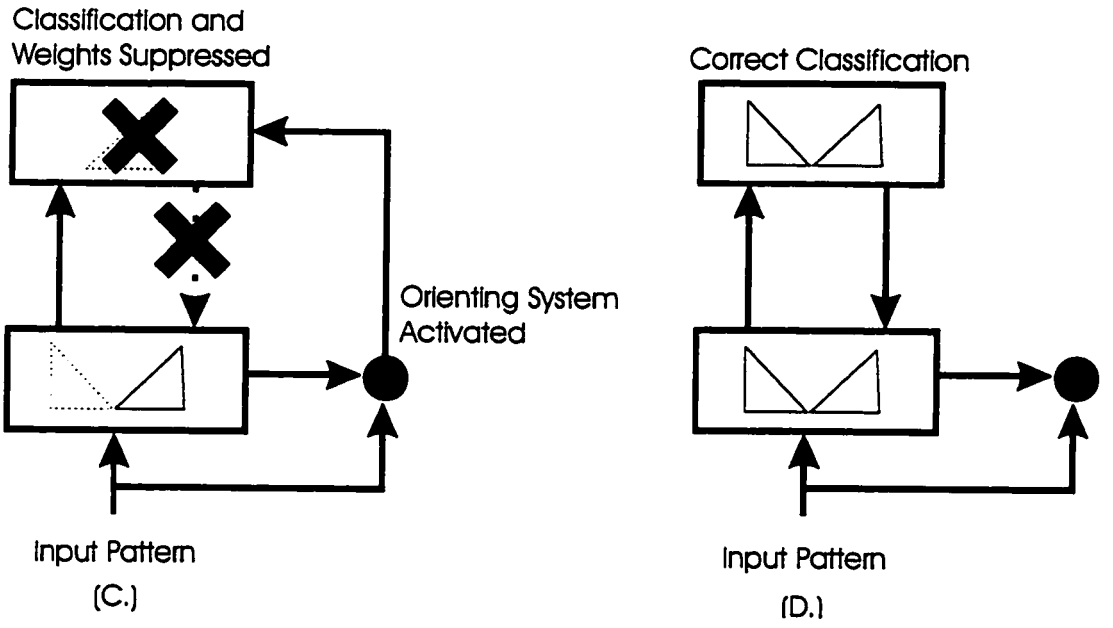


Figure 2.4: (c.) If the classification pattern is not sufficiently similar to the input pattern then the active F2 node is suppressed by the orienting subsystem. (d.) A new classification forms over F2 which correctly categorizes the input.

where u_j is the activation output from F2 and z_{ji} are the top-down weights from F2 to F1. The G value in equation 2.2 is the output of the gain control subsystem which can be described as:

$$G = \begin{cases} 1 & \text{if } I \neq 0 \text{ and } U = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

where I is the number of active inputs in the input pattern and U is the number of non-zero outputs from F2. If there is an input pattern present and F2 is not producing an output then G is 1.

The initial activation equation for F1 units, when no input pattern is present and F2 in inactive simplifies to:

$$\dot{x_i} = -x_i - (B + Cx_i) \tag{2.5}$$

When the activation in these nodes settle to an equilibrium ($\dot{x_i} = 0$) the nodes are held to a negative activity level:

$$x_i = \frac{-B}{1 + C} \tag{2.6}$$

Once an input pattern is applied to the F1 layer the gain control value (equation 2.4) becomes active and changes to a one value. This causes equation 2.2 to become:

$$\dot{x_i} = -x_i + (1 - Ax_i)(I_i + BG) - (B + Cx_i) \tag{2.7}$$

and the equilibrium activity of the F1 nodes to become:

$$x_i = \frac{I_i}{1 + A(I_i + B) + C} \tag{2.8}$$

Units which receive non-zero inputs from I generate activity values which are greater than zero. Units which receive inputs which are equal to zero have their activity level raised to zero through non-specific excitation from G.

The output vector for F1 is calculated simply using:

$$s_i = h(x_i) = \begin{cases} 1 & x_i > 0 \\ 0 & x_i \leq 0 \end{cases} \tag{2.9}$$

A certain amount of simplification to these calculations can be achieved when this method is simulated in software. The process of calculating F1 activation at the initial input stage can be reduced to activating F1 units which receive non-zero I_i inputs and setting all other F1 activations to zero. In a hardware implementation of the network or when inputs to F1 occur from F2 this will not be possible.

2.4.2 F2 Activation

Activation for each element in the F2 layer are calculated using the sum of products from the weighed connections (z_{ij} leading to each element and the activations (s_i) output from F1. The activation for each F2 processing element t_j is calculated using:

$$t_j = \sum_{i=1}^{M} s_i z_{ij} \tag{2.10}$$

The F2 layer is competitive and allows only one node to remain active. The node with the largest activation value has its activation set equal to one, the rest have their activation set equal to zero, (see figures 2.3 and 2.4). The output vector U for F2 is therefore set equal to:

$$u_i = \begin{cases} 1 & T_j = max(T_k) \forall k \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

The node which remains active represents the classification initially chosen for the input pattern. This can change if the network already has classified other patterns using this node which are not sufficiently similar to the new input pattern.

2.4.3 Top-Down Processing

The purpose of the top-down pass is to compare the network's representation of the pattern associated with the winning node in F2 to the input pattern. To do this a pattern of activation across F1 is generated using the active F2 node after classification and the top-down weight matrix connecting this active F2 node to F1, (figures 2.3 and 2.4).

After the winning F2 node has been selected, it creates a pattern of activation across F1 calculated using equation 2.3. This top-down activation affects both the attentional gain control subsystem and the F1 field. The gain control system described in equation 2.4 shuts off and the activation equation 2.2 for F1 becomes:

$$\dot{x_i} = -x_i + (1 - Ax_i)(I_i + DV_i) - (B + Cx_i) \tag{2.12}$$

where the equilibrium value for x_i is:

$$x_i = \frac{I_i + DV_i - B}{1 + A(I_i + DV_i) + C} \tag{2.13}$$

The value which x_i stabilizes upon depends on the relative sizes of the input values I_i and V_i and of the constants. Analysis shows that the optimum values for B and D obey the following:

$$max(D, 1) < B < D + 1 \tag{2.14}$$

to implement the 2/3 rule successfully.

2.4.4 Orienting Subsystem and Vigilance

After a new pattern of activation S_i has been generated across F1, it is compared with the input pattern I_i. The pattern S_i represents an exemplar for the pattern associated with the currently active F2 unit and the original input pattern I_i. A degree of match is calculated between these two values through the use of a similarity measure. The similarity is calculated as:

$$Sim = \frac{|S|}{|I|} = \frac{\sum_{j=1}^{M} s_j}{\sum_{j=1}^{M} i_j} \tag{2.15}$$

This value is the percent of the active nodes created by the exemplar pattern which correctly match the active input pattern.

If this percentage is sufficiently large then an appropriate match has occurred and the top-down and bottom-up weight matrices can be adjusted to more easily facilitate this matching in later presentation. If the percentage is too small then the matching between

the exemplar and created pattern is not appropriate and the active F2 node is suppressed by the orienting subsystem. As soon as the active F2 node is suppressed the F1 layer is allowed to pass activation values to F2 layer which again competes to identify the largest winning node with the largest activation. The exemplar associated with the new winning F2 node is measured against the input pattern. This cycle involving a classification in the F2 layer and comparison of exemplar to the input pattern continues until a sufficient match occurs between the two or until no nodes remain in F2 which are still active. If all of the F2 nodes have been deactivated then there is no sufficient match between the categories learned by the network and the current input pattern. In a network with F2 nodes which have not had a pattern associated with them (no weight changes have occurred in relation to that F2 node) then those nodes will be recruited when the input pattern does not match any of the existing F2 categories.

The parameter which the similarity value Sim is compared against to determine the similarity between the exemplar S and the input I is called the vigilance. A high vigilance requires very close matching between the exemplar and the input patterns and a low value allows more dissimilar patterns to be clustered together. Higher values produce classifications of many categories each containing only a few or a single pattern. Low vigilance values produce fewer categories which contain many different patterns which are relatively similar.

2.4.5 Weight Adjustments

Adjusting the weight matrices allows the bottom-up weights to more rapidly group input patterns and the top-down weights to store the exemplar patterns. After the network has stabilized the presentation of a previously seen input pattern will be able to *directly access* the F2 node to which it is associated. Misclassification resulting in the suppression of F2 nodes will occur infrequently and classification will take place rapidly.

The slow learning formula for the bottom-up weight adjustments is:

$$\dot{z_{ij}} = \begin{cases} K[(1 - z_{ij})L - z_{ij}(|S| - 1)] & \text{if } v_i \text{ and } v_j \text{ are active} \\ -K[z_{ij}|S|] & \text{if } v_i \text{ is inactive and } v_j \text{ is active} \\ 0 & \text{if } v_j \text{ is inactive} \end{cases} \tag{2.16}$$

where K and L are constants which control the rate of learning and $|S|$ is equal to the count of active F1 nodes. The fast learning equation is:

$$z_{ij} = \begin{cases} \frac{L}{L-1+|S|} & \text{if } v_i \text{ is active} \\ 0 & \text{if } v_i \text{ is inactive} \end{cases} \tag{2.17}$$

where L is a constant greater than one.

The top-down slow learning equation is:

$$\dot{z_{ji}} = \begin{cases} -z_{ji} + 1 & \text{if } v_i \text{ and } v_j \text{ are active} \\ -z_{ji} & \text{if } v_i \text{ is inactive and } v_j \text{ is active} \\ 0 & v_i \text{ and } v_j \text{ are inactive} \end{cases} \tag{2.18}$$

and the top-down fast learning equation is:

$$z_{ji} = \begin{cases} 1 & \text{if } v_i \text{ is active} \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

2.5 Fuzzy Art

Fuzzy ART incorporates fuzzy set theory into an ART 1 network. It is also an autoassociative, two layer, pattern classification network. The traditional set theoretic operators are replaced with fuzzy operators [CGR94]. This allows the network to form stable categories in response to analogue input patterns. Several changes are required to modify the ART 1 network to work with fuzzy sets, the most significant being the method of choosing a classification node in the F2 field. The processing element j with the largest activation T_j based upon the following equation is the category which best classifies the input pattern I.

$$T_j = \frac{|I| \wedge w_j}{\alpha + |w_j|} \tag{2.20}$$

The variable w_j represents all weights attached to node j, the $\wedge$ is the fuzzy MIN operator, α is a constant. In addition to the fuzzy operator, input patterns are normalized either through preprocessing or through complement coding. Normalizing input pattern a simply involves setting $a_i = \frac{a_i}{|a|}$ where $|a| = \sum_i^M a_i$. Compliment coding doubles the size of the input pattern by adding pattern elements which are the compliment of the original pattern. For example, given input pattern $a = (a_1, a_2, ..., a_n)$ the complement coded version of $a_c = (a_1, a_2, ..., a_n, 1-a_1, 1-a_2, ..., 1-a_n)$. Either method ensures that the sum of outputs from F1 will not exceed n which is required for the use of fuzzy set theory. The network differs from ART 1 in that it uses a single weight layer between FA and FB instead of separate top-down and bottom-up layers.

The algorithm describing the ART network is usually simplified when the systems are implemented using digital computers. The following sections describe the models used for processing in this work (from [CR93]).

Each fuzzy ART subsystem includes a field, F_0, of nodes that represent a current input vector; a field, F_1, that receives both bottom-up input from F_0 and top-down input from a field, F_2, that represents the active code, or category. The F_0 activity vector is denoted $\mathbf{I} = (I_1, I_2, ..., I_M)$, with each component I_i in the interval $[0,1](i = 1, ..., M)$. The F_1 activity vector is denoted $\mathbf{x} = (x_1, x_2, ..., x_M)$ and the F_2 activity vector is denoted $\mathbf{y} = (y_1, y_2, ..., y_N)$. The number of nodes in each field is arbitrary.

Weight vector: Associated with each F_2 category node $j(j = 1, ..., N)$ is a vector $\mathbf{w}_j \equiv (w_{j1}, ..., w_{jM})$ of adaptive weights, or LTM traces. Initially, when each category is said to be *uncommitted*,

$$w_{j1}(0) = ... = w_{jM}(0) = 1. \tag{2.21}$$

After a category is selected for coding it becomes *committed*. Each LTM trace w_{ji} is monotonically nonincreasing through time and hence converges to a limit. The fuzzy ART weight vector $\mathbf{w}_j$ formally represents both the bottom-up and top-down weight vectors of ART 1.

Parameters: Fuzzy ART dynamics are determined by a choice parameter $\alpha > 0$; a

learning rate parameter $\beta \in [0,1]$; and a vigilance parameter $\rho \in [0,1]$.

Category choice: For each input $\mathbf{I}$ and F_2 node j, the *choice function* T_j is defined by:

$$T_j(\mathbf{I}) = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} \tag{2.22}$$

where fuzzy AND, or intersection, operator ($\wedge$) is defined by:

$$(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i) \tag{2.23}$$

and where the norm $|\cdot|$ is defined by:

$$|\mathbf{p}| = \sum_{i=1}^{M} |p_i| \tag{2.24}$$

for any M-dimensional vectors $\mathbf{p}$ and $\mathbf{q}$. For notational simplicity, $T_j(\mathbf{I})$ is written as T_j when the input $\mathbf{I}$ is fixed.

The system is said to make a *category choice* when at most one F_2 node can become active at a given time. The category choice is indexed by J, where:

$$T_J = \max\{T_j : j = 1...N\}. \tag{2.25}$$

If more than one T_j is maximal, the category j with the smallest index is chosen. In particular, nodes become committed in the order $j = 1, 2, 3....$ When the J^{th} category is chosen, $y_J = 1$; and $y_j = 0$ for $j \neq J$. In a choice system, the F_1 activity vector x is characterized by the equation:

$$\mathbf{x} = \begin{cases} \mathbf{I} & \text{if } F_2 \text{ is inactive} \\ \mathbf{I} \wedge \mathbf{w_J} & \text{if the } J^{th} \text{ node is active} \end{cases} \tag{2.26}$$

Resonance or reset: *Resonance* occurs if the *match function* $\frac{|\mathbf{I} \wedge \mathbf{w_J}|}{|\mathbf{I}|}$ of the chosen category J meets the vigilance criterion:

$$\frac{|\mathbf{I} \wedge \mathbf{w_J}|}{|\mathbf{I}|} \geq \rho \tag{2.27}$$

that is, when the J^{th} category is chosen, resonance occurs if

$$|\mathbf{x}| = |\mathbf{I} \wedge \mathbf{W_J}| \geq \rho|\mathbf{I}| \tag{2.28}$$

Learning then ensues, as defined below. *Mismatch reset* occurs if:

$$\frac{|\mathbf{I} \wedge \mathbf{w_J}|}{|\mathbf{I}|} < \rho \tag{2.29}$$

that is, when

$$|\mathbf{x}| = |\mathbf{I} \wedge \mathbf{w_J}| < \rho|\mathbf{I}| \tag{2.30}$$

the value of the choice function T_J is set to 0 for the duration of the input presentation, to prevent the persistent selection of the same category during search. A new index J is then chosen, by 2.25. The search process continues until the chosen J satisfies 2.27.

Learning: Once search ends, the weight vector $\mathbf{W_J}$ is updated according to the equation:

$$\mathbf{w}_J^{(new)} = \beta(\mathbf{I} \wedge \mathbf{w}_J^{(old)}) + (1 - \beta)\mathbf{w}_J^{(old)} \tag{2.31}$$

Fast learning corresponds to setting $\beta = 1.0$.

2.6 ART Map

ARTMAP is built using the ART 1 model and is a multi-layer heteroassociative network. The purpose of ARTMAP, or predictive ART, is to match pattern pairs from the ordered pairs built from sets (A, B). Each of two ART 1 networks is trained with one of the sets, A or B, such that the classification field groups the input patterns as would a ART 1 network. These networks are identified as ART_A and ART_B to indicate which set they have be trained with. The classification fields of ART_A and ART_B are connected to a each other by an Inter-ART Associative Memory, (figure 2.5). This connection allows the classification field of ART_A to induce an activation pattern in the classification layer of the ART_B network. Once the classification layer of ART_B becomes active it produces an

activation pattern in the input layer through the top-down weight matrix which is one of the elements from set B.

ARTMAP uses an adaptive control technique over the vigilance parameter. When a mismatch occurs between the desired and created patterns in the two classification fields the vigilance parameter is increased to correct the predictive error. This forces all pattern pairs to be classified correctly.

2.7 The Formal Avalanche

Another network architecture developed by Grossberg is the Formal Avalanche [Gro78][FS92]. The purpose of this network was to learn and recall arbitrary sequence of events. It consists of two layers of nodes joined by a layer of weighted connections. Each event at a given time t is associated with an event classification node, (see figure 2.6). When one of these nodes is activated it is processed through the weighted layer and creates an output pattern. Once a temporal event node has been activated at time t, it signals the next event node for time $t + \Delta t$ to become active. In this way a series of events is recalled through the sequential activation of temporal event nodes.

The important aspect of this architecture to the development of a sequence processing network is its use of lateral connections in the classification layer to represent a sequence of events. The avalanche network makes no attempt to minimize the size of the representation. A mechanism similar to this, but with a more compact representation is developed as part of this work for sequences storage and recall.

2.8 Self-Organizing Maps

The Self-Organizing Map (SOM) developed by Teuvo Kohonen [Koh89] is a two layer competitive network which classifies input vectors. Patterns presented to the network are filtered through a layer of weighted connections and form a classification. Training a SOM results

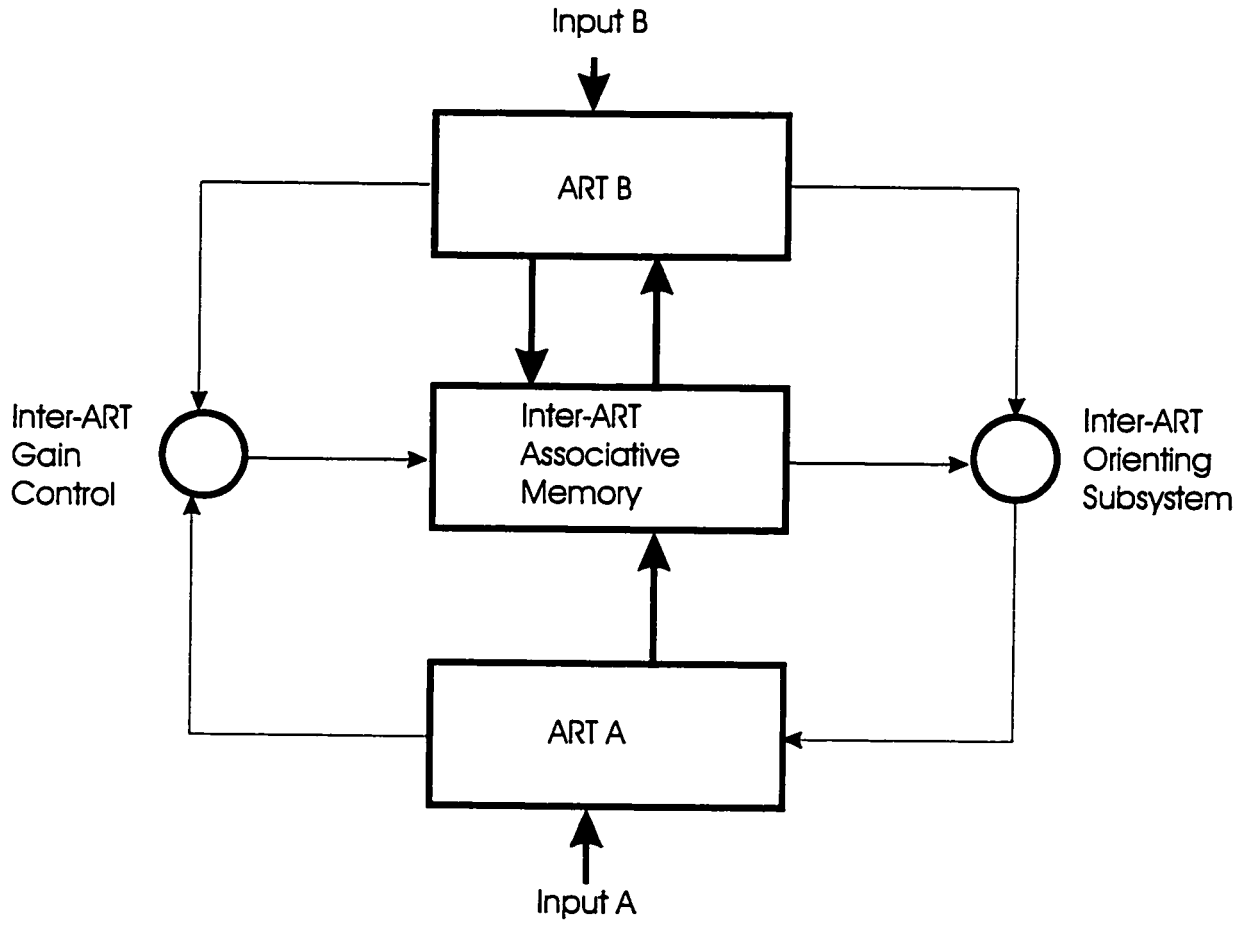


Figure 2.5: ART Map consists of two ART 1 networks joined by an associative memory and two subsystems.

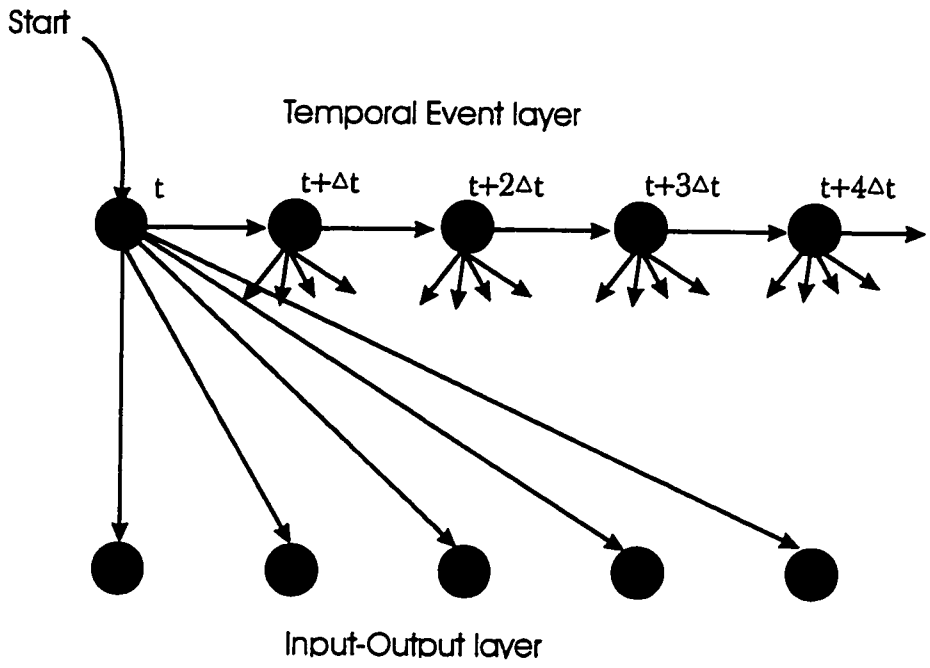


Figure 2.6: Formal avalanche network.

in a *topology-preserving map* of input data to classification nodes. This results in similar input patterns activating nodes which are physically near to each other in the classification layer. This differs from *random maps* such as ART, in which the classification nodes for similar input patterns do not necessarily appear near each other in the network.

The classification layer is generally a one or two dimensional array of nodes in which each node has its own weight matrix connected to the input layer. The initial value of these weights is randomly assigned. A series of lateral connections between the classification nodes is also part of the SOM (see figure 2.7).

The activation equation for clustering node y_i is:

$$\dot{y_i} = -r_i(y_i) + \sum_{j=1}^{n} x_j w_{ji} + \sum_{k=1}^{m} z_{ki} y_k \tag{2.32}$$

where $r_i(y_i)$ is a loss term which, for this work, is simply set to $r_i(y_i) = y_i$. The second term represents the input pattern from node x_j filtered through the weighted layer w_{ji} to node

y_i where n is the number of input nodes. The third term represents the lateral connection z_{ki} in the classification layer which is used to filter the activation from classification nodes y_k and transfer the resultant activation to node y_i.

Once a pattern of activation has been formed in the categorization layer then a winning node (category) is chosen using:

$$\|\mathbf{x} - \mathbf{w}_c\| = \min_i\{\|\mathbf{x} - \mathbf{w}_i\|\} \tag{2.33}$$

where $\mathbf{x}$ represents an input vector, $\mathbf{w}_i$ is the weight vector associated with classification node i, and c is the winning classification unit.

