SELF-ORGANIZING DISCOVERY, RECOGNITION, AND PREDICTION OF HEMODYNAMIC PATTERNS IN THE INTENSIVE CARE UNIT

A Thesis

by

RONALD GLEN SPENCER

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

June 1994

Major Subject: Bioengineering

SELF-ORGANIZING DISCOVERY, RECOGNITION, AND PREDICTION OF HEMODYNAMIC PATTERNS IN THE INTENSIVE CARE UNIT

A Thesis

by

RONALD GLEN SPENCER

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Charles S. Lessard **(Chair of Committee)**	**Tep Sastri** **(Member)**
Fidel Davila **(Member)**	**Way Kuo** **(Head of Department)**
Brad Etter **(Member)**	

August 1994

Major Subject: Bioengineering

ABSTRACT

Self-Organizing Discovery, Recognition, and Prediction of Hemodynamic Patterns in the Intensive Care Unit.

(August 1994)

Ronald Glen Spencer, B.S.E.E., GMI Engineering & Management Institute

Chair of Advisory Committee: Dr. Charles S. Lessard

In order to properly care for critically ill patients in the intensive care unit (ICU), clinicians must be aware of hemodynamic patterns. In a typical ICU a variety of physiologic measurements are made continuously and intermittently in an attempt to provide clinicians with the most accurate and precise data needed for recognizing such patterns. However, the data are disjointed, yielding little information beyond that provided by instantaneous high/low limit checking. While instantaneous limit checking is useful for determining immediate dangers, it does not provide much information about temporal patterns. As a result, the clinician is left to manually sift through an excess of data in the interest of generating information. In this study, an arrangement of self-organizing artificial neural networks (ANNs) is proposed to automate the discovery, recognition, and prediction of such hemodynamic patterns in real-time and ultimately lessen the burden on clinicians. ANNs are well suited for pattern recognition and prediction in a data-rich environments because they are very trainable and have a tendency to *discover* their own internal representations of knowledge, thus reducing the need for *a priori* knowledge in symbolic form. Results from actual clinical data are presented.

ACKNOWLEDGMENTS

I would like to thank Dr. Charles Lessard for setting up this project and making the initial connections with Dr. Davila at Scott & White Memorial Hospital in Temple, Texas. His willingness to be a part of global solutions to problems in the health care system in addition to his collaborative skills as a researcher are very much respected.

I would also like to thank Fidel Davila, M.D., Chief of Pulmonary Critical Care at Scott & White Memorial Hospital, for his guidance throughout this study. Dr. Davila served as the 'backbone' of the project. He is very much respected for his skills as a medical doctor, scientist, and researcher. The most admired characteristic is his willingness to dream a little for the sake of advancing traditional procedures and techniques which, for some clinicians, are 'set in stone'. Any practicing clinician that can care for critically ill patients and still find time to read about fractals and chaos theory is my kind of person. Dr. Davila realizes the need for long term thinking *nested within* short term productivity.

I would also like to thank Dr. Brad Etter and Dr. Tep Sastri for their suggestions and corrections of the manuscript. Dr. Sastri's careful reading for *content* was very much appreciated. Dr. Sastri is interested in applying artificial neural networks to problems in industrial engineering, an area that I, myself, spent a lot of time in. In fact, problems in industrial engineering are what *led* me to neural networks in the first place. So we have much in common.

And finally, I would like to thank Jennifer for her loving support through rough times, late nights, and missed events. She was very helpful and understanding throughout.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
A/D	Analog to Digital
AI	Artificial Intelligence
ANN	Artificial Neural Network
ART	Adaptive Resonance Theory
ART2	Continuous-Valued Adaptive Resonance Theory
AST	Adaptive Spectral Timing
BPM	Beats Per Minute
C/D	Concentrated to Distributed
CO	Cardiac Output
CVP	Central Venous Pressure
CS	Conditioned Stimulus
DAP	Diastolic Arterial Pressure
D/C	Distributed to Concentrated
FIFO	First-In, First-Out
HYPRT	Hypertension
HYPOT	Hypotension
HR	Heart Rate
ICU	Intensive Care Unit
IPP	Increased Pericardial Pressure
IVD	Intravascular Volume Depletion
IVO	Intravascular Volume Overload
ISI	Interstimulus Interval
LI	Leaky Integrator

Abbreviation	Description
LTM	Long Term Memory
MAP	Mean Arterial Pressure
PAD	Pulmonary Arterial Diastolic Pressure
PP	Pulse Pressure
SAP	Systolic Arterial Pressure
STM	Short Term Memory
SV	Stroke Volume
SVR	Systemic Vascular Resistance
TDNN	Time Delay Neural Network
US	Unconditioned Stimulus
UR	Unconditioned Response
VC	Vasoconstriction
VD	Vasodilation