Training in a SOM is accomplished similarly to other networks, through adjusting the weight matrix. A key difference between the SOM and most other training algorithms is its method of adjusting weights associated with nodes other than those attached to the winning classification node. Nodes physically near to the winning node also have their weight matrices adjusted. It is through this neighborhood approach to updating weights that the network achieves its topology-preserving behavior. A neighborhood of nodes is usually defined as a group surrounding the winning node. Various functions are used to describe the neighborhood, but they generally simplify to include nodes which are physically near the winning node in each dimension of the categorization layer. The training process adjusts the weights using the following:

$$\mathbf{w}_i(t+1) = \begin{cases} \mathbf{w}_i(t) + \alpha(t)(\mathbf{x} - \mathbf{w}_i(t)) & i \in N_c \\ 0 & \text{otherwise} \end{cases} \tag{2.34}$$

where $\mathbf{w}_i$ is the weight matrix associated with selected classification node i, $\mathbf{x}$ is the current input vector, and N_c is a set of nodes including the winning node and those nodes physically near the winning node which are to have their weight values adjusted. The $\alpha(t)$ term is used to control the rate of change in weight values as training progresses. Initially the physical neighborhood of nodes which have their weights adjusted along with the winning node is relatively large. As training progresses the list of nodes which are updated in addition to the winner is gradually decreased until only those weights leading to the winning node are

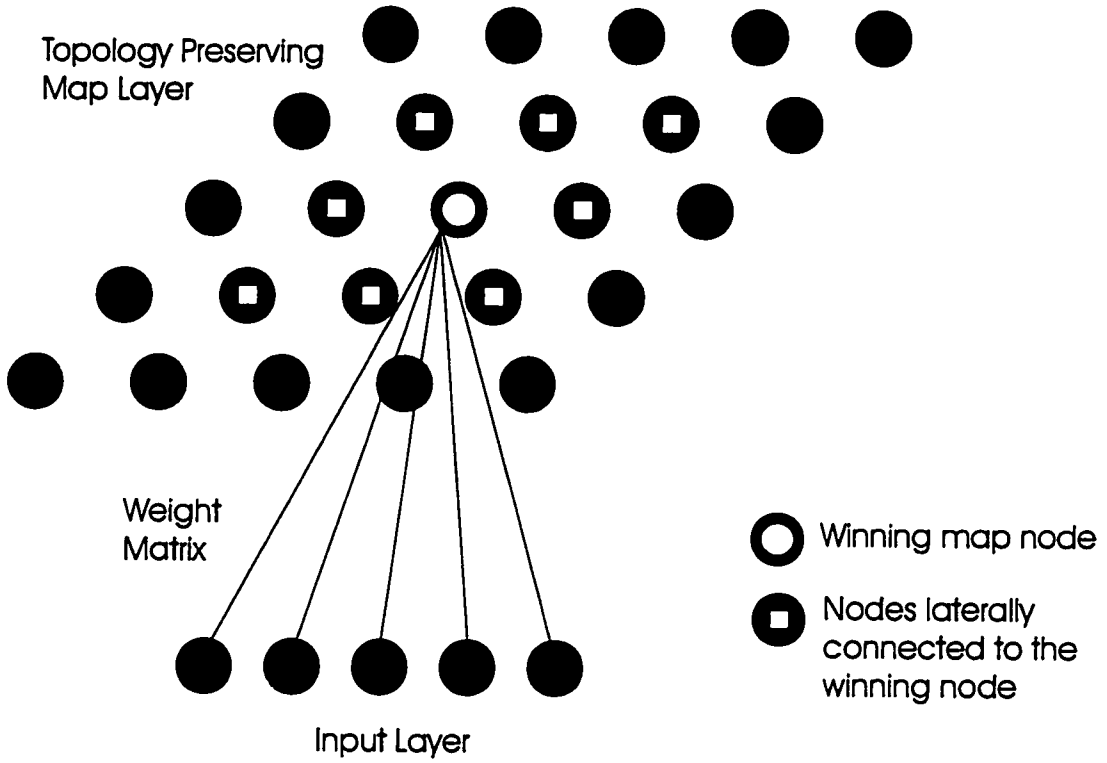


Figure 2.7: Self-Organizing Map with a two-dimensional classification layer.

adjusted. This reduction in the size of the training regions allows the network to initially form large regions of similarity which gradually divide into smaller and more distinct areas as training progresses. Ultimately each classification node is associated with a single input pattern and adjacent nodes are matched to similar, but not identical, input patterns.

2.9 Existing Neural Methods for Temporal Processing

Sequence analysis, for the purpose of categorization or prediction, has been performed using many methods. In the past ten years, neural network methods have been used more frequently. Several of the more prominent neural methods for sequence analysis are ex-

amined here. Many existing temporal neural network methods use variations of the Back-Propagation training algorithm.

2.9.1 Windowed Methods

The methods used to expand traditional Back-Propagation algorithms for use with temporal information all have one feature in common. They attempt to take temporal events and give them a spatial dimension within the artificial neural network (ANN). One or more new layers of nodes are added to the network in which a temporal context is stored.

One of the earlier methods for representing temporal events to an ANN was demonstrated in the NETtalk experiments of Sejnowski and Rosenburg [SR86]. The method involves the presentation of several temporal events simultaneously to a traditional BP network. Each event occupies one "window" within the input layer which represents a particular time slice. A flow of events is modeled by shifting each event to the next timing window, (see figure 2.8). This method acts very much like a shift register moving through a string of n sequentially occurring events. It requires a number of input nodes equal to $n \times p$ where n is the number of temporal windows and p is the number of elements within one window of time. The network is trained to recognize a group of temporal events as a single event.

There are several disadvantages to this representation, one of which is that the number of input windows must be the size of the longest input sequence which is to be stored in the network. This is wasteful if the input sequences are not all of the same length. Sequences which are shorter than the longest must bear the computational burden of training inputs which have no value. A second disadvantage to this method is its dependence on positioning of the input. A pattern that is familiar to the network which has been offset by one time step will likely not be classified similarly to the original pattern.

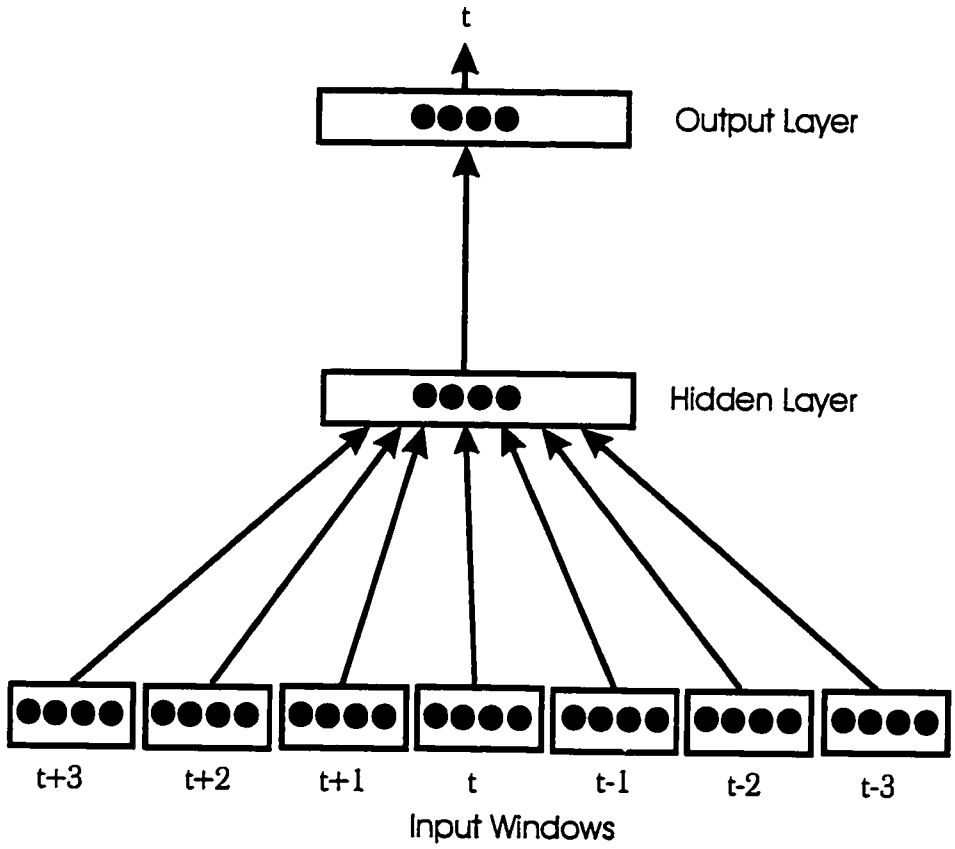


Figure 2.8: NETtalk Architecture. Center window (time t) is the ASCII character to associate phoneme with, and other times create a context.

2.9.2 Back-Propagation Recurrent Temporal Methods

An alternative to the windowed sequence of inputs is to maintain a memory in the network of historical events. This allows the ANN to provide a context for the current inputs based upon its previous patterns. Several models have been suggested for this sort of network, each attempts to perform the same function in a different manner.

The model which Jordan [Elm90] developed makes use of a new layer of context nodes which receive their input from the network's output and pass their own output to the hidden layer (see figure 2.9). The output from the network is used to help shape the ANN's internal representation of the sequence. The new processing elements, called memory, contain feedback loops which allow information from more than one previous state to be blended back into the context nodes. The decay of these feedback loops make it difficult for this network to utilize information that occurred early in a sequence to modify later events.

A variation on this idea was proposed by Elman [Elm90] who used recurrent connections from the hidden nodes, instead of the output nodes, to generate a context for the network, (see figure 2.10). Processing from the input to the hidden units proceeds as normal. Outputs from the hidden layer are passed to the output layer as they would in a traditional BP network, but they are also passed to a recurrent layer. Connections leading to recurrent units are fixed to 1.0 and do not change. One weighted connection leads from each hidden node to one recurrent node. This results in the hidden unit values being copied to the recurrent nodes and implies that there must be an equal number of nodes in each layer.

Elman states that the context nodes provide an internal representation of time. The hidden units use this context and the inputs to produce the desired output. The internal representations which are developed are therefore sensitive to sequential events. These representations are task and input dependent and may not have a literal mapping to the problem.

Several conclusions are drawn from Elman's work:

- Problems can change in nature when cast in a temporal form. Changes in the state

of the network can be used as information in finding a solution to a normally non-temporal problem.

- The size of the error for various input patterns and network states can be used as a measure of structure for the data.

- Increasing sequential dependencies can make a problem easier if they are structured properly. If the sequences follow some regular structure it can assist the network during training.

- Time and memory representation are highly task dependent. The representation which each element in the sequence takes on within the network is dependent on the ANN's encoding of the problem. The structure created by the network within the bounds of the physical architecture will be different for each problem.

- The internal representations are not necessarily unstructured. They may have a hierarchical structure which is determined by the network and does not require a priori knowledge of the problem to determine the network's architecture.

Elman concludes that this type of recurrent network has a "considerable" memory capacity and suggests that its inductive power can be used to discover structure and representation in tasks over time.

A variation on this style of network was used by Servan-Schreiber et. al. [SSCM88] to create a finite state machine (FSM) which they tested extensively. The network's architecture is identical to that used by Elman but the use of the ANN as a FSM is new, (see figure 2.12). It uses seven input nodes and a like number of output nodes. Two of the nodes are used to represent start and end conditions of sequences, and the remaining five nodes represent states in a finite state machine. The network is trained such that an input pattern at time t representing a state in the FSM will elicit an output pattern of a state at $t+1$ in the FSM. Several conclusions are drawn by the authors of this work. They suggest that the network initially learns to distinguish the input patterns independent of a temporal context,

and later uses the information stored in the context layer to distinguish different patterns. Two cases where recurrent networks will not recognize patterns correctly are identified: when two sequences contain identical subsequences involving the same prediction and when the number of hidden units is not sufficient to store redundant representations of similar patterns. They also note that adding recurrent connections to a BP network means that the network will no longer be guaranteed to perform a gradient descent in error space.

A variation on this network is described by Mozer [Moz89] with the context layer placed in-line with the input and hidden layers. The input to the context layer comes from the input nodes and from recurrent connections linking context units to themselves, (see figure 2.11). Added to this model is a decay value which modifies the recurrent connections in the context layer. The decay factor allows old information to fade over time. Strangely, after describing the faults found in "shift register" style representations of time, as that used in the NETtalk work, Mozer then includes them in his model. His suggestion is that this method allows the context layer to identify both sequences of elements and sequences of features of elements. He goes on to state that this model will not need such a large buffer as the "shift register" style network would require. Without the use of the input buffer this model is not capable of learning nonlinear interactions across time. Two problems identified with this network are an instability created by using a decay factor greater than one and the networks inability to learn long patterns with little information contained in each sequence element. Using a decay value larger than one leads to exponential growth of the activation value in the context units. This can lead to difficulty training with long patterns as the connections leading to the hidden layer must adapt to properly accept the large values.

Werbos suggests another approach for Back-propagation which uses two or more context layers [Wer90]. In this system the current input to the network can be combined with the previous inputs, hidden and output activation values. He calls this method back-propagation through time. This method is similar to that of Mozer in its use of buffered (shift register) input sequences, although they are not explicitly stated as such. In addition, this network allows connections to a node from any of the previous time period layers.

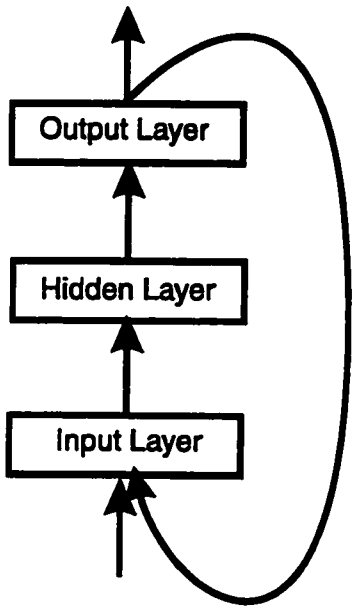


Figure 2.9: Recurrent BP architecture using Jordan's architecture.

All of the methods for recurrent BP examined have some degree of success. The most complete method appears to be that of Werbos which is a superset of the other methods. This technique would likely be slower learning than the others, due to its larger number of connections, but in return it provides a more general architecture. The main limitation to this architecture is its use of shift register style input buffers. The architectures of Elman and Servan-Schreiber et. el. do not use this style of input registers and are therefore potentially immune to some of the drawback inherent to such a system.

2.9.3 Temporal Methods using ART Networks

A method which calls upon a structure similar to that used in the recurrent Back-propagation methods, but which uses an ART network was proposed by Mannes and Dorffner [MD90]. This technique uses recurrent connections similar to those found in the Elman style of network, but connects them to the classification layer of an ART network. The activation

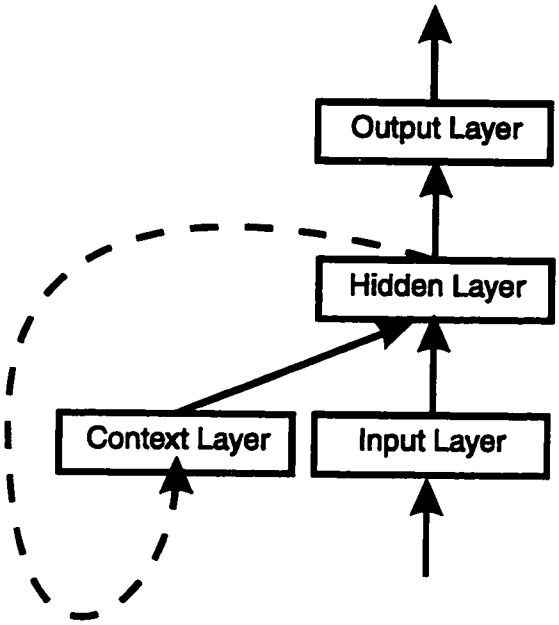


Figure 2.10: Recurrent BP architecture using Elman's architecture.

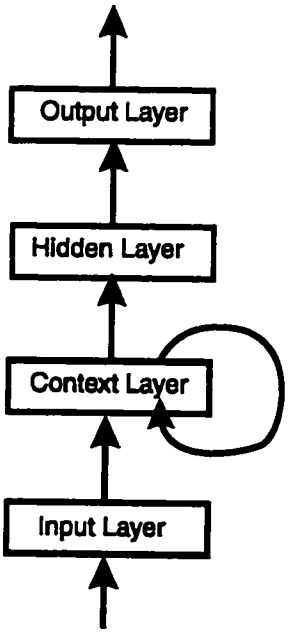


Figure 2.11: Recurrent BP architecture using Mozer's architecture.

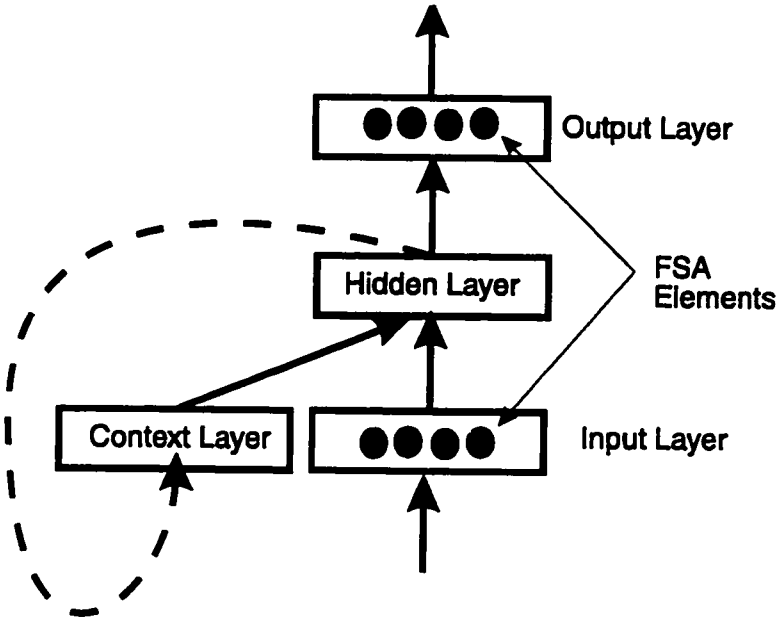


Figure 2.12: Recurrent BP architecture using a variation on Elman where the input and output layers act as the input and output of a FSM.

values in this new layer are copied directly from those in the classification layer, (figure 2.13). Weighted connections then process these patterns as they influence the next classification generated by the presentation of a new input pattern. The context layer (at time t) always contains a copy of the previous classification pattern (from time $t-1$). The authors mention that the network's response to repeated input patterns generally "worked well" but that the repetition cannot be repeated indefinitely because the classification layer will eventually settle upon a pattern which has occurred before or the input pattern will force a classification by overpowering the context units. In either case, this suggest that long sequences or repeated characters may be difficult to train using this architecture. The authors further mention that the capacity of the system depends primarily on the network size, but that there are limits to the length of sequences that can be learned. Learning in this system is influenced by the degree of sparseness of the data. Patterns which are "well filled" form more easily than sparse patterns. This can be influenced by choosing different learning rules based upon the overlap of the pattern sequences.

This system proved stable and was capable in most cases to reliably recognize and recreate sequences of arbitrary input patterns. When presented with sequences which exceeded the network's capacity it tended to pick sub-sequences which appear most frequently.

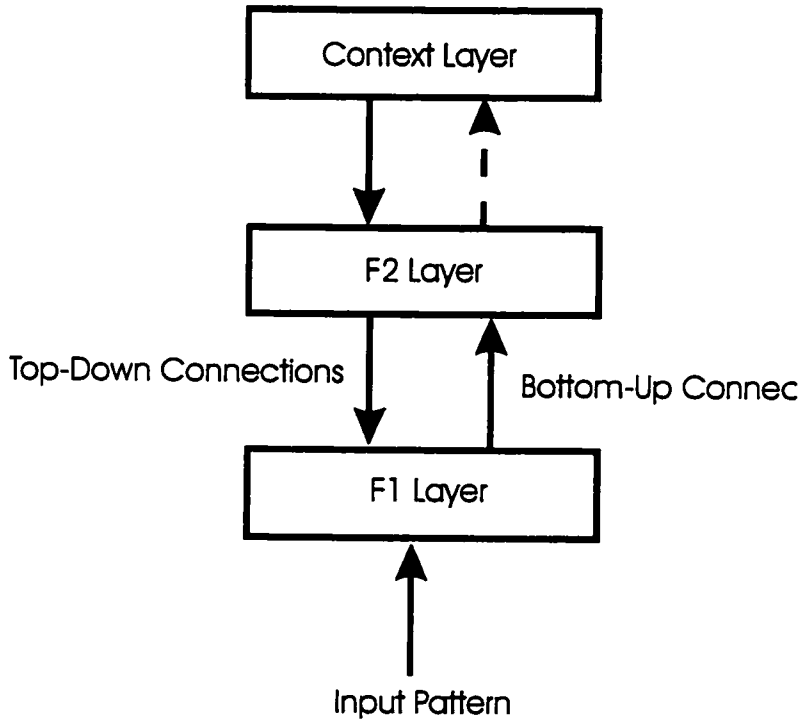


Figure 2.13: Temporal ART developed by Mannes and Dorffner. The bottom-up connections between F2 and the context layer act as a copy operation, not an adaptive filter.

Chapter 3

Network Architecture

3.1 A Temporal Neural Network

The network developed in this chapter uses a combination of existing ANN architectures and a newly developed sequence processing system. The networks which existed prior to this work are the Fuzzy ART and Self Organizing Map (SOM), both of which are used to categorize input-output patterns.

The Fuzzy ART network has several properties which make it a good choice for use as a component in a larger system. These include its ability to form stable categorizations, provide arbitrarily course or tight classifications, and scale well to large problems [Nig93]. The Fuzzy ART network is constructed in a modular fashion which facilitates the modification of its components as required when it is integrated into a larger system.

The second network used is the Self-Organizing Map. It was chosen because of its topology preserving structure. Unlike ART which does not readily provide a method for identifying similarity between clusters the SOM provides a means to identify similar clusters and uses this information as part of the sequential network. Both networks are examined in this work in an attempt to identify their characteristics when used in the sequential network.

The pattern clustering operations performed by the ART and SOM networks remain unchanged when they are placed within the proposed temporal network. The input processor need not be limited to these two networks, any method which categorizes input vectors can be used with the proposed architecture. However, it is necessary that the chosen technique allows for the identification of the relative similarity between categories.

3.2 The Processing Model

The processing model is the method for learning and recall of sequences which was developed as part of this work. The basic model is uncomplicated, quite accurate at storing and retrieving sequences, and has characteristics which make it amenable to operating within the framework of an artificial neural network.

The model uses a distance measure to indicate the similarity between the training and testing data set. Training simply involves loading the training data into a one dimensional array and initializing a corresponding distance array. These two arrays must be of the same size. Elements in each array with the same index value are related such that the distance value stored in location i indicates the similarity between the training and testing sequence at location i. Note that the distance value at i represents the similarity between the current input sequence when compared to the same number of training values where the last value in the sequence is element i. Figure 3.1 illustrates the data structures for this system.

Testing of this system involves comparing an input pattern to each value in the training set and adjusting the distance associated with each of these values. During testing the sections of the training set which are dissimilar to the testing values will have large distance values associated with them indicating that they are not a close match to the testing values. Areas of the training sequence which are similar to the testing values will have smaller distances. The algorithm for the process model is:

1. Read the training data into array Y of size n.

2. Initialize distance values in array D to θ. There are n values in D and θ is a value between 0 and 1.

3. Set $i = 1$.

4. Obtain a testing value x_i from the input sequence X.

5. Compare x_i to each value in $y_j \in Y$ and and adjust the distance value in each $d_j \in D$ using the following method:

$$d_j = \begin{cases} d_j - \alpha & \text{if } x_i = y_j \\ d_j - \beta & \text{if } |x_i - y_j| < \delta_1 \\ d_j & \text{if } \delta_1 \leq |x_i - y_j| \leq \delta_2 \\ d_j + \gamma & \text{if } |x_i - y_j| > \delta_2 \end{cases} \tag{3.1}$$

   where $\delta_1 < \delta_2$.

6. Move each value d_j to d_{j+1} for $j = 1...(n-1)$.

7. Increment i.

8. Return to step 4 until all of X has been presented or the desired accuracy has been reached.

The α, β, and γ values are the degree to which the distance values are changed based upon the similarity of training and testing values. The two δ values are thresholds used to identify regions of similarity between the sequence values. When the training y_i and testing x_i elements are identical then the distance is decreased by the value α. Relative similarity and difference between the two patterns is indicated by their difference being less than δ_1 and greater than δ_2 respectively. The most interesting part of this equation is the range between δ_1 and δ_2 in which the distance is not changed. This region is used to provide a range of pattern comparisons which are not modified if there is some degree of similarity. This allows a degree of local variation in the testing set which will not drastically alter the distance values. This is desirable as it allows some amount of noise to appear in the test data without it immediately effecting the network's recall.

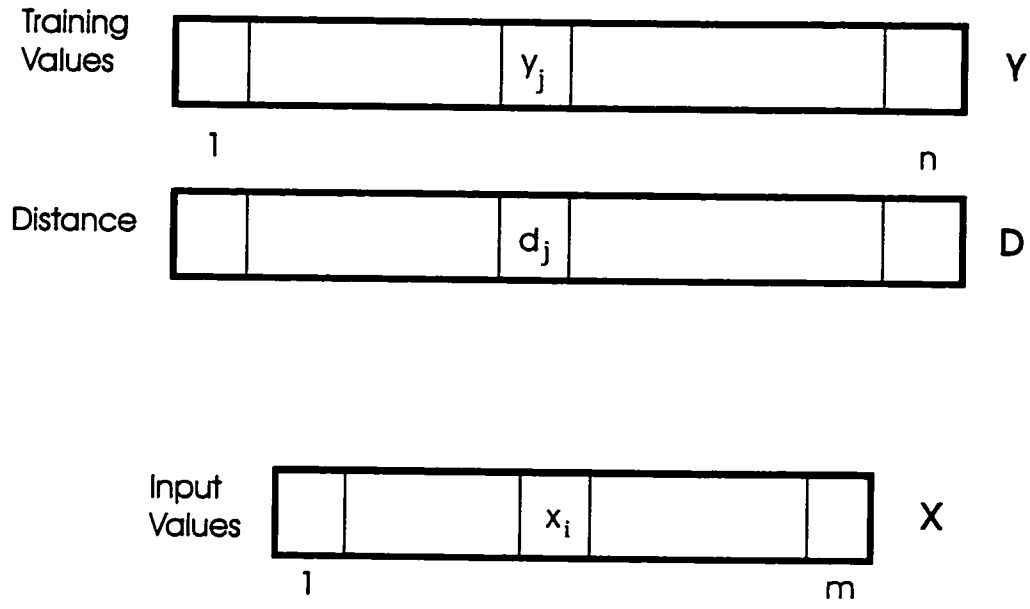


Figure 3.1: The Processing Model used to develop the sequential network.

There are several features of the processing model which make it a good candidate for development within a neural network framework. Its use of a large number of values which can be evaluated in parallel suggest that the model can take advantage of the parallel operations found in most ANNs. In addition, the simple training method should allow for a very fast, single pass, training network.

3.3 Translating the Processing Model to Operate in a Neural Network

Several changes to the processing model are made in order to execute it within a neural system. The two main difficulties in translating the processing model are its expectation of one-dimensional data and its method of advancing each distance measure one time step with each new input value. Converting the process model to work with multiple inputs is relatively simple and involves little more than the use of a multi-valued distance measure.

However, converting a neural architecture requires more effort.

To reduce multi-dimensional data into one-dimension the ART and SOM networks are used. Each of these networks reduces the dimensionality of the input and returns a category representing the multi-dimensional inputs. A pair of either ART or SOM networks are used to categorize and recall input and output values respectively. As a result, the system is no longer processing the raw data itself but is instead processing categories. The process model uses the similarity between input and training values to generate a distance measure for the two sequences. A variation of the process model which uses some relative distance between categories is developed for use within a neural architecture. The ART networks do not initially provide a technique for identifying similar categories, so one is developed. The neighborhood of the SOM allows for a group of similar categories to be easily identified and this information is used to provide the similarity measure required by the processing model.

As a side effect of using categorization methods on the inputs to this system it is possible to vary the degree to which similar values are grouped together. Categories containing large numbers of values are possible. In applications where precise mappings of inputs-output pairs is required this is not a very useful feature. In situation where larger ranges of values may be grouped together with no loss of functionality this provides a means to generate a smaller system by reducing the number of categories and thereby decrease the size of the network.

To simulate the mechanism in the process model which advances each distance measure one time step with each new input, a *time stamp* is assigned to each category generated by either the ART or SOM networks. This time stamp is used to identify the relative times when each classified event may occur and is used to activate the distance measures stored in the temporal network. The temporal network consists of one layer of nodes and two layers of weights between two ART or SOM classification networks.

3.4 Architecture of Temporal ART

The Temporal ART architecture involves two fuzzy ART networks for input and output classification, and an internal layer which builds an association between these two networks, (see figure 3.2). Input and output categorization layers are labelled F1 and F2 respectively. The clustering provided by each F2 layer creates the elements which will be sequenced. Input to F1 is provided as a sequence of events which are presented in order of appearance within the data set. No input buffering is used, only one sequence element is present at the network input F1 at any given time. When an input pattern is presented to the F1 layer of the first fuzzy ART it is contrast enhanced and the resultant binary pattern is stored in the F2 layer. Similar input patterns will appear as the same binary pattern in F2. The second fuzzy ART network is used to cluster the element which is the desired output given the pattern which is presented to the previous network. An association is then formed between the two F2 layers using a hidden layer X and two weight layers Wa and Wb.

The F2 field found in ART networks is a competitive layer in which a single processing element remains active after a classification. This is a reasonable method of operation when attempting to categorize one input pattern but it creates a significant limitation when applied to sequence classification. Due to the method of presenting only one sequence element to the network at any time it is possible to enter an ambiguous state where the network will be incapable of identifying which is the appropriate classification given the current inputs. An example of this can be seen with the sequences ABC and ABB. If the Temporal network has been trained with both patterns and is presented with the subsequence AB, there is not sufficient information to classify which sequence is being presented. The current input is a valid subsequence of both larger sequences so neither choice should be eliminated. If the sequence network acts as a competitive winner-take-all classifier then upon the presentation of the first sequence element it must make some categorization of the pattern. Due to the lack of information, this classification can be little more than a guess from a selection of possible sequences, all of which begin with the same element(s).

An alternative to the competitive classifier is to develop a distance based classifier. The

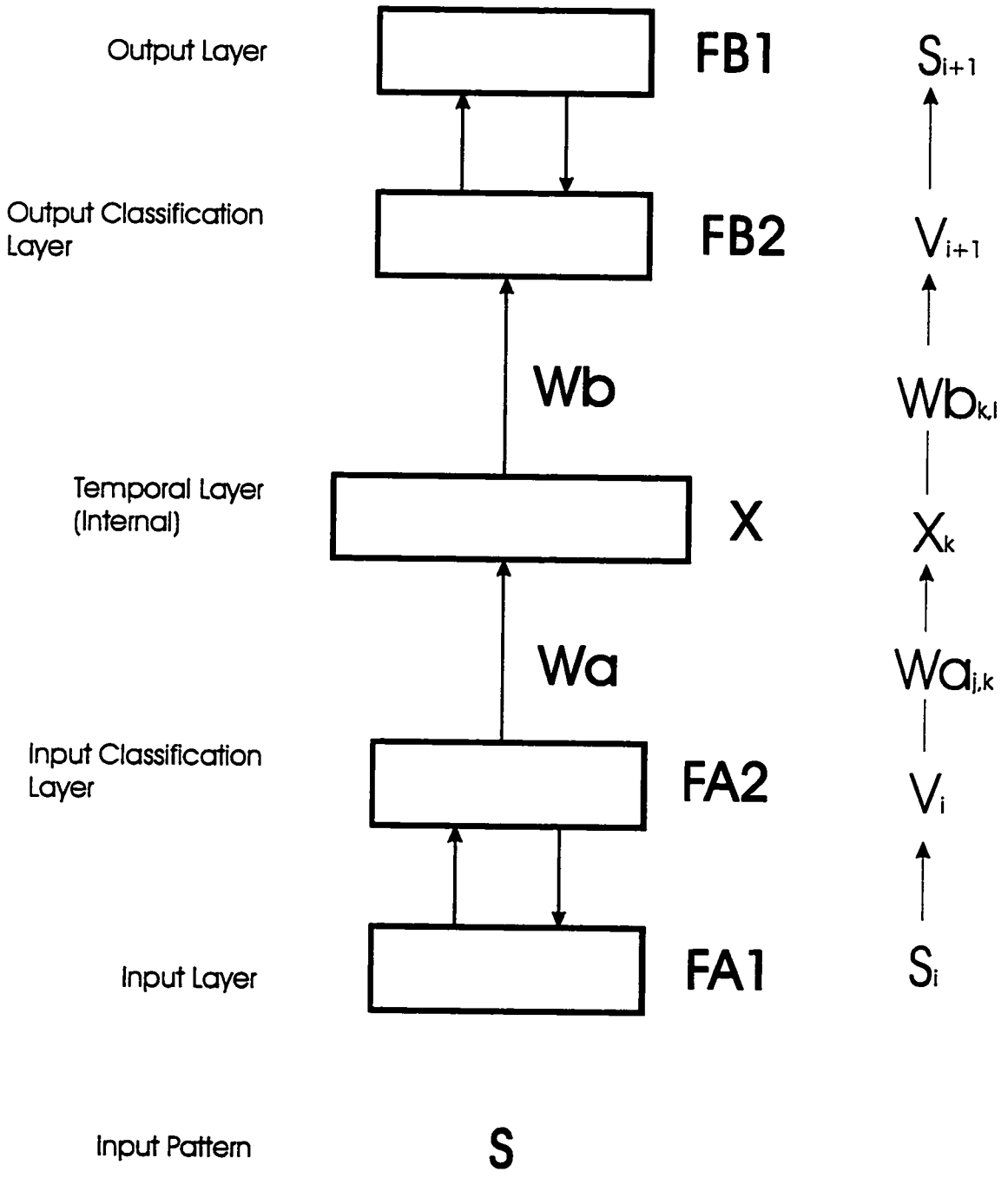


Figure 3.2: Names of processing layers, vector layers, and variable names.

basis of the competitive strategy is that a single class best represents the input pattern. With inputs leading to ambiguous classifications this cannot be true. If there is no clear classification which can be derived from the current inputs then several sequence elements must remain active. This allows all hidden nodes which are potentially correct to remain active until sufficient information is presented to make a correct classification. In the above example, the classification nodes in layer X which represent ABC and ABB would both remain active until the third sequence element (C or B) was presented to the network. At that time, the hidden node which classifies the incorrect sequence would have its activation suppressed to some degree and the correct classification node would achieve a higher activation. The algorithm suggested by this method is one in which the sequence network acts to classify the current input within the context of an input sequence. The network may receive inputs from two sources. An F2 field from one of the ART networks or the hidden field itself can cause activation to occur within the temporal sequencing network.

Inputs from F2 are the categorization which the current input pattern and its neighboring categories create. In general, only one F2 processing element indicates a categorization at any time. The process model when modified to operate in a neural network requires multiple categories to remain active which indicate similarities between the categorizations. Multiple activations in ART can be achieved through the use of a technique similar to *match tracking* [CGM92]. The purpose of match tracking was to raise the vigilance value to a point where resonance occurs in an ARTMAP network after an initial matching has failed. By raising the vigilance value, a better classification is formed in the network by narrowing the allowable range of matching patterns. By performing the opposite operation, decreasing the vigilance value, more categories will become active. These categories will have similar input patterns associated with them.

The sequence network receives inputs from one of the ART F2 field and has lateral connections internal to the field itself. Internal connections are required to allow the field to activate another sequence classification node in the sequence layer X once a currently active node cannot process a new input. When this occurs the first element of a new

sequence will be classified by the next available sequence node. The sequence field should then attempt to activate the next sequence classifier using this input and the current state of the field derived from the lateral connections.

3.5 Architecture of Temporal Self Organizing Map

The architecture of the SOM based temporal network is similar to the Fuzzy ART based system, but uses SOMs to classify input and output patterns. Both systems consist of two layer classifiers connected through a sequence processing layer.

For notational simplicity, a common notation is used to describe the major components in Fuzzy ART and SOM networks as seen in figure 3.2. The input and output networks are FA and FB respectively. Layers in each of these networks are labelled FA1, FA2, FB1, and FB2 where layers labelled with a 1 process input and those labelled 2 contain the classification nodes. Layer X processes the sequencing information.

The modifications to the ART architecture which are required to identify similar categories are not required in the SOM. The topology preserving nature of this network provides the required information through the placement of similar categories into physically adjacent classification nodes.

3.6 Algorithm Description

There are two parts to the algorithm used in this system, one for training and the other for recall. The training algorithm is quite simple which results in fast training. The recall algorithm is more complex, but facilitates easy modification.

The training method for the sequence network differs greatly from the processing model. To store the relative positions of the training elements, the order in which they occur is encoded in a pair of weight matrices which contain time-stamp values. These weight matrices Wa and Wb are processed through a temporal network X which associates the

input pattern to the output pattern. The input and output patterns are both processed through the categorization networks before they are given a temporal association. This provides the ability to use multidimensional sequences which was not possible with the simple processing model.

A sequence is defined as $S = (s_1, s_2, ..., s_n)$ where each s_i is $n = |s_i|$ bits long. The input sequences are $(s_1, s_2, ..., s_{n-1})$ and the output sequences are $(s_2, s_3, ..., s_n)$ which is used to identify ordered pairs of input-output elements $((s_1, s_2), (s_2, s_3), ..., (s_n, s_{n-1}))$. Each input-output pair is contrast enhanced to produce a vector consisting of a binary pattern where a single winning category represents a cluster of all similar input patterns. The input-output vectors can be represented as $V = ((v_1, v_2), (v_2, v_3), ..., (v_{n-1}, v_n))$ which corresponds to the ordered pairs of sequence elements. The categorization of s_i which creates v_i is performed by a pair of either Fuzzy ART or SOM networks.

The internal network between FA and FB builds an association between the two vector patterns (v_i, v_{i+1}). The association between the ordered pair of vectors is marked by a time stamp t_i where $\mathbf{T} = (t_1, t_2, ..., t_{n-1})$ and n is the number of elements in the training sequence. The time stamp values are stored in the weight layers of the sequence network and are processed through the internal layer of nodes during recall.

Each association is represented by two weighted connections $wa_{j,k}$ and $wb_{k,l}$ which lead from the input represented by the active i pattern v_i to the active output pattern v_{i+1}, (see figure 3.2). The set of weighted vectors for $wa_{j,k}$ and $wb_{k,l}$ are represented using Wa and Wb respectively. The active node in pattern v_i is represented by FA2 and the active v_{i+1} pattern is represented by FB2. The weight $wa_{j,k}$ links FA2 to an internal node x_k, the output of which is processed through $wb_{k,l}$ and connected to FB2. The initial values for Wa and Wb are zero which represents no association through the weight matrix. Multiple input nodes are connected to each x_k and each x_k is connected to multiple output nodes. The $wa_{j,k}$ and $wb_{k,l}$ weights are set to the same value t_i for a given pattern pair at time i. For a given input-output pair this means there will be a set of weights, one in each of Wa and Wb which joins the input classification node v_j, to the temporal layer node x_k, and

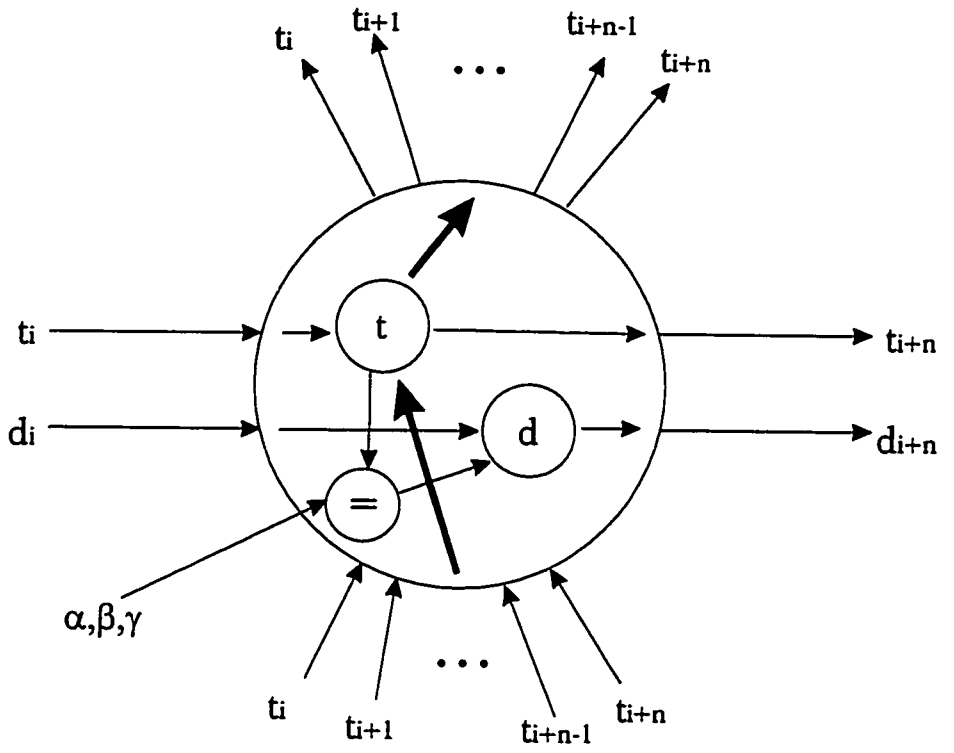


Figure 3.3: Inputs, outputs, and internal values in a sequence processing node.

from there to the output classification node v_l.

The values assigned to the weights are chosen such that $wa_{j,k}$ at time i is less than that at time $i+1$. The same holds true for $wb_{k,l}$. After training, there exist two of equally weighted values between each input-output vector pair. Figure 3.3 shows the contents of each sequence processing node in layer X. The inputs to the node include the weighted time values t_i to t_{i+n} from FA. A similar set of weighted values exit the node and lead to FB. Inputs also include a lateral connection t_i and a corresponding lateral distance value d_i from an earlier sequence node. Another lateral connection and distance values also leave the node and are passed to a later sequence node. The current modifier to the distance value, either α, β, or γ, is sent to the node and allowed to affect the internal distance value d only when an input time t_i is equal to the internal time value t.

A single sequencing node x_k from the set X is used to process input-output pairs until the $wa_{j,k}$ value at time i would be the same as at time $i+n$, or the $wb_{k,l}$ value at time i would be the same as at time $i+n$. This ensures that two time stamp values in **Wa** or **Wb** cannot be stored on the same weighted connection. Once the input-output patterns are categorized and two unused weights exist between them, the weights $wa_{j,k}$ and $wb_{k,l}$ are adjusted to represent the current input pattern. If the current $wa_{j,k}$ does not equal zero then it has already been assigned a value and k must be incremented in order to select a new internal node to store the association between v_i and x_k. The same holds true for adjusting $wb_{k,l}$, if the weight between x_k and v_{i+1} is already assigned, then k is incremented. This forces a new x_k node to be chosen when a weight leading either to or from it would be overwritten.

A set of lateral weights connects adjacent nodes in the temporal layer X. Each weight is set equal to the largest time-stamp value $wa_{j,k}$ which is coincident to the node k. These lateral connections are used to transfer the contents of node k to node $k+1$. When largest time-stamp value which contributes to node k becomes active, it creates an activation in the node equal to $wa_{j,k}$. This activation matches the lateral connection and is used to transfer the value stored in node k to $k+1$. This process forces the activation values stored in the temporal layer to be shifted to adjacent nodes as patterns are presented to the network.

To recreate the processing model it is necessary to store a distance value in each node of the temporal layer. This is in addition to the activation value (time-stamp) which is also stored in each of these nodes.

One of the benefits of this structure is its method of processing several sequential events in a single hidden node. This helps to reduce the size of the network by allowing several time stamps to use one processing unit. When forced to use a new internal processing node for each time stamp then the system operates similarly to the original processing model.

3.6.1 Training Algorithm

1. Train the input and output categorization networks with the input data. Either the

Fuzzy ART or SOM networks are used for this purpose. The temporal layer is not active at this point.

2. Set $k = 1$.

3. Activate the first temporal node x_1.

4. The index i in the training sequence is set equal to 1.

5. Set weight matrices Wa and Wb equal to zero.

6. Set the time-stamp $t = 1$.

7. Categorize input pattern s_i using the input network and s_{i+1} using the output network where $i = 1$ to $n - 1$ and n is the length of the training sequence. The input categorization appears node j and the output categorization is in node l.

8. If the weights $wa_{j,k}$ and $wb_{k,l}$ are both uncommitted (both have no time stamp associated with them) then set them equal to t.

   If either of these weights has a time-stamp associated with it:

   (a) Set the lateral weight between x_k and x_{k+1} equal to t.

   (b) Activate the next temporal node x_{k+1} by incrementing k.

   (c) Set the weights to this node $wa_{j,k}$ and $wb_{k,l}$ equal to t.

9. Increment t.

10. Move to the next input pattern i (increment i) until the $i = n - 1$.

11. Return to step 7.

3.7 Recall Algorithm

This system differs from the processing model in the way in which it compares the similarities of sequence elements. The processing model compares the raw data values. Due to the

categorization of inputs by either fuzzy ART or SOM networks, it is no longer possible to compare the data values so directly. Instead, a method is used which compares the time stamp values and produces similar results. The similarity of events stored in the temporal network cannot be measured in terms of time-stamp values. Similarity in time-stamps is only an indication of the ordering of events and does not provide any information regarding their relative similarity before categorization. To compare the similarity of values a method is developed which identifies groups of time-stamps based upon the similarity of the categories with which they are associated.

Recall in this system is performed in three stages. First an input pattern is presented to one of the categorization networks, it is then processed through the internal layer, and finally it proceeds through the second categorization network. The input is first contrast enhanced and transformed to a bit vector by the FA ART or SOM network. The bit vector is then passed through the weight matrix Wa which forms an activation in the internal layer X. The activation in x_k is set equal to $wa_{j,k}$ when $v_j = wa_{j,k}$. The activation x_k is then filtered through Wb to produce a categorization in $FB2$. This new binary vector is then processed by the FB network to produce an output pattern. Each sequence node x_i at time t uses the activation function:

$$x_k(t) = \begin{cases} w_{j,k} & \text{if } x_k(t-1) = w_{j,k} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

and the FB2 layer uses the activation function:

$$v_l = \begin{cases} w_{k,l} & \text{if } x_k(t) = w_{k,l} \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

Recall of patterns from the training set is quite simple due to the explicit nature with which the sequence patterns are encoded. If the training data is presented again, as it was during training, then the system will correctly predict the next element in all cases. If a subsequence of the training set is used then the network may require several elements to be input before it can identify which part of the sequence is being presented, but after this occurs the system will correctly recall all of the successive patterns.

If this system was intended to recall only patterns which it was trained with, then the described architecture would be sufficient. With data which were not trained upon, the network does not perform well. Its reliance upon the testing data appearing exactly as the training set is too restrictive and leads to poor modeling of the data. As is the case in the process model, a distance measure is used to indicate the similarity of an input sequence to the model of the training set.

The system generates a distance value based upon the training data and the current input sequence by relating the new data to the old. The associations built between the classification networks during training are used as exemplars during testing. The testing patterns $s_i, ..., s_{i+n}$ are matched to the sequence of exemplars stored in $wa_{j,k}$ which most closely model their structure. To achieve this, a distance measure is used to calculate the similarity between the active exemplars in X and the current sequence of inputs.

When an activation is formed in X it is likely that more than one x_k node will become active indicating that several exemplars are somewhat similar to the current input sequence. This multiple activation in X will continue throughout the presentation of input sequences. Maintaining multiple activation and distance values, each of which represent possible solutions and their relative similarity to the training sequences, is not common in neural models. There are several benefits to maintaining this information. It allows the users of the model to choose the criteria for selecting a winning category. Maintaining multiple solutions also creates a measure of the testing data against several sections of the training set instead of only the most similar matching. This allows testing sequences which are initially a poor match to the training data to improve their similarity measure with the presentation of more data.

The distance measure is modified substantially from the processing model due to the need to compare time-stamped categorizations instead of the raw data values. The method used to find similar input patterns in both Fuzzy ART and SOM networks involves relaxing the categorization criteria in the networks. If these criteria are relaxed in a controlled manner it is possible to identify similar categorizations which represent the similar groupings

of input patterns. It is then possible to compare the similarity of the categorization nodes instead of the similarity of the raw data to generate a distance measure. Some information is lost when using this technique. It is possible that relatively different categories could be identified as the most similar because there are no others which have a greater degree of similarity. This method measures for relative similarity of categories, it does not provide an indication of the distance between them.

To compare classification nodes for the purpose of generating a distance between them the network must either create a topology preserving map or allow for a technique which can extract the distance information. Creating topology preserving categorizations is not a feature of the Fuzzy ART architecture. To achieve this behavior the network algorithm requires some modification. The winning categorization node is normally chosen using a vigilance value which controls the degree of match between the input pattern and the categorization. By reducing the vigilance parameter after a category has been selected, it is possible to activate other categorization nodes which are similar to that which was first selected. The choice function which selects the winning category must also be modified to allow more than a single node to be active.

Match tracking in a Fuzzy ARTMAP network occurs when the following is true:

$$|\mathbf{x}^{ab}| < \rho_{ab}|\mathbf{y}^{b}| \tag{3.4}$$

The $|\mathbf{x}^{ab}|$ term represents the map field classification and $|\mathbf{y}^{b}|$ is the vector representing the output classification. The ρ_{ab} value is the vigilance parameter. If the map field classification does not match the output vector sufficiently well, the vigilance value ρ_a is increased until a new classification is found through either a better match or a newly created classification in FA. This forces a new activation in $\mathbf{x}^{ab}$ which can then be used to match the desired output vector. For this work the opposite effect is desired. A less focussed classification can be used to generate a larger number of active classification nodes. To achieve this, a reverse form of match tracking is developed which lowers the vigilance parameter to the point where some number of classification nodes become active. This operation will require a modification to the method of choosing winning categorization nodes in Fuzzy ART. The

traditional method for choosing a categorization in Fuzzy ART uses:

$$T_J = \max\{T_j : j = 1...N\} \tag{3.5}$$

and

$$T_j = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} \tag{3.6}$$

where $|\mathbf{I} \wedge \mathbf{w}_j|$ is the sum of the fuzzy-intersection of input pattern $\mathbf{I}$ and the weight matrix $\mathbf{w}_j$ attached to classification node j. Resonance occurs if the chose category J meets the vigilance criterion using:

$$\frac{|\mathbf{I} \wedge \mathbf{w_J}|}{|\mathbf{I}|} \geq \rho \tag{3.7}$$

The modified categorization algorithm simply removes the requirement that a single node remain active:

$$T_J = \{T_j : j = 1...N\} \tag{3.8}$$

and allows a set of active classifications to form across the Fuzzy Art network.

Identifying similar categories using the SOM network is relatively simple in comparison to the same operation in a Fuzzy ART network. The network's topology preserving structure provides a group of nodes surrounding the winning categorization which are relatively similar. The similarity of category nodes can be identified through their distance to the winning node, with greater distances between nodes representing less similarity between their input patterns. Nodes adjacent to the winning category will therefore represent the most similar input patterns.

Once a relative similarity between the winning category and other categories can be measured it is possible to apply a distance measure. Adjusting the distance measure uses a method similar to the processing model. An exact match between the categorization of a testing pattern and an exemplar reduces the distance associated with the exemplar. Categories similar to the exemplar also have their distance reduced but to a lesser extent then an exact match. A range of exemplars which are less similar then those previously matched have no change made to their distance measure. The least similar categories receive an increase to their distance.

Sequences are presented to the network until a winning X node becomes apparent and thus indicates an exemplar representing a sequence has been chosen. For the purposes of this work, the winning node is chosen by selecting the smallest distance value and the smallest array index.

In addition to allowing the network to operate with variations in the testing set the distance measure allows the network to classify sequences containing noise. The types of noise which are addressed by this system include addition, deletion and translation[1]. Isolated and infrequently occurring translations can be considered as similar to the combination of additions and deletions to the training sequences, but more frequent translations create a greater problem. If the translation is part of a trend or cycle[2] then it can be modeled to some degree by the network. Cyclic sequences are readily learned by the network. The only consideration when training upon cyclic data is that the training data set contains representative samples of the cycles such that the system will be able to accurately model them. Trends are more difficult for the system to represent if the goal is for the network to predict outside the time span of the training set. Due to the system's method of explicitly storing the training information and not building a generalized model of the data, it cannot predict values outside the range of the training set. An increasing or decreasing trend implies that the sequence will eventually reach a point greater or less than the training values and it is at this point that the network cannot be used for prediction. The system can still be used to model the data, but only within the bounds of the training set.

[1] An addition involves inserting one or more patterns into a sequence. Deletion involves removing element(s) from the sequence. Translation occurs when the pattern value itself is modified but does not involve the addition or deletion of new patterns. All types of noise are considered within the context of the training set. If the testing set contains valid sequences which are not represented in the training set then they will be treated as noise by the network.

[2] A *trend* is the underlying direction and rate of change in a time series, when allowances have been made for the effects of seasonal components, cyclic components, and irregular variation. A *seasonal component* is the component of variation in a time series based upon the time of year. The *cyclic component* is is a non-seasonal component which varies in a recognizable cycle. *Irregular variation* is variation within a time series which is not accounted for by trend, seasonal component, or cyclic component. [Por91]

Noise added to discrete data will appear as additions and/or deletions and can be correctly processed using a distance measure. Noise in continuous data is dealt with in two ways. Relatively small amounts of noise in continuous data can be absorbed by the input ART or SOM unit during clustering of the input patterns. Larger amounts of noise in continuous data will appear as insertions and deletions and can be controlled by a distance measure. All three types of noise have a similar effect on the network's behavior. A small number of additions, deletions or translations will not disrupt the time-stamp sequences greatly due to the slow decay of the distance values. Larger numbers of noisy patterns may cause the distance value to increase to the point where the network must refocus on a more appropriate subsequence.

3.7.1 Testing Algorithm

1. Select the first input pattern s_i where $i = 1$.

2. Present s_i to FA1. This will create a categorization v_i in FA2.

3. All allocated (non-zero) connections $wa_{j,k}$ from v_i to layer X transfer their value to x_k. If the weight value equals the current activation in x_k then the distance value for all of these x_k nodes is decreased by α.

4. Expand the neighborhood of winning nodes around v_i to include categories most similar to the winning category.

   - For the SOM network, use all nodes adjacent to the winning classification.
   - For the Fuzzy ART network, decrease the vigilance value ρ to the point where more categorization nodes become active.

   Decrease the distance value in all x_k nodes which have non-zero weights connecting FA2 to the newly activated categorization nodes by β.

5. Expand the neighborhood of active nodes again, using the method in the previous step. Either reduce the ART vigilance until another group of nodes becomes active,

or expand the active nodes in the SOM to include another layer of nodes adjacent to those which are currently active. The newly active nodes are processed through **Wa** to create an input across X, but they do not change the values stored in X.

6. All nodes in X which do not currently have an activation presented to their inputs have their distance values increased by γ.

7. Select the x_k node with the lowest distance value. If there are multiple x_k nodes with the same distance value then arbitrarily[3] choose the node with the lowest index k.

8. Filter the winning x_k through **Wb** to create an activation vector across FB2. The active FB2 node l is selected by matching the winning x_k activation against the weight values leading from this node to layer FB2. When $x_k = w_{k,l}$ the $FB2_l$ is given an activation value of one and all others are set to zero. This node represents the categorization of the output pattern.

9. The winning FB2 node is then filtered through the output network weights to form an output activation pattern across FB1. This pattern is the prediction of the element s_{i+1} which follows the input pattern s_i.

10. All active (non-zero) nodes in X have their activation values incremented by one. If the lateral connection between and x_a and x_{a+1} is equal to the activation in x_a then transfer the activation and distance values from x_a to x_{a+1} and set $x_a = 0$.

11. Deactivate categorization layers FA2 and FB2. Maintain the activation and distance values in X.

12. Retrieve the next input s_i where $i = i + 1$ and return to step 2. If there are no more input patterns then end.

[3]The node with the lowest index is selected simply to avoid a deadlock in the network when there is insufficient information to choose a more appropriate node. It is possible to make a more likely selection from the active nodes if a history of the most recently active nodes is maintained. In this case, the currently active node which is closest in proximity to the previously active nodes can be selected in hopes that it will be a better choice than the node with the lowest index.

3.8 Training and Recall Sample

3.8.1 Sample Training

The sample training sequence is $\mathbf{S} = (s_1, s_2, s_3, s_4, s_5, s_6) = (A, B, C, A, C, C)$. The initial step in training this network is to train the categorization networks with the sequence data. These networks serve only to reduce the dimensionality of the patterns and to group data into distinct categories. They can therefore be trained without the involvement of the temporal network.

Initialize the weight matrices **Wa** and **Wb** to zero. Set the time-stamp value $t = 1$. The first node in the temporal layer $x_{k=1}$ is activated.

An input pattern $s_1 = A$ is presented to the input layer FA1 and categorized by node j in the layer FA2. Do the same for pattern $s_2 = B$ and the output node l in layer FB2. There will now be a binary vector stored in $FA2_j$ and $FB2_l$ with one bit active in each which represents the categorization of the input and output patterns respectively. The weight values $wa_{j,k}$ and $wb_{k,l}$ between these active categorization nodes and the currently active x_1 node have a value of zero and are therefore unallocated. Set the values of both weights equal to the current time-stamp ($t = 1$) and advance the time-stamp, (see figure 3.4).

Present the next pair of patterns $s_2 = B$ and $s_3 = C$ to the input and output categorization networks respectively. This will create new categorizations in $FA2$ and $FB2$. The weights between the newly selected categories $FA2_j$, $FB2_l$ and x_1 are unallocated, so they are assigned the time stamp value t. The time stamp is incremented.

The third pattern pair $s_3 = C$ and $s_4 = A$ is presented in the same manner as the previous examples, (see figure 3.5). The weights between the selected nodes are unallocated so the $w_{j,k}$ and $wb_{k,l}$ values are set equal to t, and the time stamp is incremented.

The fourth pattern pair $s_4 = A$ and $s_5 = C$ is categorized by $FA2$ and $FB2$ as above. However, the weights between the winning $FA2$, x_1, and $FB2$ have already been allocated with the first pattern pair (time stamps 1 and 3). The active x_1 node is therefore deactivated

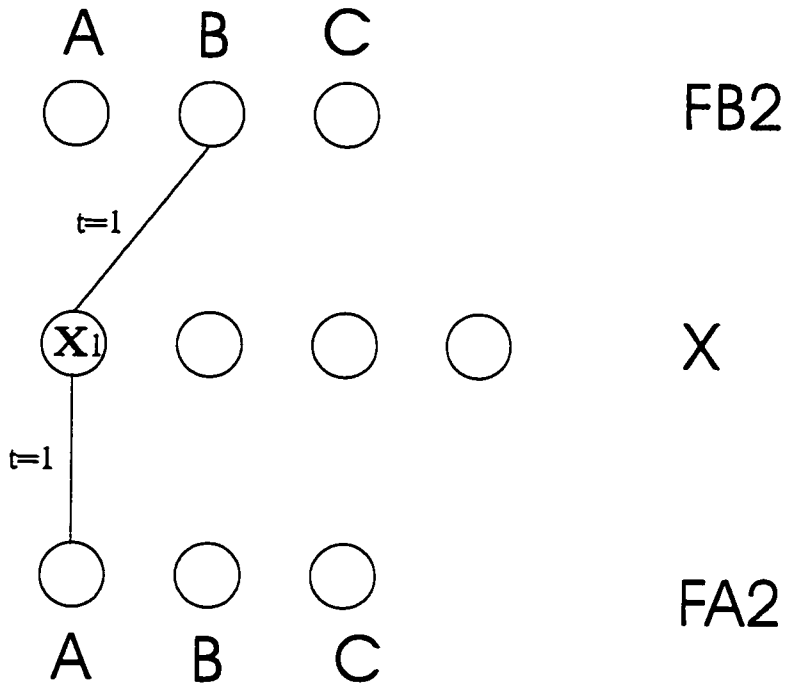


Figure 3.4: Sequential network with one input-output pattern trained.

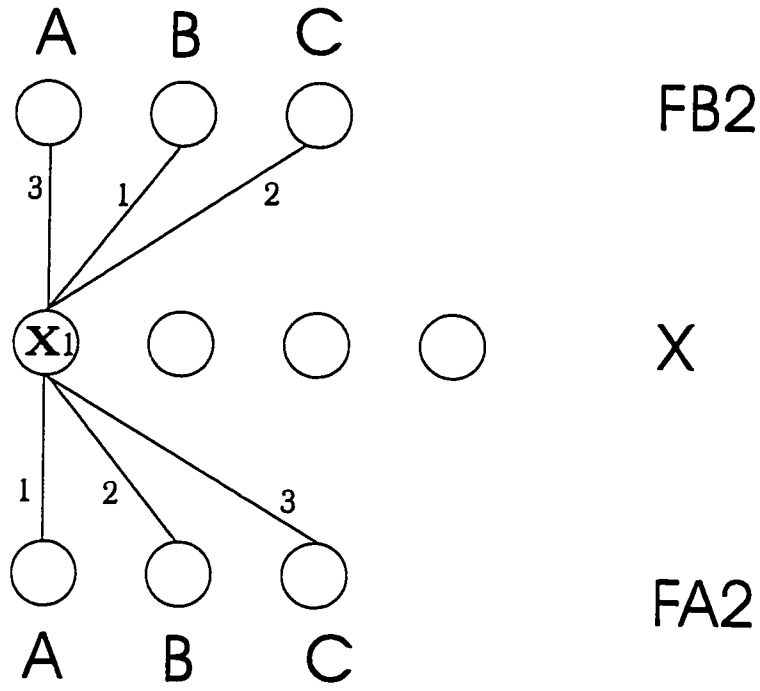


Figure 3.5: Sequential network with three input-output patterns trained.

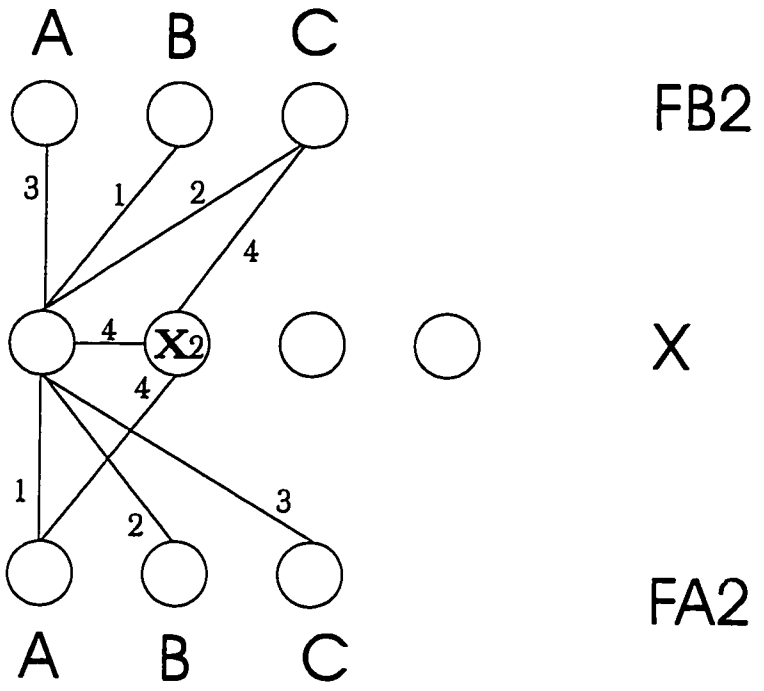


Figure 3.6: Sequential network with four input-output patterns trained.

and the next sequence node x_2 is activated. None of the weights in **Wa** or **Wb** leading to x_2 have been allocated, so the time-stamp is assigned to $wa_{j,2}$ and $wb_{2,l}$, (see figure 3.6). A lateral weight equal to the current time stamp 4 is added between x_1 and x_2. The time-stamp is advanced.

The final pattern pair ($s_5 = C$,$s_6 = C$) also requires an allocated weight $wb_{2,l}$ to store the time stamp, so the active X node is shifted to x_3 and the weights leading to this node are allocated, (see figure 3.7). The lateral weight between x_2 and x_3 is set to the current time stamp value of 5.

3.8.2 Sample Recall

Assume the training of the above network continued until the sequence
$S = (A, B, C, A, C, C, A, B, D)$ is stored in the network. The resulting network is seen in

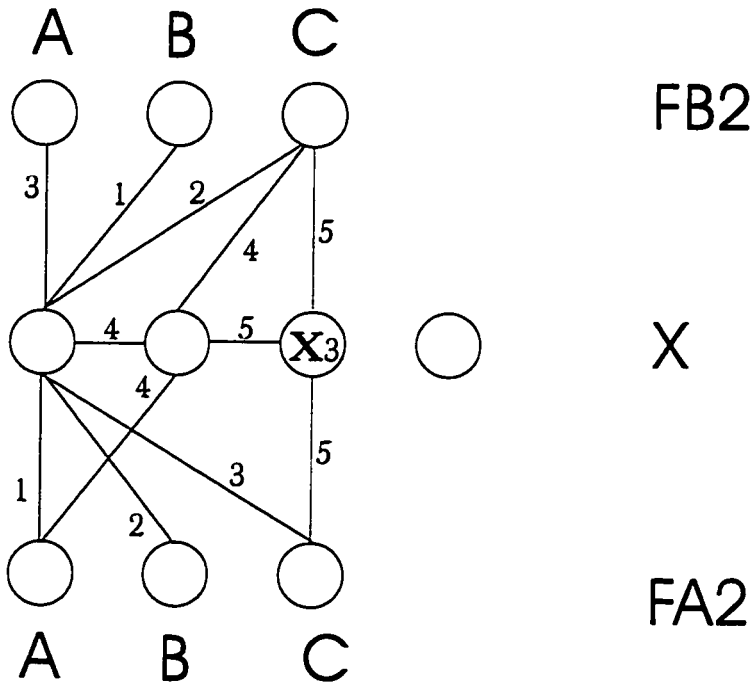


Figure 3.7: Sequential network with five input-output patterns trained.

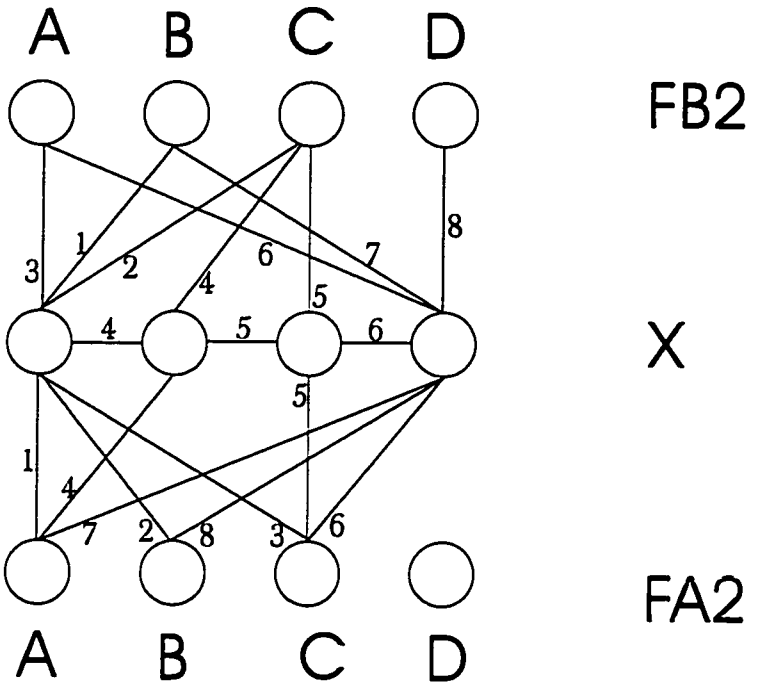


Figure 3.8: Sequential network with eight input-output patterns trained.

figure 3.8. The distance modifiers used for this test case are $\alpha = 0.1$, $\beta = -0.1$, and $\gamma = 0.1$. The distance values in X are set to an initial value of 0.5. If the training set is presented during testing the network will correctly predict the next element for the active set.

The first value $s_1 = A$ presented to FA1 will cause and categorization vector to form in FA2. All allocated weights $wa_{j,k}$ from the active category j are used to create activation values in x_k. Three weights are associated with category A, with the time stamps 1, 4, and 7. Any distance values d_k associated with x_k are decreased by $\alpha = 0.1$. Distance values which do not receive input are increased by $\beta = 0.1$. The activation vector is then $X = (x_1 = 1, x_2 = 4, x_3 = 0, x_4 = 7)$ and the distance vector is $D = (d_1 = 0.4, d_2 = 0.4, d_3 = 0.6, d_4 = 0.4)$, (see figure 3.9). The nodes with the smallest distance values are those which most represent the input sequence. The weights from X to $FB2$ which match the activations in X represent the output pattern which is likely to follow the current input. By following these weights from the nodes in X with the smallest distance value the next

element in the sequence can be identified. In this example the patterns which are most likely to follow the input A are B and C.

Activation values in X are then incremented to create $X = (2, 5, 0, 8)$ in preparation for the next input. The activation value in x_2 is now equal to the lateral weight between x_2 and x_3 which means the activation and distance in x_2 are transfered to x_3. This occurs because the distance value of x_2 is less than that of x_3. The values stored in x_2 are reset to their initial values after the transfer. When the next value $s_2 = B$ is presented to the classification network it forms a new activation in $FA2_j$. Two weights $wa_{j,k}$ are associated with the classification node for B, they are time-stamped 2 and 8. The activation for B is processed through the weight layer and appears as an activation on nodes x_1 and x_4, (see figure 3.10). The value from the weight matches the activation found in x_1 and x_4 which causes the distance value of this node to be reduced by α and produces $D = (0.3, 0.5, 0.5, 0.3)$. The activation transfered to x_3 is not matched to an input so the distance for that node increases by β (from 0.4 to 0.5). The lowest distance values appear in nodes x_1 and x_4. The weights which match the activations currently in these nodes lead to the output nodes for patterns C and D. This suggests that the pattern following the input sequence A, B should be either C or D which matches the training sequence.

The time-stamps in X are incremented creating an activation vector of $X = (3, 0, 5, 6)$. The activation in x_3 matches the lateral connection to x_4 but the distance value in x_3 is greater than that of x_4. Because of this the activation and distance are not passed from x_3 to x_4. The node x_3 is reset to an activation of 0 and a distance of 0.5. The activation in x_4 is now 9 which is larger than the network has stored so it can be reset as was x_3. The next input pattern C activates three weight values connected to X. The weights 3, 5, and 6 lead to nodes in X, (see figure 3.11). The weight 3 matches the activation in x_1 so the distance is reduced to 0.2. The nodes x_3 and x_4 were both recently reset to their initial values and they now both receive inputs from weights 5 and 6 respectively. As they are uncommitted and have no activation value they take on the values of their inputs and their weights are reduced by α. The x_2 node receives no input so its distance value in increased by β. The resulting

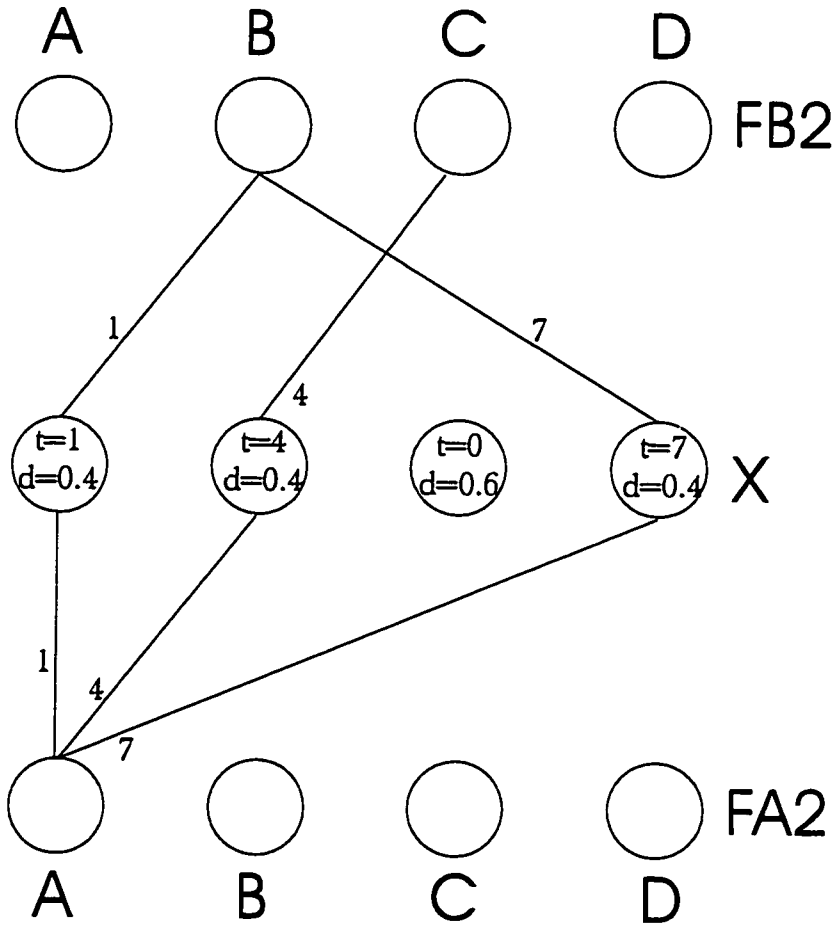


Figure 3.9: Sequential network recalling first trained pattern.

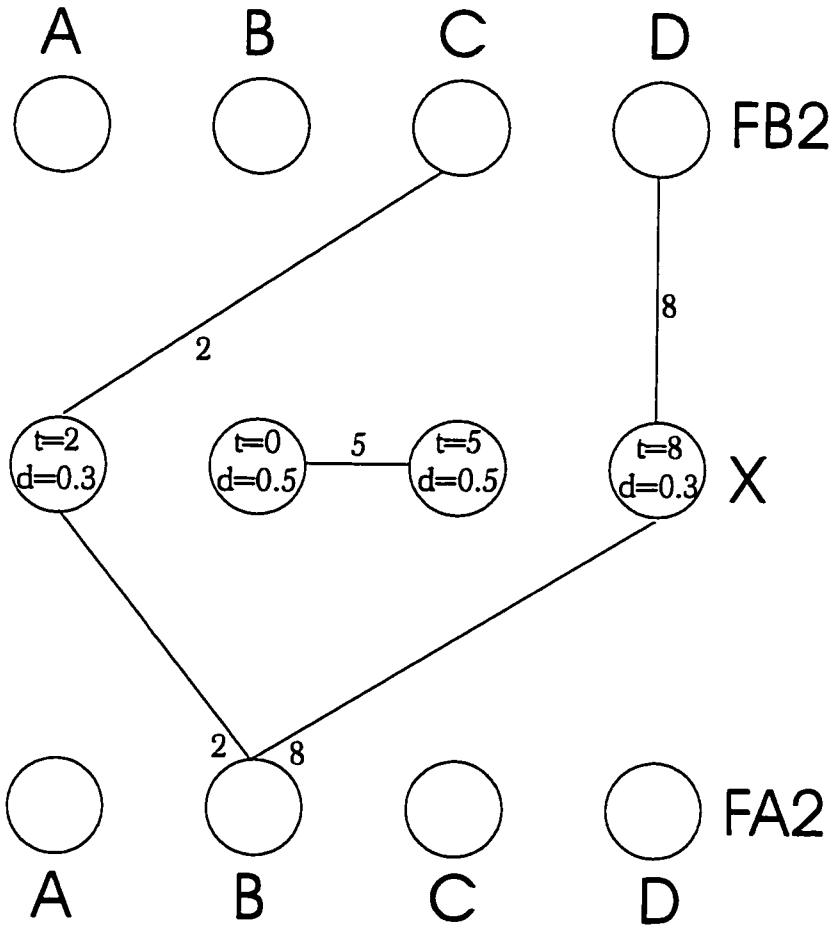


Figure 3.10: Sequential network recalling second trained pattern.

activation vector is $X = (3, 0, 5, 6)$ and the distance vector is $D = (0.2, 0.6, 0.4, 0.4)$. The sequence node with the smallest distance is x_1 with an activation of 3. The associated output category matching the activation of 3 is A which is the correct pattern to follow an input of A, B, C. This is the first time when there has been a single smallest distance value in X which identifies one potential output pattern. At this point the network has correctly identified the location in the training sequence based upon the first three input patterns.

The time stamps in X are advanced again leading. The activation values in nodes x_1 and x_3 are equal to the lateral weights leading to x_2 and x_4 respectively. The distance in x_1 is less than that of x_2 so the activation and distance are transfered to x_2. The activation and distance value for the closest matching pattern have moves from sequence node x_1 to x_2. The distance associated with x_3 is not greater than that of x_4 so the values are not transfered to the next node and x_3 is reset. The next input pattern A is presented to the network which causes the weights 1, 4, and 7 become active, (see figure 3.12). The weights are matched to the activations in X which reduces the distance values in x_1, x_2, and x_4 to be reduced by $\alpha = 0.1$. The node x_3 does not receive input and is increased by β. The sequence node with the smallest distance is x_3 and when its activation is matched to the weights leading to $FB2$ the output pattern is C. This is the correct output prediction given an input sequence of A, B, C, A.

3.9 Network Characteristics

Desirable traits which this architecture supports are:

1. Sequences should be identified based upon the current input. If more than one sequence identification is currently possible based upon input patterns, then the network should reflect this.

2. The next element(s) in the sequence should be predictable based upon the current state of the system.

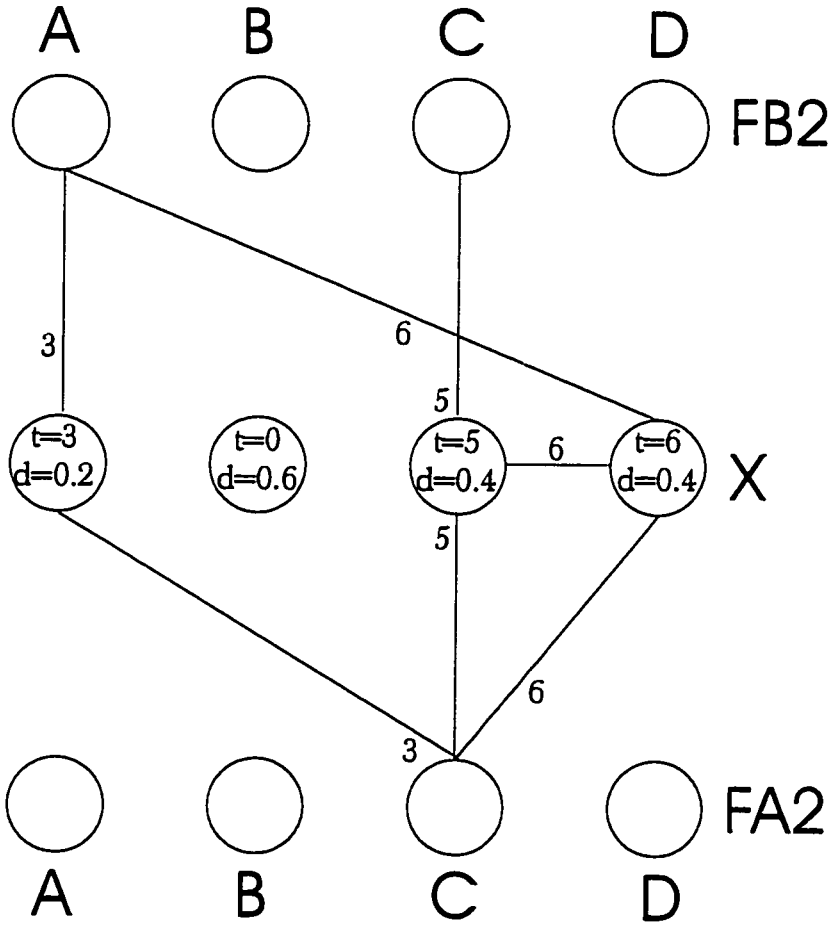


Figure 3.11: Sequential network recalling third trained pattern.

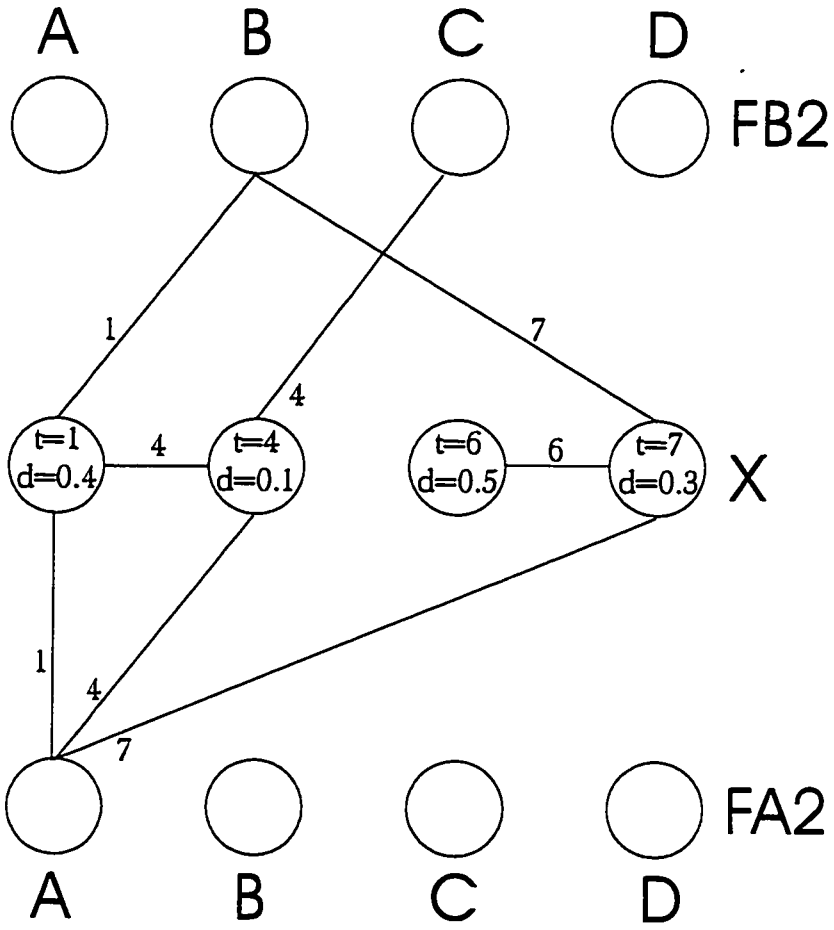


Figure 3.12: Sequential network recalling fourth trained pattern.

3. The current sequence classification should affect the next classification.

4. Sequences with noise superimposed over their elements will suppress the noise and continue with the classification.

5. This architecture should not buffer input patterns. This restriction is somewhat artificial and can be relaxed once the sequence classification dynamics are well defined. The objective of first developing a non-buffering network was to force a design which could achieve as much as is reasonably possible without relying on buffers to maintain the entire temporal context.

6. Repeated identical elements should be identifiable as different. This prevents patterns A and AAAAAAA from being classified as identical.

Desirable traits which the proposed architecture does not support include:

1. Some form of compression of the training sequence is desirable. Sub-sequences should be associated with the same node in the categorization layer. These sub-sequences could be either of similar patterns grouped in a common classification node, or may be arbitrary parts of the training set.

2. Sequences which are both post-pended and pre-pended with other elements should be classified as the same.

3. Similar strings which are slightly different (i.e. $ABCD$ and $ABQD$) should be classified by the same categorization nodes.

Some of these desirable characteristics are contradictory in nature. In particular, the characteristics of pattern compression and maintaining exact representations of the input sequences are not likely to both be entirely satisfiable.

3.9.1 Guiding Themes in Development

Due to the virtually unbounded nature of developing a new system, several themes were used to guide this work. These themes were not intended to be strict rules which were slavishly followed to exclusion of all other methods, but were instead chosen because they provide some form of structure to the development and each provides a group of benefits which are hopefully complimentary.

The first overriding theme in this work was to develop a neural network based system for sequence processing. This provided a basic architecture for the system with a collection of associated benefits and limitations. Benefits include such things as an architecture which lends itself to parallel processing, fault tolerance, graceful degradations, simple representations and simple processing. Limitations which arose from this include the restriction of processing to relatively simple operations which are appropriate for a node or weight in a network, and a largely static architecture which does not readily reorganize its components or allow large sections to be moved or reorganized in some way. Another ANN characteristic which is adhered to is the relative consistency with which training is performed. Specifically, training involves the repeated application of an algorithm which adjusts the weights within the network. Parameters to this algorithm are likely to change, but the basic operations within the algorithm are the same throughout training. Generally, networks have only a single weight between any two nodes.

"Sequential leader clustering"[4] is used in conjunction with the neural network theme. Its method of developing exemplars for clusters based upon the order in which sequences are presented to it provides an appropriate model for neural sequence processing.

Multiple elements are not presented to the system simultaneously. The information

[4]The first input is chosen as the exemplar for the first cluster. The next element is compared to the first cluster exemplar. If the distance between the two is smaller than a given threshold then the new input is clustered with the first. If it is greater than a threshold then a new cluster is created. This is continued with more inputs. The number of clusters grows with time and is dependent upon the distance measure and threshold value. [Lip87] [Har75]

gained from multiple simultaneous inputs can be achieved through sequential presentation of the same data, without the overhead of a larger number of nodes and weights to process the input data. Multiple inputs also lead to a similar architecture as the windowed systems described in Chapter 2 which have a variety of undesirable characteristics.

3.9.2 System Features

Features of the system include:

1. Training occurs in a single pass and is therefore quite fast.

2. This time-stamp method has reasonable growth characteristics when compared to existing ANN systems which perform similar functions.

3. Each pattern is explicitly coded so there no ambiguity with previously seen data and no data is destroyed through further training.

4. The system attempts to take advantage of the large number of weights associated with a neural network by storing a great deal of the training data verbatim. It then uses this information during recall to develop predictions for testing data.

5. Due to the initial clustering of inputs, both discrete and continuous data may be modeled using this system.

6. The system does not require any modification to model sequences which are combinations of two or more disparate subsequences. The network does not create a generalized representation of the training sequence, it stores an explicit representation of the data. Due to this method of storing the training set the network is not required to either identify the different types of subsequences and model them using local systems or to form a global model for dissimilar subsequences.

7. The time-stamp network makes no attempt to group similar elements into the same weights. This is not optimal in terms of compression in the network, where a highly

compressed network would store all repeated patterns in the same set of weights. However, it may not be as wasteful as it first appears. Due to the many uses to which the time-stamp values are applied, it may be an acceptable waste in order to gain the benefits from storing all sequences explicitly.

8. Each hidden node in the internal network is used to store varying sized subsequences. This does provide some grouping, but in terms of nodes, not weights. This is not as desirable because grouping by weights reduces the overall number of weights in the system.

9. The system uses parallel evaluation of sequence elements. Due to the nature of the training algorithm, each internal node can only receive one time-stamp for a given input. This guarantees that if an input element appears multiple times in the sequences, it will be evaluated by multiple internal nodes.

10. This architecture is built using components which lend themselves to modification. It is clear what function each component performs and how they achieve their state. This is unlike gradient descent methods which have a more opaque state.

11. The architecture allows for any of three layers to be activated in order to generate an output. If either of the Fuzzy-ART networks are activated, then either the next or previous sequence element will become active in the opposite network. If an internal network node is activated, then the sequence pair associated with that time-stamp will become active. If both Fuzzy-ART networks are activated, then all time-stamps associated with those pairs will become visible in the internal network.

12. A one-to-many mapping of inputs to outputs does not provide enough information for the network to resolve a single active exemplar node. All potentially correct nodes remain active until enough inputs have been presented to choose the appropriate exemplar. This behavior is not often found in neural networks where the norm for many systems is to select a single association based upon the current inputs. The ability to maintain multiple activations which represent several potentially correct

hypothesis and not to choose one prematurely is an important characteristic of this technique.

13. Fuzzy-ART networks allow for categorization of input data into a smaller number of groups if desired with more patterns in each category. This can be used to reduce the network size.

3.9.3 System Limitations

This system doesn't build a generalized view of the data. It instead memorizes pairs of sequence elements. Other techniques such as Time-Series modeling and State-space embedding provide a model which provides a broader representation of the entire data set.

The system does not recognize trends in the data as such. It will learn the input-output patterns and will model them as they appear in the training set, but it cannot predict a trend outside the range of the training values. This problem generalizes to an inability in the network to model any behavior which does not appear in the training set. This is due to the explicit method for storing and recall of the training set values. Because the system does not create a generalization of the dynamics in the training set it cannot predict values outside of this range. This is not a severe limitation as there are methods available to address the problem. It is possible to identify trends through exploratory data analysis and remove them through preprocessing of the data. Many existing neural network models also rely upon a memory model for storage and retrieval of patterns and have a similar limitation in ability to retrieve patterns outside the range of the training set.

The parameter values α, β, and γ cannot be easily identified before training begins. A method to predict or at least estimate these values before testing begins would greatly improve in the network's usefulness, particularly with continuous data.

Since the system is a neural network, there is an inherently large number of values stored in the system. This is a common feature found in massively parallel systems.

Chapter 4

Testing Methodology and Results

4.1 Test Data

Four data sets are examined during testing. They include a set of traditional tests for sequential networks, a set of generated data containing well defined characteristics, two sets of data from the Santa Fe time series competition, and the NETtalk data set. In each test case the sequential network is trained upon a data set and then tested for its ability to predict the next sequence elements given a partial testing sequence.

4.1.1 Traditional Test Sets

Several test sets which are applied to sequential networks are not very interesting when used with this system. Many of these data sets have been devised in order to examine a network's ability to develop a generalized representation of the data. Because this system works as a memory and does not build a generalization of the data it is not very difficult for the system to model these sequences. It does require that the system is presented with an exhaustive training set, as it is not capable of generalizing from partial data. Two of these data sets have been tested with the sequence network, they are the sequential XOR

problem [Elm90] and the reproduction of sequences [Moz89]. In both cases the network was able to learn and reproduce the desired sequences with one hundred percent accuracy.

4.1.2 Generated Data

The initial data sets used in testing were generated to have a variety of characteristics. These test sets have a specific property which is examined once the network has been trained upon the data set. Each set is used to test a property of the system without interference from other features of the data set.

1. The ramp data consists of a linear, increasing trend with no repeated values or cycles. (see Figure 4.1). The feature of interest in this data set is the trend.
2. The randomly reordered ramp data, (see Figure 4.2), is the previous data set, but randomly ordered. It is non-linear, contains no trend, cycles, or repeated elements. This data set has no regular structure.
3. The Sine data set consists of a cyclical sine function superimposed over an increasing trend. It also contains repeated elements.

4.1.3 Santa Fe Test Set

The Santa Fe institute maintains a collection of temporal sequence data which has been studied using a variety of methods [WG94]. The data sets are:

- Set A – A physics laboratory experiment containing the fluctuations in an infrared laser, approximately described by three coupled nonlinear ordinary differential equations.
- Set B – Physiological data monitoring the heart rate, chest volume, blood oxygen concentration, and EEG of a sleeping patient suffering from apnea.

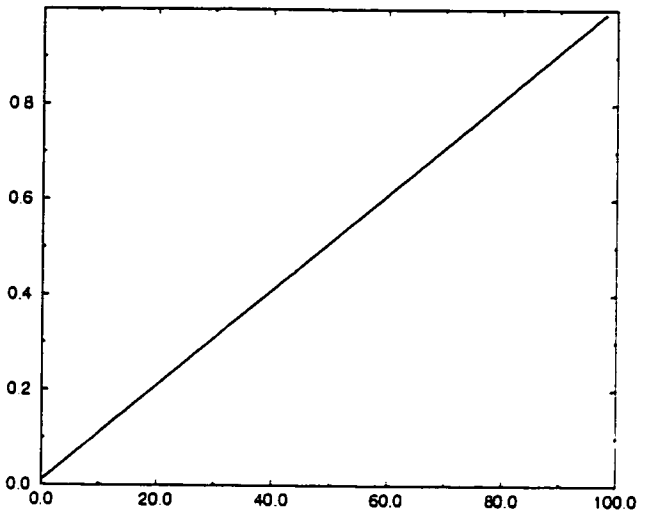


Figure 4.1: Ramp data, 99 points from 0 to 0.99 in 0.1 increments.

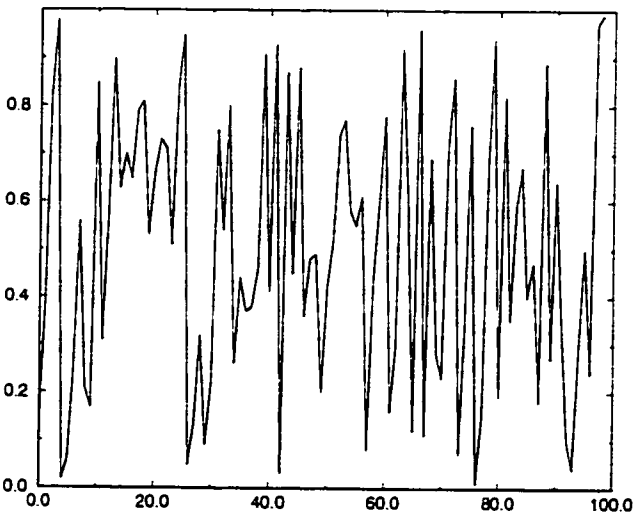


Figure 4.2: Randomly ordered ramp data, 99 points from 0 to 0.99 in 0.1 increments.

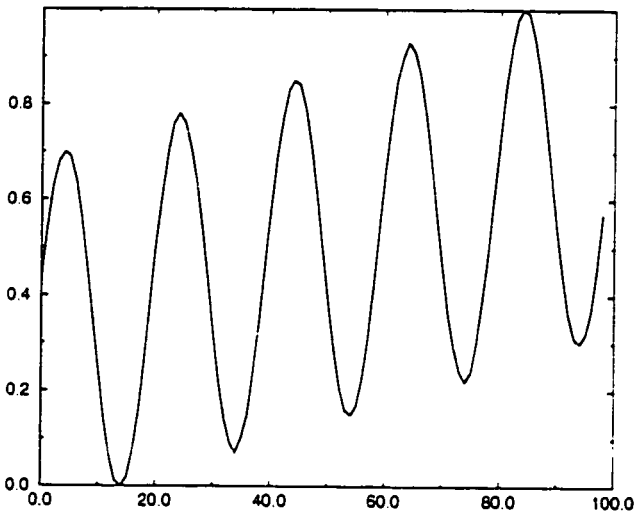


Figure 4.3: Sine data with increasing trend, 99 points.

- Set C – High-frequency currency exchange rate data representing the exchange rate between the Swiss franc and the U.S. dollar.
- Set D – A numerically generated series created by integrating the equation for motion of a damped, driven particle:

$$\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + \nabla V(x) = F(t)$$

- Set E – Astrophysics data from a variable white dwarf star.
- Set F – An incomplete Bach Fugue.

These data sets provide a variety of challenges for temporal analysis methods. Included in these sets are highly predictable and unpredictable data, noisy and clean samples, artificial and naturally occurring data, and at least one chaotic set. Many different sequential analysis methods have been applied to these data sets which will provide a wealth of results to compare with the performance of Temporal ART.

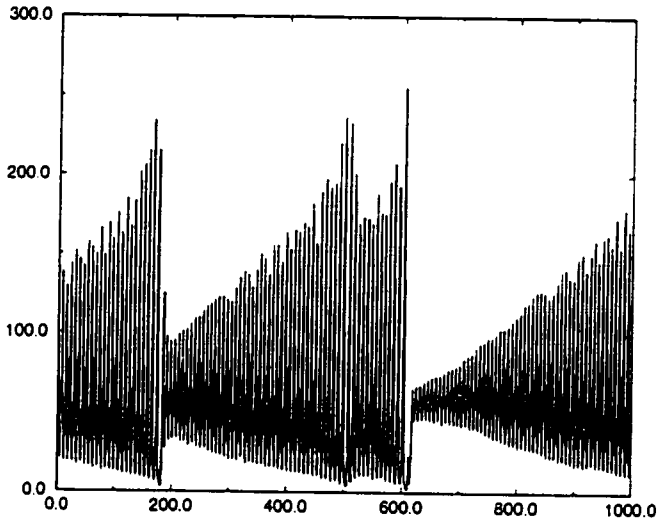


Figure 4.4: Training data set A - 1000 points.

Data sets A and D have had the largest number of different analysis techniques applied to them and for that reason they will be a focus of this work. Data set A (see Figure 4.4) consists of 1000 values in the training set and 100 in the testing set, figure 4.5, which do not overlap with those in the training set. It consists of a repeated cycle which increases in amplitude until it reaches a maximum at which point a catastrophic collapse occurs. There is no trend in this set and the training set is relatively short.

Data set D contains 20000 values in the training set and 500 values in the testing set, (see Figure 4.6). This data set oscillates but with no obvious frequency. It does not contain a trend and consists of a large amount of data.

4.1.4 Text to Speech Data Set

The problem of converting ASCII encoded text to its equivalent phonemic representation was chosen because of its familiarity from the NETtalk experiment. NETtalk is a often cited

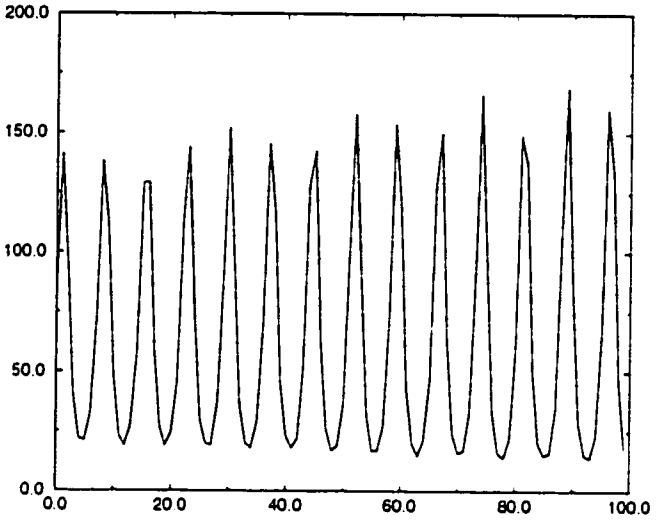


Figure 4.5: Testing data set A - 100 points.

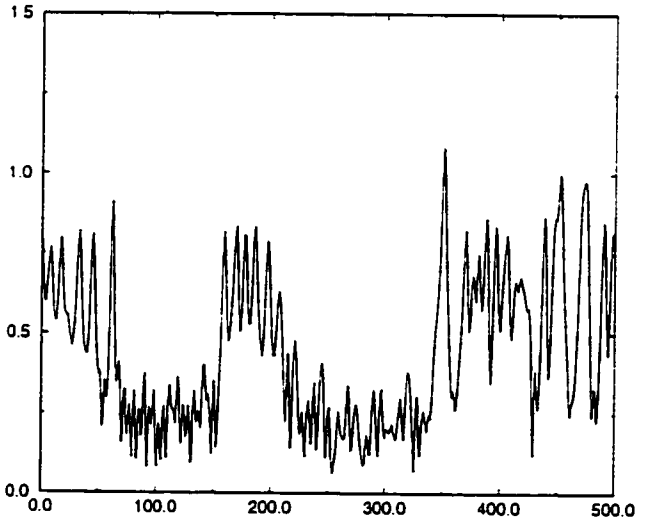


Figure 4.6: Testing data set D - 500 points.

Text Data	Phonemic Data
ardvark	a-rdvark
aback	xb@k-
abacus	@bxkxs
abaft	xb@ft
abalone	@bxloni
abandon	xb@ndxn
abase	xbes-
abash	xb@S-
abate	xbet-

Table 4.1: Sample of NETtalk data set.

neural network experiment which uses input windows as a means to maintain a temporal context [SR86]. Due to its familiarity, readily available data set, and its temporal nature it is a useful problem to examine in this work.

The NETtalk data set contains approximately 20000 English words and their phonemic equivalents (see table 4.1). The words range from one to twenty-six characters in length. Mappings from character combinations to phonemes are not unique, so it is possible to have the same subset of letters in two different words which do not produce the same phonemes. In this case, the context information gained from other parts of the ASCII word must be used to produce the correct output for the identical subpatterns.

A feature of the NETtalk set is its use of a different set of data for the input and output of the network. The previous data sets all consist of input and output from the same domain and the network is used to predict the next element given the current input. The NETtalk data set requires the association between input and output to represent the mapping from the alphabetic set to the phonemic set.

4.2 Results

4.2.1 Generated Data Results

The results from testing with the generated data sets are listed in appendix A. The data sets are used as a simple test of the network's ability to predict the next sequence value when presented with the sequence elements which proceed it. The networks were tested with a range of ρ, learning rates, and different epoch values for the ART network to determine which where the most appropriate for use in within the sequence network. The network was trained upon the data set and then presented with the same elements during testing. After each input pattern is presented to the system the output is calculated and compared to next sequence value. The output at this point is the prediction of the next pattern in the data set.

The ramp and randomized ramp data both successfully model the data sets when large learning rate and ρ values are used. Increasing the number of epochs has little effect on the network's ability to recall values when the learning rate and *rho* parameters are large, but it does improve recall when these values are smaller. The sine data set does benefit from a larger number of epochs and a smaller learning rate value. This is due to the overlapping nature of the data set which requires more training on the data to build a distinction between its similar values. The ramp and random data do not have any overlapping sequence values. Each value in the data set is therefore unique and easier for the network to learn.

In all cases the data sets were modeled with 98 to 100 percent accuracy. The system successfully modeled the data with the 98 percent result coming from the sine data set. This was do to the overlap in the values appearing in that sequence. The other sequences were modeled with 100 percent accuracy.

4.2.2 Santa Fe Test Set Results

Tables 4.2 and 4.5 summarize the results from [WG94] in which several different techniques were compared for data set A and D. The tables list all of the neural network methods tested (conn) and the most successful non-neural methods, a local linear model (loc lin) using delay coordinate embedding, a linear method (linear), and a kd-tree (tree) method.

The normalized mean square error (NMSE) which is used to compare the results for these data is calculated using:

$$\text{NMSE(N)} = \frac{\sum_{k \in \tau}(\text{observation}_k - \text{prediction}_k)^2}{\sum_{k \in \tau}(\text{observation}_k - \text{mean}_\tau)^2}$$

A NMSE of 1.0 indicates that the system predicted the mean in all cases.

The sequential network with Fuzzy ART classification consists of FA1 and FB1 layers of width 2, FA2 and FB2 layers each of which contain 260 nodes, and a X layer of 300 nodes. The Fuzzy ART parameters are fixed to $\rho = 1.0$ to generate individual classifications in F2 for each input pattern. The β value is set to 1.0 for fast training and as a result only one epoch of training is required. The network using the SOMs as classifiers consists of an input layer of two nodes, a one-dimensional classification layer of 255 nodes, and a X layer of 1000 nodes.

Two different tests were run with each data set. In the first test the correct sequence element was always used as the input to the system. In the second test the predicted output was fed back to the input of the network. The former test illustrates the network's behavior when predicting a single point in the future, the latter test shows how well the network is able to predict multiple time steps into the future.

As a test of the network's ability with single point prediction. It was trained upon 1000 points and tested by presenting each element of the testing sequence. The results generated in this work are comparable to the results in table 4.2. The processing model is comparable to the best connectionist results and slightly better than the most successful non-connectionist method. The connectionist models used in this work do not perform quite

as well as the processing model. The Fuzzy ART based network does not perform as well as the best connectionist and non-connectionist Santa Fe systems, but it does return results which make it the second most accurate connectionist system for prediction with data set A. The sequential network using SOMs for classification is not particularly successful when used to classify data set A, producing the fourth best results overall.

When multiple point prediction is used, the network is again trained upon 1000 points and then allowed to predict the next 100 sequence elements. These 100 point predictions where compared against the continuation of the sequence to determine the system's ability to predict over larger numbers of elements. As would be expected, the performance of the network deteriorate with predictions further into the future. The behavior of the ART based network is still comparable to the better results in table 4.2 but the SOM based network performs considerably poorer with multipoint prediction.

Although the sequential network does not consistently produce the best results on data set A, the processing model and the ART sequential networks did perform comparably to the best Santa Fe methods and were 86000 and 7200 times faster than the most successful connectionist model.

Tables 4.5, 4.6 and 4.7 show the results for data set D from both the Santa Fe competition and from this work. The results are the NMSE for each system when tested with sequences of 15, 30, 50, and 500 input patterns.

As with data set A, the Fuzzy ART networks which data set D is trained upon have $\rho = 1.0$ and $\beta = 1.0$ and are trained for 1 epoch. The FA1 and FB1 layers consist of two nodes each. FA2 and FB2 are 1100 nodes, and the X layer is 900 nodes. The SOM network tested with data set D has 2 nodes in each of FA1 and FB1, a one-dimensional classification layer of 500 nodes, and an X layer of 3000 nodes.

The single point prediction results developed herein are consistently more successful on data set D than any of the Santa Fe methods. In most cases the NMSE decreases as more data is presented to the network. This is to be expected as it provides more of a context which the network may use in making a prediction. Because the sequential network uses

Method	Type	Computer	Time	NMSE
conn	FIR Network	SPARC 2	12 hrs	0.028
loc lin	Delay Coordinate Embedding	DEC 3100	20 min	0.080
conn	Feedforward Network 50-350-50-50	386 PC	5 days	0.38
tree	K-D tree	VAX 6420	20 min	0.62
loc lin	21 dim, 30 nn	SPARC 2	1 min	0.71
conn	Feedforward Network 200-100-1	CRAY Y-MP	3 hrs	0.77
conn	Feedforward Network 50-20-1	SPARC 1	3 wks	1.0
loc lin	3 dim time delay	Sun	10 min	1.3
conn	Recurrent Network	Vax 8530	1 hr	1.4
conn	Feedforward Network	SPARC 2	20 hrs	1.5
conn	Feedforward, weight-decay Network	SPARC 1	30 min	1.5
linear	Wiener Filter, width 100	MIPS 3230	30 min	1.9

Table 4.2: Results for Santa Fe data set A from [WG94] for 100 points prediction.

Method	Computer	Time	NMSE
Processing Model	Pentium 120	0.5 sec	0.037
ART	Pentium 120	1 min	0.267
SOM	Pentium 120	7 min	0.538

Table 4.3: Results from applying temporal network developed in this work to Santa Fe data set A for single point prediction.

Method	Computer	Time	NMSE
ART	Pentium 120	1 min	0.372
SOM	Pentium 120	7 min	0.805

Table 4.4: Results from applying temporal network developed in this work to Santa Fe data set A for 100 points prediction.

Method	Type	Computer	Time	NMSE(15)	NMSE(30)	NMSE(50)
conn	Feedforwd Net	CM-2	8 days	0.086	0.57	0.87
tree	k-d tree	VAX 6420	30 min	1.3	1.4	1.4
conn	Recur Net	Vax 8530	n/a	6.4	3.2	2.2
conn	FIR Net	SPARC 2	1 day	7.1	3.4	2.4
linear	36 AR(8)	SPARC 2	10 min	4.8	5.0	3.2
conn	Feedforwd Net	SPARC 2	20 hrs	17.0	9.5	5.5

Table 4.5: Results for Santa Fe data set D from [WG94].

Method	Computer	Time	NMSE(15)	NMSE(30)	NMSE(50)	NMSE(500)
Proc Model	P120	1 min	1.027	0.470	0.277	0.097
ART	P120	32 min	0.184	0.134	0.122	0.129
SOM	P120	19 min	0.025	0.023	0.029	0.090

Table 4.6: Results from applying temporal network developed in this work to Santa Fe data set D for single point prediction.

a distance based measure which requires several input patterns before it can focus on an appropriate solution it does not perform well with the shorter test sequences. The time required to generate these results was again considerable less than those listed in the Santa Fe results.

The multiple point predictions are also quite successful when compared to table 4.5. Again, the single point predictions tend to outperform the multiple point predictions.

Although the SOM based network outperformed the ART network for prediction in this case, it did not work as well with data set A. This suggests that the SOM network may be more susceptible to influences from the data than the processing model or the Fuzzy ART based network.

The parameters which produces the best results for data sets A and D for single point

Method	Computer	Time	NMSE(15)	NMSE(30)	NMSE(50)	NMSE(500)
ART	Pentium 120	32 min	0.394	0.358	0.301	0.471
SOM	Pentium 120	19 min	0.067	0.088	0.082	0.598

Table 4.7: Results from applying temporal network developed in this work to Santa Fe data set D for multiple point prediction.

prediction appear in tables 4.8 and 4.9. With the processing model for both data sets cases, small α and β values produce the best results. The wide range of γ values which appear with data set A suggest that it has little effect on the distance value. The range of γ values for data set D suggest it is either similar to data set A and has little influence on the pattern matching, or that it is has an effect, but only with larger values. The Fuzzy ART networks also appear to benefit from a small α value but differ greatly in the β and γ values which produce the best results with the two data sets. The SOM based networks require quite different in the parameter setting than the ART based networks to generate good results. The SOM networks work best with high α and low β values. The more successful γ values for the SOM based networks are not consistent over the two data sets.

It appears in both data sets that the ART based networks generally generate better results with smaller α and larger β and γ values. The inverse is true of the SOM networks which appear to work best with large α values and smaller β and γ values. This suggests that the sequence networks with ART classifiers rely more upon near misses (β values) and far misses (γ values) to moderate the distance measure and that the relatively small α does not have much effect on the sequence network. The opposite is true for the SOM network which appears to rely heavily on large effects (large α) from exact matching.

When generating multiple point predictions, the parameters for the SOM based networks which produce the better results are the same as those in tables 4.8 and 4.9. The parameters for the Fuzzy-ART networks which produce the best results are different from those in the single-point predictions. Better results are achieved when all three parameters are set to 0.1.

Method	α	β	γ
Processing Model	0.01	0.01	0.01-0.99
Fuzzy ART	0.01	0.1	0.5-0.99
SOM	0.90	0.01	0.01

Table 4.8: Parameters which produce the best sequence recall for data set A.

Method	α	β	γ
Processing Model	0.01	0.01-0.1	0.5-0.99
Fuzzy ART	0.01-0.1	0.5-0.99	0.3
SOM	0.7-0.99	0.01-0.1	0.1-0.7

Table 4.9: Parameters which produce the best sequence recall for data set D.

4.2.3 NETtalk Data Set Results

The NETtalk testing used two different data sets. The first contains 20008 words, which consists of 166934 input-output pattern pairs, and the second contains 1000 words, with 6438 input-output pairs. Both tests involved training on the larger data set. Testing was performed on each of the two data sets. The parameters for the network used in these tests are $\alpha = 0.1$, $\beta = 0.01$, $\gamma = 0.01$.

In all of the previous test sets the input-output patterns did not consist of different types of data. With this data set the input pattern is built using 26 letters and the output consists of 52 symbols which represent phonemes. This creates a situation which is not possible in the previous tests where two identical input sequences can have different associated output patterns. This occurs when the input words are synonyms. There are a total of 139 synonyms in the 20008 word data set. With the synonym input patterns there is not sufficient information to decide which output sequence is appropriate. In this case a context larger than that of individual words must be maintained in order to select the correct output. This was not done in this work and as such the synonym patterns are often

categorized incorrectly.

This testing was necessarily different from the original NETtalk experiments. The sequence network requires more information to determine which is the appropriate output than a single input pattern. As such, predictions for each output pattern were made only after the end of a word was reached on the input to the network. The network was therefore configured to output a sequence of output patterns at the end of each word and this sequence represented the appropriate output patterns given the previously input word. This was necessary because the network developed herein does not receive multiple inputs simultaneously, as NETtalk does, and it therefore cannot generate a context for each input pattern based upon a single input presentation. The network must receive several inputs in order to generate the context which is present with a single input pattern in the NETtalk system.

When training and testing of the network is performed using the larger test set then the number of input-output patterns which are not categorized correctly is 1191. Given the total number of 166934 input-output patterns the error with this data set is 0.71 percent. The total number of characters which appear in synonymous input-output patterns is 3821 so the percentage of correctly identified synonyms is 0.69.

Testing with the 1000 word data set is performed on the network which was trained with the 20008 word data set. The smaller data set contains no words which are synonyms so the recall of the patterns works with one hundred percent accuracy.

The percentage of correct classifications when using the original NETtalk system are 98 and 90 percent for the small and large corpora respectively. The NETtalk results [SR86] are less successful than those generated using the network described in this work.

4.3 Justification for New Method

Benefits of this neural network solution to temporal classification are:

- The training algorithm is quite fast and allows for single pass training of temporal associations.

- The system produces stable outputs in both the long term and short term. It is not prone to chaotic outputs when presented with novel inputs and will always generate outputs derived from the training data.

- Using exemplars to store the training data has several benefits. The matching function for exemplars to testing data is quite easy and allows for rapid comparison. The output of the network is guaranteed to be similar to the training set and due to the relatively slow rate of change in the distance measure it is less likely to produce random or chaotic outputs than many existing network architectures.

- The proposed method can be used with any technique which provides a classification for a given input pattern. However, it does require a method which is capable of identifying the relative distance between categories.

- The technique does not rely on probabilistic or gradient descent methods. It is not susceptible to statistical error and cannot be trapped in a local solution. Due to the deterministic nature of the distance measure the system cannot be trapped in a local solution and is not susceptible to statistical error. A locally optimal solution will appear as a short term decrease in one of the distance values associated with the training sequence, but will not unduly force the system to select it as the best matching unless it persists sufficiently long enough to no longer be a local phenomena.

- The system capable of learning very long sequences because of its explicit coding of training data.

- The traditional benefits to a neural network are also available in this model, such as distributed representation, fault tolerance and graceful degradation, and potentially fast operation when built as a parallel hardware system.

Chapter 5

Analysis and Conclusions

The use of a distance measure to represent pattern matching provides several benefits not found in many popular neural network models. These include ease of analysis and modification to the system's training and recall algorithms. This is largely due to the well understood nature of distance measures and the elegant manner in which they translate a complex comparison to a single value. It is therefore important to establish that the method of generating distances in this network will produce a true distance measure. There are several other characteristics of the network which can be established by examining the structure and algorithm of the system.

5.1 The Distance Measure

The distance measure consists of two parts, identifying the classifications which have the smallest distance to the input and then modifying the distance values stored in the network. The proximity of categories in the ART networks is first determined by reducing the vigilance parameter ρ and noting the point when other categories become active. The first categories which become active are those nearest in distance to the original categorization and input pattern. When one or more neighbors to a categorization are determined the

distance values for the stored in the network are updated. Once it has been established that the method of modifying the vigilance parameter actually generates a distance measure then several characteristics of the network can be assumed.

Three properties should be present in a distance measure [Mes83]. Given points A, B, C the distance should be defined such that the distances AB, BA, BC, and AC have the following properties:

1. $AB = 0$ if and only if $A = B$
2. $AB = BA$
3. $AB + BC \geq AC$

The distance measured between two input patterns presented to the network must be considered in relation to their classification by the ART network. This is due to the sequence network's method of creating a distance value between the categorizations created by the ART network. When a classification is made the winning node v_i in the F2 layer becomes active and initial vigilance value ρ is set to 1.0. The distance between categorizations is generated by reducing ρ to the point when another classification node becomes active. The distance value is derived from the amount which ρ must be reduced.

To meet the first requirement for a distance measure it must be true that $AB = 0$ if and only if $A = B$. Given two categorizations in F2 for patterns A and B the distance between them, which is the amount ρ must be reduced for them to be equivalent, must be 0. If patterns A and B are categorized by nodes $F2_A$ and $F2_B$ then it must be shown that $F2_A = F2_B$ only when $A = B$. If we assume that patterns A and B create different classifications, that is $F2_A \neq F2_B$, then there must be a set of weights in the network $\mathbf{W_A}$ and $\mathbf{W_B}$ leading from the input layer to the classification nodes A and B respectively. However, these weight matrices cannot be equivalent due to the training method used in this work.

Fast learning ($\beta = 1.0$) was used in all cases in this work, so the training algorithm for weights simplifies to:

$$\mathbf{W_J}^{(new)} = \mathbf{I} \wedge \mathbf{W_J}^{(old)} \tag{5.1}$$

where I is the input pattern and $\mathbf{W_J}$ is the weight matrix leading to classification node J. This also implies that each weight matrix leading to node J is modified only once during training. After a pattern has been classified using this method, it will alway be classified by the same F2 node. The activation formula for classification nodes $T_j$is:

$$T_j(\mathbf{I}) = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} \tag{5.2}$$

where α is a constant as is $|\mathbf{w}_j|$ which is equal to the width of the input pattern when complement coding is used. The selection criteria for a winning F2 node is to choose the maximum T_j value. The value T_j is therefore maximized when $|\mathbf{I} \wedge \mathbf{w}_j|$ is its largest and this can only occur when input pattern I is equal to weight matrix $\mathbf{w}_j$.

Given that the maximum F2 node is that with the weight matrix equal to the input pattern it is possible to show that the first requirement for a distance measure is met. There cannot be two equivalent weight matrices leading to different F2 nodes because the network cannot categorize the same input patterns differently. The selection formula 5.2 is used throughout training and after it has adjusted a weight matrix to categorize input pattern A, then pattern B will also be associated with the same F2 node. The distance between A and B is therefore 0 because ρ was not decreased in order to activate both categories in F2.

The second requirement for a distance measure is that given points A and B the distance AB and BA are equivalent. To show that the distances are equivalent it must be proven that the amount ρ is decreased to activate both nodes is equivalent given either A or B as the initial classification. When ρ is decreased in order to allow more F2 nodes to become active the equation:

$$\frac{|\mathbf{I} \wedge \mathbf{w}_j|}{|\mathbf{I}|} \geq \rho \tag{5.3}$$

is used to determine which nodes are considered active. As ρ decreases, more nodes are activated. Given that $\mathbf{I}$ will be equal to $\mathbf{w_J}$ for any initial classification J we can say that

each input pattern A and B will be equal to their respective weight matrices $\mathbf{W_A}$ and $\mathbf{W_B}$. Given an initial classification in $F2_A$, the degree υ to which the vigilance must be reduced for the $F2_B$ classification node to become active would therefore be:

$$\frac{|\mathbf{I_A} \wedge \mathbf{w}_B|}{|\mathbf{I_A}|} \geq (\rho - \upsilon_B) \tag{5.4}$$

for input pattern A and weight matrix associated with pattern B. If the inverse situation occurred then the value of υ_A required to activate classification node A given the initial activation generated by input pattern B would be:

$$\frac{|\mathbf{I_B} \wedge \mathbf{w}_A|}{|\mathbf{I_B}|} \geq (\rho - \upsilon_A) \tag{5.5}$$

The values $|\mathbf{I_A}|$ and $|\mathbf{I_B}|$ will both be constant and equal due to compliment coding so the denominator in the above will not have an effect on the classification. We know that $\mathbf{I_A} = \mathbf{w_A}$ and $\mathbf{I_B} = \mathbf{w}_B$ so the numerators in the above equations 5.4 and 5.5 must be equal which leads to:

$$\frac{|\mathbf{I_A} \wedge \mathbf{w}_B|}{|\mathbf{I_A}|} = \frac{|\mathbf{I_B} \wedge \mathbf{w}_A|}{|\mathbf{I_B}|} \geq (\rho - \upsilon) \tag{5.6}$$

The amount ρ must be reduced given either A or B as an initial classification in order to classify the other pattern will therefore be the same.

The third property which must be present in a distance measure is that given A, B, and C then the distances $AB + BC \geq AC$ (the triangle inequality). This property can be demonstrated, but only within the context in which distances are measured in this network. The operation of adding two distances $AB + BC$ involves reducing the ρ value until pattern B becomes active after the initial activation of pattern A. The ρ value is then again reduced until pattern C becomes active. This can be achieved through a reduction of ρ by υ_B at which point category B activates, followed by another reduction of ρ by a value of υ_C to activate category C. The same effect can be achieved by starting with a classification of A and then through the reduction of υ_C to activate C. The number of categories which become active while reducing ρ has no effect on the size of υ which must be used to activate a desired node. Therefore the C pattern will require the same υ_C value given any number

of intermediate patterns which become active between A and C. The distance from A to C is the sum of the distances between A and the intermediate pattern, $v_A + v_B = v_C$.

There are several characteristics of a distance based system which are desirable. Distance is a transformation of the original data which reflects the similarity between the patterns being compared. This allows us to draw some conclusions about the similarity between patterns based upon the distance. Obviously, similar and dissimilar patterns will produce small and large distances respectively. An intermediate distance value indicates that the patterns have some similarity, but there may be something preventing the value from reaching a more informative value. The cause of this is always something which generates testing data which differs greatly from the training data. Sources of such changes in the data include new cycles, translations, and low levels of noise in the testing data. In either case, the distance value is a translation of the original data patterns which allows easy training and comparison within a neural network.

The method by which distance is used in this network allows the system to avoid some of the problems associated with other neural techniques. Networks which use gradient descent or energy minimization techniques have several problems which are not found in the system which was developed herein. This is due to the method in which a distance measure is used to calculate similarity between sequences. Problems addressed by the distance measure specific to this work are:

- susceptibility to being trapped in locally optimal solutions
- expensive energy minimization methods
- oscillations in weight values due to conflicting data
- unstable behavior when presented with novel data

After the distance between categories has been determined the distance values internal to the network are updated. This is done by modifying the distances relative to their similarity to the current input pattern. The α, β, and γ values are used to modify the

distances based upon this similarity. The distance value generated is therefore not directly applied to the values stored within the network, but is instead simplified to be one of four values which is applied. This allows for simpler processing in the network, which is one of the goals of this work, and also creates several parameters (α, β, γ) which must be set before training can occur in the network.

The three distance modifiers are applied such that they create balanced distance values within the network. Matching patterns have their distance decremented by either α or β which indicates the similarity of patterns, and the dissimilar patterns have their distance increased by γ. The choice of both incremental and decremental modifiers is such that the distance values in the network cannot saturate at a maximum or minimum value and be fixed there indefinitely. This allows for multiple learned patterns in the network to become active during testing. The most appropriate subsequence at a given time will have its distance value degrade when it becomes a poorer match than another subsequence.

5.2 Number of Steps to Refocus

The focus on the training set is the element from the training sequence which currently has the smallest distance value within the network. The number of time steps which the network requires to change the focus from one training sequence to another non-adjacent element is the time required for the network to refocus. The number of time steps for the network to refocus can be calculated using the step size for each time step taken from the processing model:

$$\Delta = \begin{cases} d_j - \alpha & \text{if } x_i = y_j \\ d_j - \beta & \text{if } |x_i - y_j| < \delta_1 \\ d_j & \text{if } \delta_1 \leq |x_i - y_j| \leq \delta_2 \\ d_j + \gamma & \text{if } |x_i - y_j| > \delta_2 \end{cases} \tag{5.7}$$

where each distance value can change in magnitude by either α, β, γ or 0. The x_i and y_i are the classifications time stamp values as determined with the distance measures, x_i is the original classification, and y_i is a similar classification. This calculation also required

the starting values for each distance values $d_1(t)$ and $d_2(t)$ at time t. The initial minimum distance $d_1(t)$ must increase and the initial distance $d_2(t)$ must decrease to the point when $d_1(t+n) > d_2(t+n)$ in n time steps for the focus to change to distance d_2.

The number of time steps n to refocus the system is therefore:

$$|d_1(t) - d_2(t)| \geq \sum_{i=1}^{n} (\Delta_1(t+i) + \Delta_2(t+i)) \tag{5.8}$$

where $\Delta_1(t+i)$ is the change made to distance d_1 at time $t+i$ and $\Delta_2(t+i)$ is the change in d_2 also at time $t+i$. The number of time steps to refocus the system is n.

The formula for the maximum number of time steps can be determined by assuming the initial values for the distances are at their minimum and maximum values of 0 and 1. The left-hand-side then becomes:

$$|d_1(t) - d_2(t)| = |0 - 1| = 1 \tag{5.9}$$

and the above inequality is now:

$$1 \geq \sum_{i=1}^{n} (\Delta_1(t+i) + \Delta_2(t+i)) \tag{5.10}$$

and represents the maximum number of time steps n for the system to refocus.

It is desirable to analyze the above system when it is presented with data sets containing a variety of characteristics. This would result in the ability to discard some of the terms in equation 5.7, since they do not occur in all data, and to draw a stronger conclusion about the number of time steps required to refocus. This is not possible for all data. If the data is cyclical and not diverging then all of the α, β, and γ must remain because of the high likelihood of the testing values overlapping with the trained values. If the testing values are diverging from a training sequence and the data is not cyclical then stronger statements about the time to refocus can be made.

The number of time stamps given that the training and testing patterns are diverging and not staying relatively stationary to each other can be calculated. Assuming the distance

d_1 is rapidly diverging and distance d_2 is rapidly converging to the testing data then the number of steps n would be:

$$1 \geq \sum_{i=1}^{n} (\gamma + \alpha) \tag{5.11}$$

Alternatively, if d_1 is slowly diverging and d_2 is slowly converging then the number of time steps to refocus n would be:

$$1 \geq \sum_{i=1}^{n} 2\beta \tag{5.12}$$

In both cases, if distance d_1 does not diverge then d_2 is not guaranteed to decrease to a value less than d_1.

5.3 Complexity of the Algorithm

The rate sequence processing network operates in a linear time, $O(n)$, during training and in constant time, $O(1)$, during recall. The training time is bounded by the number of presentation of the training data set. The n data items must be processed by the system once during training. During testing the system simultaneously compares all learned patterns to the current input element in a single pass. Recall therefore occurs in a constant time which is not linked to the size n of the training set. It must be stressed that recall in this system occurs in parallel and this accounts for the low time complexity. If the system is simulated on a serial then the algorithm will operate in linear time as it must compare the current input to each learned pattern.

5.4 Stability in Training and Testing

The stability of a network can be described in terms of both its training and recall algorithms. The training algorithm developed in this work does not use the same model to develop associations between the input and output layers as many popular algorithms. It is therefore not subject to the stability problems associated with these other methods. The

training algorithm adjusts each weight once and does not change it thereafter. The weight changes indicate an association between the input to hidden, and the hidden to output layers. If two values need to be assigned to the same weight, indicating more than one association between the same input-output pair, then a new weight is selected by advancing to a different hidden node. Because there is no gradient descent the algorithm is not subject to settling upon a local optimum as is back-propagation. The Stability-Plasticity Dilemma described by Carpenter and Grossberg [CG91d] also poses no problem to this network due to its unique encoding of each occurrence. The network encodes all inputs explicitly and it therefore does not suffer from the problem of being too stable or too plastic given a novel input pattern. There is no stochastic element to the training mechanism so the system does not require a noise component to move it from local solutions as does the Boltzmann machine.

Stability during testing is also assured by this architecture due to the dynamics of the distance measure used to select the winning output. The network will always return a subsequence which it was trained upon. Given the number of time steps which the network must traverse before it is capable of changing the subsequence (as shown above) the system will return a stable subsequence from the training set. The latency of the distance measure helps to ensure that the output subsequences cannot cycle too rapidly. The network cannot rapidly shift output patterns to different subsequences so the output is at least a group of subsequences taken from the training patterns. If the network is used solely for prediction, meaning that new inputs are generated by the system's own output, then the network will rapidly settle upon a pattern which it was trained upon and will recreate that as the output. In this case there will be no instability in the network because it is simply recreating a pattern it was trained upon. If the network is instead presented with new inputs throughout the testing, which are not generated from its own outputs then the system can generate a series of different subsequences taken from the training set. The subsequences are selected based upon their similarity to the input patterns presented during testing. As the system must always return a sequence presented to it during training, it is therefore impossible

for it to generate an unstable output unless the original training data itself represented an unstable sequence. In this case, the only concern is that the subsequences taken from the training set do not cycle so rapidly that they generate a series of too short output sequences.

Long term stability occurs because the network will always return a sequence it has been trained upon. This prevents the system from generating an unstable or chaotic output. Short term stability is a feature of the latency derived from the distance measure. As the distance measure for a particular sequence must be overcome before a different sequence can appear on the output it is not possible for isolated noise to force the system to become unstable.

The stability of the distance measures themselves is a product of two of its features. Distance measures are inherently transformations of the original data into simpler forms. It is therefore not likely the distance measure will become unstable given a stable training set. If the distance measure does generate unstable or chaotic values then it is modeling the behavior of the testing data in relation to the training set. The distance measure used in this work is also bounded to prevent values from becoming excessive in size which also aids in the predictable behavior of the network.

5.5 Distance Measure Characteristics

There are several desirable characteristics of the distance based network. The distance measure indicates which of several training sequences is most similar to the current testing sequence. It also provides a measure of similarity between the testing sequence and all other learned sequences. This was not explored in this work, but is potentially useful.

The system's behavior when the closest matching sequence diverges from the input which causes the system to refocus on a new training sequence is predictable. The number of input patterns which the network requires to change focus can be predicted using the network parameters.

The method of storing temporal sequences as a set of ordered spatial representations

allows the system to be easily extensible. Adding new sequences, after the initial training, does not degrade those already learned. Training is easily achieved through encoding new sequences with their appropriate time stamp values. Converting temporal events into spatial events therefore creates a flexible method for storing sequences.

5.6 Analysis of Testing

The SOM based network was not as successful as the ART network when used to identify similar classifications. There are two likely reasons for this poorer performance and both are related to the method which SOMs use to organize their classification layer. When using the SOM with a two-dimensional classification layer there are always eight neighbors adjacent to any classification. This forces eight distance values to be updated using the β parameter given any input. The ART network generally produced two or three neighboring classifications given the same circumstances. The larger number of related nodes created by the SOM has a detrimental effect on the performance of the sequential network. A second problem arising from the SOMs method of categorization is its ability to store related patterns in topologically dissimilar locations. This is particularly a problem in the SOM with a one-dimensional classification layer. The SOM does generate a topology preserving classification map, but it does not guarantee that the neighboring classifications must be those with the minimum difference from the input pattern. It is possible to develop more than one area in the classification layer which is similar to the input pattern. When using a one-dimensional classification layer this is almost guaranteed. This has the effect of not modifying the distance values which are most appropriate for the given input pattern and as a result degrades the sequential network's performance.

For the Santa Fe data set A the parameters which produces the best results were, $\alpha = 0.01, \beta = 0.1, \gamma = 0.5$. The time to refocus, given diverging sequences and those parameters, can be determined using equation 5.11. For the system using Fuzzy ART categorization networks, the number of time steps for rapidly diverging sequences is 2 and

the number of time steps for a gradual divergence can be calculated using equation 5.12 as 50. A similar calculation can be made for Santa Fe data set D. The parameter values which produced the best results with Fuzzy ART categorization are $\alpha = 0.01, \beta = 0.5, \gamma = 0.3$. which produces a rapidly diverging number of time steps of 4 and a slow divergence of 2 time steps. This produces the startling result that the network will take more time steps to refocus for a rapidly diverging sequences than for a slowly diverging sequence. This result is not as surprising as it first appears. Data set D appears to oscillate between high and low values at a relatively slow frequency and also contains higher frequency oscillations superimposed over this pattern, (see figure 4.6). The number of time steps for rapidly diverging sequences suggests that the network does not quickly refocus given an apparent shift in sequences because of the dynamic nature of the high frequency oscillations which serve to generate the false appearance of a change in focus. The network does not refocus quickly because the data contains misleading features which suggest a change has occurred when the no change in focus may be required. The system therefore waits for enough inputs to help ensure that unnecessary refocusing does not occur.

The results for refocusing the network given the SOM based classifiers require more analysis. The number of time steps required for the network to refocus for rapidly and slowly diverging sequence in Santa Fe data set A are both two time steps. The same is true for Santa Fe data set D. However, as mentioned above the SOM based network was not as successful as those based on the ART network. The parameters which produce the best results for data set A are $\alpha = 0.9, \beta = 0.01, \gamma = 0.01$ and for data set D are $\alpha = 0.7, \beta = 0.01, \gamma = 0.01$. These suggest that the β and γ values had little to no effect on the operation of the network. It appears that only exact matches between the input pattern and the training data produced the best results. This suggests that the network was attempting to match the input immediately to a relatively similar training value and that it is not building a matching over a number of time steps. The network is guessing at a new solution with each time step using only the proximity to the current input pattern as a matching criteria. It appears that the SOM based network produces classifications which

do not take advantage of all of the distance modifiers which then leads to poor performance by the sequential network.

5.7 Future Work

More development and analysis of this system is required to improve its performance with continuous data and to better understand its behavior with different types of data. Topics for future examination include:

- A means to estimate parameter values is needed. The parameters are currently the most difficult feature of the network to estimate. Future work will involve an effort to identify some property of the data which can be used to derive the values of the parameters α, β, and γ.
- The inability of the network to recognize trends which progress outside the training data should also be addressed. A relatively simple solution would be the addition of a moving average subnetwork to the system which maintains trend information. This can be achieved through the use of a set of simple recurrent nodes which feedback some portion of their current value and sum it with the current input value.
- The mechanism for deciding which sequence node to choose when there are several alternatives is open to more development. The selection of the lowest indexed node has a precedence in other networks, such as ART, but it can be refined to use more information.

5.8 Summary

The purpose of this work is to create a temporal neural network which is capable of modeling temporal patterns. These patterns are to be presented to the network one element at a time, without the benefit of input buffering. Two existing networks, fuzzy ART or SOM,

are used in conjunction with a newly developed system to achieve this goal. The system uses a distance measure to identify the similarity between training and testing sequences. It is capable of choosing the training pattern which is most similar to the testing data based upon this distance measure.

Testing demonstrates that the network is capable of learning and recalling arbitrary sequences of input patterns and associating them with a set of output patterns. The system demonstrates several desirable characteristics including the ability to match similar patterns based upon a distance measure, predictable behavior when selecting identifying a sequence, the ability to learn and recall very long sequences, and it is resistant to being caught in a local solution.

The network does not build a model of the characteristics found in the training data, but instead acts as a memory and learns the training sequences. These sequences act as exemplars during testing and are used to match against new input sequences. A time-stamp value is associate with each input-output pattern pair which is stored in the network weights and represents the relative timing of each event in the sequence. As a result of this method the system learns and recalls patterns very quickly and will always produce output sequences which are similar to the training set. It will not create random or chaotic patterns when presented with novel patterns. The system is inherently parallel which allows it to benefit from a parallel hardware architecture.

The strengths of this system are derived from its accuracy, and its ability to operate in a parallel hardware environment. A primary benefit of the neural architecture is its use of large numbers of simple processing elements which operate in parallel. This system is appropriate for applications where the speed obtained through parallel processing is required and the cost of many processing elements is not a deterrent. Although some of the traditional techniques examined in this work proved to generate better results, this system's ability to forecast was comparable or better in all cases. Due to the system being a memory it generated excellent results when modeling the training data set. The accuracy of the system for modeling and prediction is generally as good or better as the other systems

which are examined in this work and that it is as appropriate for use in the same situations as these other methods. In addition, the system will not become unstable and produce chaotic output when used to generate long term forecasts. This is due to the network's method of explicit storage and recall of training data. The recurrent neural networks examined in this work generally do not have this property. The system is therefore appropriate for situations where long term forecasting is desired. Short term stability is also a feature of this system due to the latency in the distance measure which allows sporadic inputs in the form of noise or unexpected values to be absorbed by the system. This prevents rapid oscillations between sequences which can lead to unstable output from the network.

The environment in which the system is best suited for is that in which the training set is sufficiently representative of the problem for the network to build a complete representation of the data. As the network does not generalize based upon the training data it should only be used in situations where there is enough data to represent the complete system being modeled. The current incarnation of the network requires some tuning of parameters during training. As there is no clearly defined technique to generate these parameter it is necessary that this technique be used in conditions which allow configuration of the network prior to its use. Traditional techniques benefit from a large number of data analysis methods to generate parameters. Similar techniques have not yet been developed for this system. It may then be appropriate to use a traditional time-series method if early parameter configuration is required.

Work involving this network continues to develop an easier method of predicting the parameter values and to improve its ability to predict values which appear outside its training set.

Bibliography

[AGV94] Bernard Ans, Yves Coiton Jean-Claude Gilhodes, and Jene-Luc Velay. A neural network model for temporal sequence learning and motor programming. *Neural Networks*, 7:1461–1476, 1994.

[APR90] James A. Anderson, Andras Pellionisz, and Edward Rosenfeld, editors. *Neurocomputing 2 - Directions of Research.* MIT Press, 1990.

[AR88] James A. Anderson and Edward Rosenfeld, editors. *Neurocomputing - Foundations of Research.* MIT Press, 1988.

[Bar95a] G. Bartfai. An art-based modular architecture for learning hierarchical clusterings. Technical Report CS-TR-95/3, Victoria University of Wellington, 1995.

[Bar95b] G. Bartfai. A comparison of two art-based neural networks for hierarchical clusterings. Technical Report CS-TR-95/11, Victoria University of Wellington, 1995.

[BB91] Ian Boardman and Daniel Bullock. A neural network model of serial order recall from short-term memory. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 879–884, 1991.

[BCG92] Gary Bradski, Gail A. Carpenter, and Stephen Grossberg. Working memory for storage and recall of arbitrary temporal sequences. In *Proceedings of the*

International Joint Conference on Neural Networks, volume 2, pages 57–62, 1992.

[BJ76] George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forcasting and Control.* Holden-Day Inc., 1976.

[BW96] Guszti Bartfai and Roger White. Art-based modular neural networks for incremental learning of hierarchical clusterings. Technical Report CS-TR-96/11, Victoria University of Wellington, 1996.

[Car91] Gail A. Carpenter. Neural network models for pattern recognition and associative memory. In Gail A. Carpenter and Stephen Grossberg, editors, *Pattern Recognition by Self-Organizing Neural Networks.* MIT Press, 1991.

[Car94] Gail A. Carpenter. A distributed outstar network for spatial pattern learning. *Neural Networks*, 7:159–168, 1994.

[CG91a] Gail A. Carpenter and Stephen Grossberg. Art 2: Self-organization of stable category recognition codes for analog input patterns. In Gail A. Carpenter and Stephen Grossberg, editors, *Pattern Recognition by Self-Organizing Neural Networks.* MIT Press, 1991.

[CG91b] Gail A. Carpenter and Stephen Grossberg. Art 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. In Gail A. Carpenter and Stephen Grossberg, editors, *Pattern Recognition by Self-Organizing Neural Networks.* MIT Press, 1991.

[CG91c] Gail A. Carpenter and Stephen Grossberg. Artmap: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. In Gail A. Carpenter and Stephen Grossberg, editors, *Pattern Recognition by Self-Organizing Neural Networks.* MIT Press, 1991.

[CG91d] Gail A. Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. In Gail A. Carpen-

ter and Stephen Grossberg, editors, *Pattern Recognition by Self-Organizing Neural Networks*. MIT Press, 1991.

[CGM92] Gail A. Carpenter, Stephen Grossberg, and Natalya Markuzon. Fuzzy artmap: An adaptive resonance architecture for for incremental learning of analog maps. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 309–314, 1992.

[CGR94] Gail A. Carpenter, Stephen Grossberg, and David B. Rosen. Fuzzy art: An adaptive resonance algorithm for rapid, stable classification of analog patterns. In *Proceedings of the IEEE International Conference on Neural Networks*, volume II, pages 411–416, 1994.

[Cle93] Axel Cleeremans. *Mechanisms of Implicit Learning: Connectionst Models of Sequence Processing*. Bradford Books, MIT Press, 1993.

[CR93] Gail A. Carpenter and William D. Ross. Art-emap: A neural network architecture for object recognition by evidence accumulation. Technical Report CAS/CNS-TR-93-035, Department of Cognitive and Neural Systems, Boston University, 1993.

[dVP92] Bert de Vries and Jose C. Principe. The gamma model: a new neural model for temporal processing. *Neural Networks*, 5:565–576, 1992.

[Elm90] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[FS92] James A. Freeman and David M. Skapura. *Neural Networks - Algorithms, Applications, and Programming Techniques*. Addison-Wesley Publishing Company, 1992.

[Gje92] Robert O. Gjerdingen. Learning syntactically significant temporal patterns of chords: A masking field embedded in an arrt 3 architecture. *Neural Networks*, 5:551–564, 1992.

[Gro78] Stephen Grossberg. A theory of human memory: Self-organizing and performance of sensory-motor codes, maps, and plans. In Robert Rosen and Fred M. Snell, editors, *Progress in Theoretical Biology.* Academic Press, 1978.

[Gro86] Stephen Grossberg. The adaptive self-organization of serial order in behavior: Speech, language, and motor control. In Eileen C. Schwab and Howard C. Nusbaum, editors, *Pattern Recognition by Humans and Machines.* Academic Press, 1986.

[Gro90] Stephen Grossberg. Self-organizing neural architectures for vision, learning, and robotic control. In *Proceedings of the International Neural Network Conference*, pages 797–800. Kluwer Academic Publishers - Netherlands, 1990.

[Gro91] Stephen Grossberg. Nonlinear neural networks: Principles, mechanisms, and architectures. In Gail A. Carpenter and Stephen Grossberg, editors, *Pattern Recognition by Self-Organizing Neural Networks.* MIT Press, 1991.

[Har75] J.A. Hartigan. *Clustering Algorithms.* John Wiley and Sons, 1975.

[Har93] Andrew C. Harvey. *Time Series Models.* MIT Press, 1993.

[HGH94] Gregory L. Heileman, Michael Georgiopoulos, and Juxin Hwang. A survey of learning results for art1 networks. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1222–1225, 1994.

[HH93] Don R. Hush and Bill G. Horne. Progress in supervised neural networks: What's new since lippmann. *IEEE Signal Processing Magazine*, pages 8–39, 1993.

[HKP91] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation.* Addison Wesley, 1991.

[HM94] K.W. Hipel and A.I. McLeod. *Time Series Modelling of Water Resources and Environmental Systems.* Elsevier Science B.V., 1994.

[Koh89] Teuvo Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, 1989.

[Lau92] Clifford Lau, editor. *Neural Networks - Theoretical Foundations.* IEEE Press, 1992.

[Lip87] Richard P. Lippmann. An introduction to computing with neural networks. *IEEE ASSP Magazine*, pages 4–22, 1987.

[MD90] Christian Mannes and Georg Dorffner. Self-organizing detectors for spatiotemporal sequences. In Jörg Kinderman and Alexander Linden, editors, *Distributed Adaptive Neural Information Processing - Proceedings of a Workshop at SchloßBirlinghoven*, pages 89–102. R. Oldenbourg Verlag, 1990.

[Mes83] Bruce E. Meserve. *Fundamental Concepts of Geometry.* Dover Publications Inc., 1983.

[MK85] D. A. Messuri and C. A. Klein. Automatic body regulation for maintaining stability of a legged vehical during rough-terrain locomotion. *IEEE Journal of Robotics and Automation*, RA-1(3):132–141, 1985.

[Moz89] Micheal C. Mozer. A focused backpropagation algorithms for temporal pattern recognition. *Complex Systems*, 3:349–381, 1989.

[Nig93] Albert Nigrin. *Neural Networks for Pattern Recognition.* MIT Press, 1993.

[NPVB90] Stefano Nolfi, Domenico Parisi, Giuseppe Vallar, and Cristina Burani. Recall of sequences of items by a neural network. In *Proceedings of the 1990 Connectionist Models Summer School*, pages 243–252, 1990.

[Por91] Roger Porkess. *The Harper Collins Dictionary of Statistics.* HarperPerennial, 1991.

[RMtPRG86a] David E. Rumelhart, James L. McClelland, and the PDP Research Group. *Parallel Distributed Processing - Explorations in the Microstructure of Cognition Volume 1: Foundations.* MIT Press, 1986.

[RMtPRG86b] David E. Rumelhart, James L. McClelland, and the PDP Research Group. *Parallel Distributed Processing - Explorations in the Microstructure of Cognition Volume 2: Psychological and Biological Models.* MIT Press, 1986.

[RW91] T. W. Ryan and C. L. Winter. Variations on adaptive resonance. In Gail A. Carpenter and Stephen Grossberg, editors, *Pattern Recognition by Self-Organizing Neural Networks.* MIT Press, 1991.

[Sch92] Robert Schalkoff. *Pattern Recognition - Statistical, Structural, and Neural Approaches.* John Wiley and Sons, Inc., 1992.

[Sch96] Jürgen Schürmann. *Pattern Classification: A Unified View of Statistical and Neural Approaches.* John Wiley and Sons, Inc., 1996.

[Sim90] Patrick K. Simpson. *Artificial Neural Systems - Foundations, Paradigms, Applications, and Implementations.* Pergamon Press, 1990.

[SR86] Terrence J. Sejnowski and Charles R. Rosenberg. Nettalk: A parallel network that learns to read aloud. Technical Report JHU/EECS-86/01, Johns Hopkins University, 1986.

[SSCM88] David Servan-Schreiber, Axel Cleermans, and James L. McClelland. Encoding sequential structures in simple recurrent networks. Technical Report CMU-CS-88-183, Carnegie Mellon University, 1988.

[Tuk77] John Wilder Tukey. *Exploratory Data Analysis.* Addison-Wesley Publishing Company, 1977.

[WA90] Deliang Wang and Michael A. Arbib. Complex temporal sequence learning based on short-term memory. In *Proceedings of the IEEE*, pages 1536–1543, 1990.

[Wer90] Paul J. Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, October 1990.

[WG94] Andreas S. Weigend and Neil A. Gershenfeld, editors. *Time Series Prediction: Forcasting the Future and Understanding the Past.* Addison-Wesley Publishing Company, 1994.

[YB93] Brian M. Yamauchi and Randall D. Beer. Sequential behavior and learning in evolved dynamical neural networks. Technical Report CES-93-25, Case Western Reserve University, 1993.

[ZT94] Xiru Zhang and Kurt Thearling. Non-linear time-series predictions by systematic data exploration on a massively parallel computer. Technical Report 94-07-045, Santa Fe Institute, 1994.

Appendix A

Tabular Results

A.1 Generated Data Sets

*v*igilance	*B*eta									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.05	1	3	2	2	1	1	0	0	0	0
0.1	1	3	2	2	1	1	0	0	0	0
0.2	1	3	2	2	1	1	0	0	0	0
0.3	1	3	2	2	1	1	0	0	0	0
0.4	1	3	2	2	1	1	0	0	0	0
0.5	1	3	2	2	1	1	0	0	0	0
0.6	1	3	2	2	1	1	0	0	0	0
0.7	1	3	2	2	1	1	0	0	0	0
0.8	1	3	2	2	1	1	0	0	0	0
0.9	5	3	2	2	1	1	0	0	0	0
0.99	14	39	32	26	17	17	13	1	0	0

Table A.1: Ramp data - number of times no solution was found during testing based on Beta and vigilance values (Epochs = 1)

*v*igilance	*B*eta									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.05	4	2	2	1	2	1	0	0	0	0
0.1	4	2	2	1	2	1	0	0	0	0
0.2	4	2	2	1	2	1	0	0	0	0
0.3	4	2	2	1	2	1	0	0	0	0
0.4	4	2	2	1	2	1	0	0	0	0
0.5	4	2	2	1	2	1	0	0	0	0
0.6	4	2	2	1	2	1	0	0	0	0
0.7	4	2	2	1	2	1	0	0	0	0
0.8	4	2	2	1	2	1	0	0	0	0
0.9	4	2	2	1	2	1	0	0	0	0
0.99	11	9	7	6	5	3	1	0	0	0

Table A.2: Ramp data - number of times no solution was found during testing based on Beta and vigilance values (Epochs = 5)

vigilance	*Beta*									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.05	6	2	5	4	5	1	4	1	4	0
0.1	6	2	5	4	5	1	4	1	4	0
0.2	6	2	5	4	5	1	4	1	4	0
0.3	6	2	5	4	5	1	4	1	4	0
0.4	6	2	5	4	5	1	4	1	4	0
0.5	6	2	5	4	5	1	4	1	4	0
0.6	6	2	5	4	5	1	4	1	4	0
0.7	6	2	5	4	5	1	4	1	4	0
0.8	6	2	5	4	5	1	4	1	4	0
0.9	6	2	5	4	5	1	4	1	4	0
0.99	6	5	14	4	5	30	16	9	14	0

Table A.3: Mixed ramp data - number of times no solution was found during testing based on Beta and vigilance values (Epochs = 1)

vigilance	*Beta*									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.05	5	2	2	3	3	4	2	1	0	0
0.1	5	2	2	3	3	4	2	1	0	0
0.2	5	2	2	3	3	4	2	1	0	0
0.3	5	2	2	3	3	4	2	1	0	0
0.4	5	2	2	3	3	4	2	1	0	0
0.5	5	2	2	3	3	4	2	1	0	0
0.6	5	2	2	3	3	4	2	1	0	0
0.7	5	2	2	3	3	4	2	1	0	0
0.8	5	2	2	3	3	4	2	1	0	0
0.9	5	2	2	3	3	4	2	1	0	0
0.99	5	2	2	14	16	9	5	1	0	0

Table A.4: Mixed ramp data - number of times no solution was found during testing based on Beta and vigilance values (Epochs = 5)

vigilance	*Beta*									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.05	4	2	3	3	8	2	3	2	2	6
0.1	4	2	3	3	8	2	3	2	2	6
0.2	4	2	3	3	8	2	3	2	2	6
0.3	4	2	3	3	8	2	3	2	2	6
0.4	4	2	3	3	8	2	3	2	2	6
0.5	4	2	3	3	8	2	3	2	2	6
0.6	4	2	3	3	8	2	3	2	2	6
0.7	4	2	3	3	8	2	3	2	2	6
0.8	4	2	3	3	8	2	3	2	2	6
0.9	4	8	10	9	12	2	3	2	2	6
0.99	16	14	19	24	21	11	13	13	9	6

Table A.5: Sine data - number of times no solution was found during testing based on Beta and vigilance values (Epochs = 1)

vigilance	*Beta*									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.05	2	2	4	2	5	5	6	6	6	6
0.1	2	2	4	2	5	5	6	6	6	6
0.2	2	2	4	2	5	5	6	6	6	6
0.3	2	2	4	2	5	5	6	6	6	6
0.4	2	2	4	2	5	5	6	6	6	6
0.5	2	2	4	2	5	5	6	6	6	6
0.6	2	2	4	2	5	5	6	6	6	6
0.7	2	2	4	2	5	5	6	6	6	6
0.8	2	2	4	2	5	5	6	6	6	6
0.9	2	2	4	2	5	5	6	6	6	6
0.99	2	2	9	7	8	5	6	6	6	6

Table A.6: Sine data - number of times no solution was found during testing based on Beta and vigilance values (Epochs = 5)

A.2 Real data

vigilance	*Beta*									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.05	1	3	2	2	1	4	3	3	3	3
0.1	1	3	2	2	1	4	3	3	3	3
0.2	1	3	2	2	1	4	3	3	3	3
0.3	1	3	2	2	1	4	3	3	3	3
0.4	1	3	2	2	1	4	3	3	3	3
0.5	1	3	2	2	1	4	3	3	3	3
0.6	1	3	2	2	1	4	3	3	3	3
0.7	1	3	2	2	1	4	3	3	3	3
0.8	1	3	2	2	1	4	3	3	3	3
0.9	1	3	2	2	1	4	3	3	3	3
0.99	1	6	2	2	1	4	3	5	3	3

Table A.7: Data set A - first 100 points - number of times no solution was found during testing based on Beta and vigilance values (Epochs = 5)

vigilance	*Beta*									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.05	22	15	20	18	13	13	13	13	13	13
0.1	22	15	20	18	13	13	13	13	13	13
0.2	22	15	20	18	13	13	13	13	13	13
0.3	22	15	20	18	13	13	13	13	13	13
0.4	22	15	20	18	13	13	13	13	13	13
0.5	22	15	20	18	13	13	13	13	13	13
0.6	22	15	20	18	13	13	13	13	13	13
0.7	22	15	20	18	13	13	13	13	13	13
0.8	22	17	20	18	13	13	13	13	13	13
0.9	46	31	24	23	18	18	13	13	13	13
0.99	122	61	33	52	31	30	13	13	13	13

Table A.8: Data set A - first 1000 points - number of times no solution was found during testing based on Beta and vigilance values (Epochs = 5)

A.3 Processing Model - Data Set A Summary Results

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.037	0.037	0.037	0.037	0.037	0.037	0.037
0.10	0.342	0.149	0.102	0.102	0.102	0.102	0.102
0.30	0.748	0.510	0.257	0.234	0.230	0.226	0.226
0.50	1.142	0.446	0.367	0.367	0.368	0.368	0.368
0.70	1.272	0.762	0.398	0.369	0.367	0.368	0.368
0.90	1.301	0.937	0.561	0.373	0.373	0.367	0.368
0.99	1.256	1.012	0.548	0.279	0.272	0.272	0.264

Table A.9: Processing model - data set A results - $\alpha = 0.01$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.349	0.043	0.043	0.043	0.043	0.043	0.043
0.10	0.405	0.164	0.117	0.117	0.117	0.117	0.117
0.30	0.809	0.535	0.254	0.233	0.233	0.216	0.216
0.50	1.237	0.452	0.369	0.367	0.368	0.368	0.368
0.70	1.296	0.762	0.405	0.367	0.365	0.368	0.368
0.90	1.342	0.939	0.560	0.309	0.303	0.295	0.291
0.99	1.299	1.012	0.548	0.309	0.303	0.303	0.295

Table A.10: Processing model - data set A results - $\alpha = 0.1$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.574	0.088	0.078	0.067	0.073	0.073	0.073
0.10	0.519	0.188	0.077	0.087	0.084	0.084	0.084
0.30	0.800	0.371	0.165	0.102	0.084	0.084	0.084
0.50	1.220	0.473	0.376	0.365	0.374	0.368	0.368
0.70	1.287	0.761	0.304	0.274	0.264	0.274	0.274
0.90	1.336	0.981	0.560	0.309	0.309	0.295	0.291
0.99	1.290	1.055	0.548	0.309	0.309	0.303	0.295

Table A.11: Processing model - data set A results - $\alpha = 0.3$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.711	0.156	0.097	0.097	0.109	0.109	0.109
0.10	0.531	0.187	0.113	0.095	0.102	0.105	0.105
0.30	0.841	0.445	0.150	0.097	0.100	0.109	0.109
0.50	1.159	0.393	0.250	0.242	0.231	0.231	0.231
0.70	1.110	0.658	0.270	0.244	0.242	0.231	0.231
0.90	1.156	0.812	0.492	0.249	0.249	0.242	0.231
0.99	1.110	0.879	0.477	0.249	0.249	0.249	0.242

Table A.12: Processing model - data set A results - $\alpha = 0.5$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.615	0.449	0.122	0.097	0.097	0.109	0.109
0.10	0.640	0.362	0.117	0.097	0.097	0.105	0.105
0.30	0.840	0.431	0.188	0.159	0.155	0.158	0.158
0.50	1.116	0.536	0.260	0.242	0.231	0.231	0.231
0.70	1.094	0.648	0.273	0.251	0.242	0.231	0.231
0.90	1.140	0.818	0.526	0.252	0.249	0.242	0.231
0.99	1.094	0.908	0.510	0.252	0.249	0.249	0.242

Table A.13: Processing model - data set A results - $\alpha = 0.7$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.620	0.421	0.166	0.104	0.104	0.097	0.109
0.10	0.620	0.473	0.207	0.199	0.199	0.193	0.193
0.30	0.821	0.655	0.220	0.193	0.194	0.191	0.191
0.50	1.109	0.541	0.246	0.245	0.234	0.231	0.231
0.70	1.087	0.645	0.258	0.254	0.245	0.231	0.231
0.90	1.142	0.795	0.504	0.252	0.252	0.242	0.231
0.99	1.096	0.884	0.510	0.252	0.252	0.249	0.242

Table A.14: Processing model - data set A results - $\alpha = 0.9$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.579	0.465	0.423	0.166	0.166	0.166	0.160
0.10	0.677	0.502	0.453	0.199	0.199	0.199	0.193
0.30	0.857	0.652	0.465	0.193	0.194	0.197	0.191
0.50	1.113	0.541	0.262	0.245	0.234	0.234	0.231
0.70	1.091	0.645	0.363	0.254	0.245	0.234	0.231
0.90	1.146	0.795	0.504	0.252	0.252	0.245	0.231
0.99	1.100	0.884	0.537	0.252	0.252	0.252	0.242

Table A.15: Processing model - data set A results - $\alpha = 0.99$

A.4 Processing Model - Data Set D Summary Results

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.584	0.098	0.099	0.099	0.098	0.098	0.098
0.10	0.371	0.257	0.101	0.097	0.098	0.097	0.097
0.30	0.197	0.134	0.131	0.130	0.130	0.130	0.130
0.50	0.193	0.166	0.154	0.154	0.152	0.152	0.152
0.70	0.201	0.187	0.162	0.158	0.154	0.152	0.152
0.90	0.198	0.198	0.171	0.162	0.162	0.154	0.152
0.99	0.196	0.199	0.172	0.159	0.159	0.159	0.153

Table A.16: Processing model - data set D results - $\alpha = 0.01$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	1.040	0.155	0.140	0.140	0.140	0.140	0.140
0.10	0.370	0.255	0.103	0.098	0.099	0.098	0.098
0.30	0.197	0.134	0.134	0.133	0.133	0.133	0.133
0.50	0.192	0.167	0.154	0.154	0.152	0.152	0.152
0.70	0.201	0.188	0.162	0.158	0.154	0.152	0.152
0.90	0.197	0.194	0.169	0.159	0.159	0.154	0.153
0.99	0.195	0.196	0.172	0.159	0.159	0.159	0.154

Table A.17: Processing model - data set D results - $\alpha = 0.1$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	1.261	0.333	0.163	0.154	0.150	0.150	0.150
0.10	0.307	0.267	0.127	0.114	0.115	0.115	0.115
0.30	0.193	0.138	0.134	0.134	0.133	0.133	0.133
0.50	0.192	0.168	0.156	0.154	0.153	0.153	0.153
0.70	0.201	0.186	0.162	0.156	0.153	0.152	0.152
0.90	0.197	0.195	0.170	0.159	0.159	0.154	0.153
0.99	0.196	0.197	0.173	0.159	0.159	0.159	0.154

Table A.18: Processing model - data set D results - $\alpha = 0.3$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.330	0.319	0.223	0.153	0.153	0.153	0.153
0.10	0.156	0.211	0.158	0.139	0.139	0.139	0.139
0.30	0.193	0.144	0.135	0.134	0.133	0.133	0.133
0.50	0.194	0.163	0.155	0.155	0.153	0.153	0.153
0.70	0.200	0.187	0.162	0.158	0.154	0.153	0.153
0.90	0.195	0.192	0.170	0.160	0.159	0.154	0.153
0.99	0.194	0.194	0.173	0.160	0.159	0.159	0.154

Table A.19: Processing model - data set D results - $\alpha = 0.5$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.193	0.243	0.168	0.160	0.153	0.153	0.153
0.10	0.185	0.187	0.143	0.143	0.142	0.141	0.141
0.30	0.196	0.146	0.142	0.137	0.135	0.134	0.134
0.50	0.196	0.167	0.154	0.155	0.154	0.153	0.153
0.70	0.201	0.185	0.162	0.158	0.155	0.153	0.153
0.90	0.196	0.192	0.170	0.161	0.160	0.154	0.153
0.99	0.195	0.194	0.174	0.161	0.160	0.159	0.154

Table A.20: Processing model - data set D results - $\alpha = 0.7$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.200	0.165	0.152	0.158	0.158	0.147	0.147
0.10	0.187	0.166	0.161	0.160	0.160	0.160	0.160
0.30	0.196	0.150	0.141	0.140	0.139	0.139	0.138
0.50	0.196	0.168	0.155	0.155	0.154	0.154	0.153
0.70	0.201	0.186	0.163	0.159	0.155	0.154	0.153
0.90	0.196	0.192	0.171	0.161	0.161	0.155	0.153
0.99	0.195	0.194	0.174	0.161	0.161	0.160	0.154

Table A.21: Processing model - data set D results - $\alpha = 0.9$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.176	0.155	0.150	0.149	0.149	0.149	0.149
0.10	0.205	0.164	0.161	0.160	0.160	0.160	0.160
0.30	0.196	0.150	0.141	0.140	0.139	0.139	0.139
0.50	0.196	0.168	0.155	0.155	0.154	0.154	0.154
0.70	0.201	0.186	0.163	0.159	0.155	0.154	0.154
0.90	0.196	0.192	0.171	0.161	0.161	0.155	0.154
0.99	0.195	0.193	0.177	0.161	0.161	0.161	0.155

Table A.22: Processing model - data set D results - $\alpha = 0.99$

A.5 Fuzzy Art - Data Set A Summary Results

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.942	0.586	0.406	0.406	0.406	0.406	0.406
0.10	1.268	1.193	0.715	0.265	0.268	0.268	0.268
0.30	1.137	1.195	0.937	0.874	0.711	0.341	0.341
0.50	0.936	1.129	0.929	0.959	1.062	1.010	1.010
0.70	0.936	1.101	0.892	1.005	1.021	1.010	1.010
0.90	1.025	1.180	1.011	1.069	1.205	1.205	1.265
0.99	0.975	1.180	1.011	1.069	1.205	1.205	1.204

Table A.23: Fuzzy ART - data set A results - $\alpha = 0.01$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.601	0.526	0.348	0.384	0.384	0.384	0.384
0.10	1.213	0.906	0.839	0.368	0.370	0.370	0.370
0.30	1.108	1.153	0.937	0.898	0.711	0.341	0.341
0.50	0.936	1.095	0.929	0.959	1.062	1.010	1.010
0.70	0.936	1.067	0.873	1.005	1.021	1.010	1.010
0.90	1.046	1.146	1.011	1.069	1.205	1.205	1.265
0.99	0.975	1.146	1.011	1.069	1.205	1.205	1.204

Table A.24: Fuzzy ART - data set A results - $\alpha = 0.1$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.587	0.445	0.362	0.355	0.384	0.384	0.384
0.10	1.197	0.817	0.863	0.362	0.364	0.364	0.364
0.30	1.108	1.225	0.982	0.921	0.811	0.441	0.441
0.50	0.936	1.133	0.912	0.959	1.062	1.010	1.010
0.70	0.957	1.105	0.892	1.005	0.997	1.019	1.010
0.90	1.046	1.132	1.011	1.069	1.205	1.205	1.265
0.99	0.975	1.132	1.011	1.069	1.205	1.205	1.204

Table A.25: Fuzzy ART - data set A results - $\alpha = 0.3$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.720	0.426	0.423	0.375	0.355	0.384	0.384
0.10	1.146	0.878	0.908	0.380	0.353	0.353	0.353
0.30	1.101	1.226	0.969	0.945	0.818	0.445	0.445
0.50	0.957	1.133	0.872	0.924	1.044	1.019	1.010
0.70	0.957	1.053	0.842	0.970	0.997	1.019	1.010
0.90	1.046	1.132	1.025	1.069	1.205	1.205	1.265
0.99	0.975	1.132	1.025	1.069	1.205	1.205	1.204

Table A.26: Fuzzy ART - data set A results - $\alpha = 0.5$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.720	0.443	0.596	0.423	0.375	0.348	0.384
0.10	1.216	0.975	0.917	0.552	0.426	0.403	0.403
0.30	1.075	1.221	0.915	0.882	0.804	0.447	0.437
0.50	0.957	1.081	0.872	0.924	1.044	1.019	1.010
0.70	0.957	1.053	0.842	0.970	0.997	1.019	1.010
0.90	1.046	1.132	1.025	1.055	1.205	1.205	1.265
0.99	0.975	1.132	1.025	1.055	1.205	1.205	1.204

Table A.27: Fuzzy ART - data set A results - $\alpha = 0.7$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.831	0.751	0.558	0.466	0.465	0.364	0.358
0.10	1.174	1.019	0.650	0.466	0.426	0.364	0.358
0.30	1.025	1.120	0.913	0.809	0.791	0.450	0.445
0.50	0.908	1.032	0.873	0.853	1.004	0.985	0.975
0.70	0.907	1.004	0.842	0.889	0.957	0.985	0.975
0.90	0.997	1.083	0.976	1.056	1.200	1.200	1.244
0.99	0.926	1.083	0.976	1.056	1.200	1.200	1.199

Table A.28: Fuzzy ART - data set A results - $\alpha = 0.9$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.838	0.751	0.558	0.466	0.465	0.364	0.358
0.10	1.174	1.019	0.650	0.466	0.426	0.364	0.358
0.30	1.025	1.120	0.903	0.809	0.791	0.450	0.445
0.50	0.908	1.032	0.873	0.853	1.004	0.985	0.975
0.70	0.907	1.004	0.842	0.889	0.957	0.985	0.975
0.90	0.997	1.083	0.976	1.056	1.200	1.200	1.226
0.99	0.926	1.083	0.976	1.056	1.200	1.200	1.199

Table A.29: Fuzzy ART - data set A results - $\alpha = 0.99$

A.6 Fuzzy Art - Data Set D Summary Results

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.293	0.361	0.327	0.305	0.309	0.304	0.300
0.10	0.303	0.172	0.279	0.227	0.198	0.221	0.209
0.30	0.322	0.159	0.143	0.184	0.178	0.182	0.202
0.50	0.272	0.149	0.132	0.169	0.157	0.162	0.162
0.70	0.272	0.150	0.129	0.169	0.155	0.162	0.162
0.90	0.270	0.144	0.133	0.151	0.140	0.140	0.141
0.99	0.270	0.144	0.131	0.151	0.140	0.140	0.140

Table A.30: Fuzzy ART - data set D results - $\alpha = 0.01$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.390	0.317	0.337	0.319	0.321	0.316	0.312
0.10	0.304	0.171	0.279	0.227	0.198	0.221	0.222
0.30	0.322	0.159	0.143	0.184	0.178	0.182	0.202
0.50	0.272	0.149	0.132	0.169	0.157	0.162	0.162
0.70	0.272	0.150	0.129	0.169	0.155	0.162	0.162
0.90	0.270	0.144	0.133	0.151	0.140	0.140	0.141
0.99	0.270	0.144	0.131	0.151	0.140	0.140	0.140

Table A.31: Fuzzy ART - data set D results - $\alpha = 0.1$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.343	0.338	0.362	0.311	0.306	0.295	0.291
0.10	0.303	0.171	0.279	0.227	0.198	0.221	0.222
0.30	0.322	0.159	0.140	0.184	0.178	0.182	0.203
0.50	0.272	0.150	0.132	0.156	0.157	0.162	0.162
0.70	0.271	0.150	0.129	0.169	0.155	0.162	0.162
0.90	0.270	0.144	0.133	0.151	0.140	0.140	0.141
0.99	0.270	0.144	0.131	0.151	0.140	0.140	0.140

Table A.32: Fuzzy ART - data set D results - $\alpha = 0.3$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.354	0.293	0.334	0.286	0.268	0.252	0.256
0.10	0.299	0.170	0.277	0.225	0.199	0.222	0.223
0.30	0.322	0.159	0.140	0.184	0.178	0.182	0.203
0.50	0.319	0.150	0.132	0.168	0.158	0.162	0.162
0.70	0.271	0.150	0.129	0.169	0.000	0.000	0.162
0.90	0.270	0.144	0.133	0.151	0.140	0.140	0.141
0.99	0.270	0.144	0.131	0.151	0.140	0.140	0.140

Table A.33: Fuzzy ART - data set D results - $\alpha = 0.5$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.314	0.293	0.260	0.279	0.264	0.265	0.265
0.10	0.296	0.170	0.274	0.223	0.197	0.220	0.221
0.30	0.321	0.159	0.140	0.184	0.178	0.182	0.203
0.50	0.271	0.150	0.132	0.168	0.157	0.162	0.162
0.70	0.271	0.150	0.129	0.169	0.155	0.162	0.162
0.90	0.270	0.144	0.133	0.151	0.140	0.140	0.141
0.99	0.270	0.144	0.131	0.151	0.140	0.140	0.140

Table A.34: Fuzzy ART - data set D results - $\alpha = 0.7$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.201	0.258	0.202	0.211	0.198	0.197	0.197
0.10	0.293	0.170	0.272	0.217	0.193	0.215	0.216
0.30	0.321	0.159	0.140	0.184	0.178	0.184	0.202
0.50	0.319	0.150	0.132	0.168	0.157	0.162	0.162
0.70	0.271	0.150	0.129	0.155	0.155	0.162	0.162
0.90	0.270	0.144	0.133	0.151	0.140	0.140	0.141
0.99	0.270	0.144	0.131	0.151	0.140	0.140	0.140

Table A.35: Fuzzy ART - data set D results - $\alpha = 0.9$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.197	0.254	0.202	0.211	0.198	0.197	0.197
0.10	0.293	0.170	0.272	0.217	0.193	0.215	0.216
0.30	0.321	0.159	0.140	0.184	0.178	0.182	0.202
0.50	0.271	0.150	0.132	0.168	0.157	0.162	0.162
0.70	0.271	0.150	0.129	0.169	0.155	0.162	0.162
0.90	0.270	0.144	0.133	0.151	0.140	0.140	0.141
0.99	0.270	0.144	0.131	0.151	0.140	0.140	0.140

Table A.36: Fuzzy ART - data set D results - $\alpha = 0.99$

A.7 SOM - Data Set A Summary Results

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.994	1.007	1.151	1.151	1.151	1.151	1.151
0.10	1.084	1.132	1.099	1.151	1.151	1.151	1.151
0.30	1.244	0.786	1.048	1.076	1.099	1.151	1.151
0.50	1.029	0.902	0.975	1.023	1.076	1.099	1.099
0.70	0.920	0.799	0.861	1.080	1.023	1.099	1.099
0.90	0.959	0.915	0.913	1.043	1.087	0.980	0.986
0.99	0.773	0.915	0.913	1.041	1.087	0.980	0.987

Table A.37: SOM - data set A results - $\alpha = 0.01$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.795	0.761	0.785	0.785	0.785	0.785	0.785
0.10	1.018	1.120	1.067	1.151	1.151	1.151	1.151
0.30	1.130	0.764	1.028	1.044	1.099	1.151	1.151
0.50	0.984	0.867	0.958	1.023	1.044	1.099	1.099
0.70	0.847	0.798	0.861	1.080	1.023	1.067	1.099
0.90	0.881	0.946	0.913	1.041	1.087	0.980	0.986
0.99	0.771	0.946	0.913	1.041	1.087	0.980	0.987

Table A.38: SOM - data set A results - $\alpha = 0.1$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.860	0.866	0.683	0.706	0.779	0.785	0.785
0.10	1.037	0.757	0.681	0.710	0.784	0.785	0.785
0.30	1.040	0.768	1.005	1.012	1.067	1.151	1.151
0.50	0.984	0.855	0.933	1.023	1.048	1.067	1.099
0.70	0.842	0.747	0.895	1.030	1.035	1.079	1.079
0.90	0.847	0.935	0.912	1.041	1.026	0.980	0.986
0.99	0.736	0.935	0.912	1.041	1.026	0.980	0.987

Table A.39: SOM - data set A results - $\alpha = 0.3$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.625	0.846	0.632	0.728	0.706	0.779	0.779
0.10	0.649	0.798	0.597	0.728	0.674	0.779	0.779
0.30	1.058	0.743	0.743	0.759	0.870	0.917	0.950
0.50	0.901	0.769	1.009	1.027	1.060	1.079	1.079
0.70	0.790	0.825	0.874	1.023	1.035	1.079	1.079
0.90	0.848	0.988	0.928	1.029	1.026	0.980	0.986
0.99	0.737	0.988	0.928	1.029	1.026	0.980	0.987

Table A.40: SOM - data set A results - $\alpha = 0.5$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.625	0.825	0.697	0.626	0.728	0.779	0.779
0.10	0.754	0.834	0.639	0.625	0.728	0.746	0.779
0.30	0.997	0.846	0.636	0.642	0.699	0.752	0.752
0.50	0.866	0.741	0.952	0.966	0.992	1.017	1.017
0.70	0.790	0.802	0.884	1.024	1.029	1.079	1.079
0.90	0.848	0.989	0.928	1.029	1.026	0.980	0.986
0.99	0.737	0.989	0.928	1.029	1.026	0.980	0.987

Table A.41: SOM - data set A results - $\alpha = 0.7$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.538	0.708	0.687	0.616	0.627	0.746	0.779
0.10	0.929	0.738	0.636	0.633	0.627	0.746	0.746
0.30	0.935	0.857	0.595	0.657	0.699	0.746	0.752
0.50	0.861	0.802	0.963	0.994	0.991	1.010	1.016
0.70	0.784	0.793	0.765	1.009	0.966	1.010	1.016
0.90	0.843	0.983	0.928	1.076	1.026	0.979	0.985
0.99	0.732	0.983	0.928	1.076	1.026	0.979	0.986

Table A.42: SOM - data set A results - $\alpha = 0.9$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.560	0.641	0.687	0.616	0.627	0.746	0.779
0.10	0.887	0.738	0.636	0.633	0.627	0.746	0.746
0.30	0.908	0.857	0.595	0.657	0.699	0.746	0.746
0.50	0.877	0.802	0.963	0.994	0.991	1.010	1.010
0.70	0.800	0.793	0.765	1.009	0.966	1.010	1.010
0.90	0.859	0.983	0.928	1.076	1.026	0.979	0.985
0.99	0.738	0.983	0.928	1.076	1.026	0.979	0.986

Table A.43: SOM - data set A results - $\alpha = 0.99$

A.8 SOM - Data Set D Summary Results

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.278	0.164	0.166	0.169	0.169	0.169	0.169
0.10	0.401	0.251	0.170	0.173	0.173	0.173	0.173
0.30	0.269	0.301	0.218	0.205	0.174	0.171	0.171
0.50	0.233	0.239	0.229	0.186	0.187	0.170	0.170
0.70	0.215	0.219	0.211	0.198	0.186	0.170	0.170
0.90	0.210	0.201	0.143	0.127	0.127	0.127	0.122
0.99	0.198	0.201	0.146	0.127	0.127	0.127	0.126

Table A.44: SOM - data set D results - $\alpha = 0.01$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.311	0.122	0.110	0.110	0.110	0.110	0.110
0.10	0.348	0.244	0.166	0.169	0.169	0.169	0.169
0.30	0.253	0.306	0.218	0.201	0.174	0.171	0.171
0.50	0.228	0.241	0.231	0.186	0.186	0.170	0.170
0.70	0.218	0.221	0.211	0.198	0.186	0.169	0.170
0.90	0.207	0.199	0.143	0.127	0.127	0.127	0.122
0.99	0.195	0.199	0.146	0.127	0.127	0.127	0.126

Table A.45: SOM - data set D results - $\alpha = 0.1$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.303	0.127	0.111	0.108	0.110	0.110	0.110
0.10	0.305	0.190	0.114	0.111	0.111	0.111	0.111
0.30	0.242	0.307	0.217	0.198	0.165	0.165	0.165
0.50	0.235	0.239	0.232	0.186	0.186	0.169	0.170
0.70	0.220	0.210	0.213	0.194	0.178	0.163	0.162
0.90	0.213	0.194	0.141	0.127	0.127	0.127	0.122
0.99	0.190	0.194	0.145	0.127	0.127	0.127	0.126

Table A.46: SOM - data set D results - $\alpha = 0.3$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.189	0.158	0.096	0.110	0.108	0.110	0.110
0.10	0.275	0.199	0.109	0.110	0.108	0.110	0.110
0.30	0.229	0.299	0.205	0.170	0.153	0.151	0.153
0.50	0.231	0.229	0.226	0.187	0.178	0.160	0.159
0.70	0.220	0.207	0.213	0.196	0.177	0.161	0.161
0.90	0.213	0.191	0.141	0.128	0.127	0.126	0.122
0.99	0.190	0.191	0.145	0.128	0.127	0.126	0.126

Table A.47: SOM - data set D results - $\alpha = 0.5$

β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.147	0.131	0.094	0.094	0.110	0.110	0.110
0.10	0.256	0.188	0.102	0.097	0.110	0.110	0.110
0.30	0.219	0.269	0.162	0.136	0.124	0.119	0.119
0.50	0.231	0.220	0.219	0.180	0.167	0.152	0.152
0.70	0.220	0.206	0.211	0.191	0.187	0.160	0.159
0.90	0.213	0.186	0.139	0.128	0.128	0.126	0.122
0.99	0.190	0.186	0.143	0.128	0.128	0.126	0.126

Table A.48: SOM - data set D results - $\alpha = 0.7$

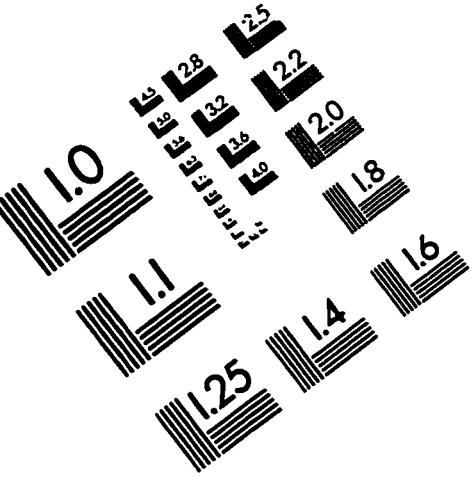
β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.096	0.121	0.098	0.094	0.094	0.109	0.107
0.10	0.206	0.146	0.090	0.095	0.096	0.109	0.109
0.30	0.211	0.223	0.135	0.120	0.116	0.114	0.114
0.50	0.222	0.209	0.175	0.158	0.149	0.137	0.138
0.70	0.214	0.196	0.183	0.164	0.158	0.137	0.138
0.90	0.206	0.181	0.131	0.125	0.125	0.125	0.118
0.99	0.184	0.181	0.136	0.125	0.125	0.125	0.124

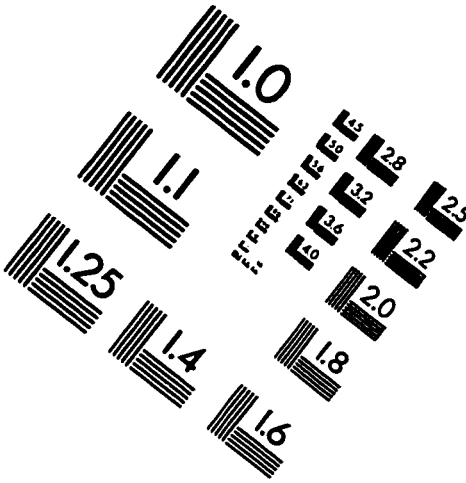
Table A.49: SOM - data set D results - $\alpha = 0.9$

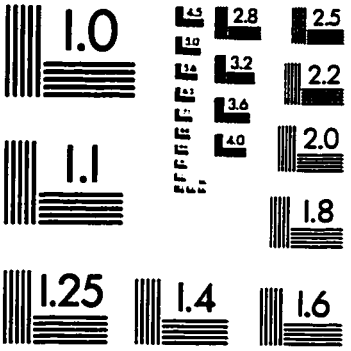
β	γ						
	0.01	0.1	0.3	0.5	0.7	0.9	0.99
0.01	0.091	0.106	0.098	0.094	0.094	0.109	0.107
0.10	0.192	0.146	0.090	0.095	0.096	0.109	0.109
0.30	0.206	0.223	0.135	0.120	0.116	0.114	0.114
0.50	0.218	0.209	0.175	0.158	0.149	0.137	0.137
0.70	0.209	0.196	0.183	0.164	0.158	0.137	0.137
0.90	0.202	0.181	0.130	0.125	0.125	0.125	0.118
0.99	0.184	0.181	0.135	0.125	0.125	0.125	0.124

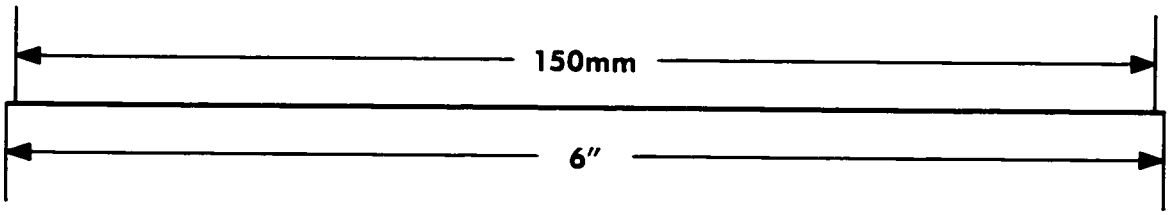
Table A.50: SOM - data set D results - $\alpha = 0.99$

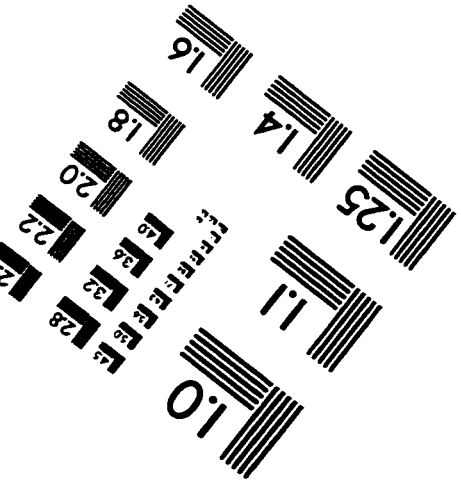
IMAGE EVALUATION
TEST TARGET (QA-3)

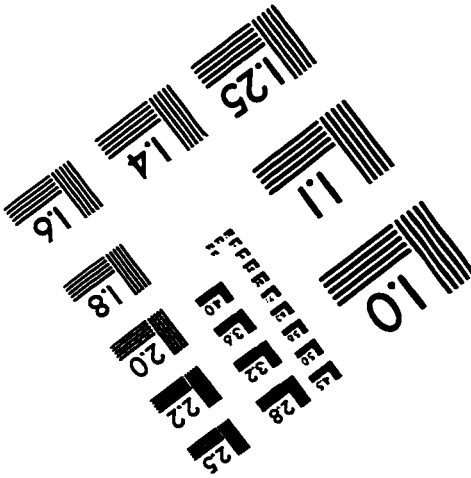












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989