INTRODUCTION

Clinicians in the ICU must be aware of hemodynamic patterns in order to properly care for patients. Recognition of these patterns requires the recognition of temporal sequences of physiologic events such as vasoconstriction, vasodilation, hypotension, hypertension, intravascular volume overload/depletion, and increased pericardial pressure. These events are recognized by observing instantaneous values and rates of change of raw physiologic variables such as heart rate, cardiac output, and blood pressure. Many of these variables are measured continuously by automatic devices but the values are only monitored to ensure that they stay within normal limits. While instantaneous limit checking is useful for determining immediate dangers, it does not provide much information about temporal patterns.

Although many of the underlying processes involved in recognizing hemodynamic patterns are perfunctory, they have not yet been automated. Currently, the information must be manually extracted by clinicians. This burdensome and time-consuming conversion of data to information makes the clinician's job more difficult, distracts from critical tasks, and lessens the chance of it being done correctly. Consequently, there is a great need for automatic recognition of hemodynamic patterns.

Background of Medical Expert Systems

In recent decades, the greatest advancements in bedside critical care monitoring have been in the accuracy, precision, and frequency of physiologic data collection. The resulting data are more reliable and abundant than ever before. However, the data are still disjointed, yielding little information beyond that provided by instantaneous high/low limit checking.

This document follows the style of Medical & Biological Engineering & Computing.

Several experimental expert systems have been created for the purpose of integrating the data and automating the recognition of certain diagnoses, but they have not made their way into

the clinic. A large number of them exist within the symbolic framework of traditional artificial intelligence paradigms and emulate the reasoning processes of clinical experts. These systems are largely rule-based and require much information or knowledge *a priori*. A very attractive attribute of these traditional artificial intelligence (AI) systems is their ability to explain deductive processes in a language that is understandable. This ability is due to the fact that knowledge exists in discrete, modular fragments that can be chained through to arrive at conclusions.

However, several problems with traditional expert systems exist due to the profuse dependence on rules. First of all, the task of programming a rule-based system is not very feasible, considering the patchy nature of rules themselves. By the law of diminishing returns, the more patchwork one must do in order to create a working system, the less productive the effort will be. No matter how extensive the rule base, there will always be exceptions. Second, the extraction of rules from experts is not always consistent. Since experts develop their own internal representations of reality, a rule extracted from one expert may be different than the same rule extracted from another. Rules that are valid within one framework may be inappropriate in another. Third, experts are sometimes not even aware of the underlying rules by which they perform. As stated by Bart Kosko, experts often kick away the 'ladder of learning' after having climbed it. In a sense, rules are the 'training wheels' of learning which are eventually discarded. And finally, losses in translation may occur from the expert to the programmer and from the programmer to the computer even if the experts are able to articulate consistent rules in the first place. As a result, the rules may become distorted.

An alternative type of expert system, the connectionist expert system, is receiving much interest (GALLANT 1988). Connectionist expert systems are composed of large numbers of distributed connections between many low-level processing components arranged in hierarchies. By nature, these systems are highly adaptive, fault tolerant, and massively parallel. They are usually very trainable and have a tendency to discover their own internal representations of

knowledge, reducing the need for *a priori* knowledge in symbolic form. These features make connectionist expert systems very attractive in data-rich environments.

Recently, there have been some signs of integration between the two paradigms (EBERHART & DOBBINS, 1991; FU 1991; LIOW & VIDAL, 1991; POLI, *et al*., 1991; RIALLE *et al*., 1991; PAPADOURAKIS *et al*., 1992). As stated by FU (1991), "Though some think of knowledge-based systems and neural networks as separate AI tools, viewing them simply as different levels of description about the same mechanism would be more insightful. Combination of these two technologies has recently emerged as an important direction toward building new-generation intelligent systems."

RIALLE *et al*., 1991 have investigated medical diagnostic systems that incorporate both high-level symbolism and low-level connectionism. They report four levels of interrelation: loose coupling, tight coupling, full integration, and transformation. An example of loose coupling comes from KASABOV (1990) where production rules which represent symbolic knowledge interact with ANNs that perform classification and pattern recognition tasks. An example of tight coupling comes from GALLANT (1986) where a system called MACIE is applied to the diagnosis and treatment of acute sarcophagal disease. This system relies on a multi-layer network for knowledge representation, but employs forward and backward chaining for process explanation like an expert system. TIAN HE & TAI JUWEI (1989) proposed a multi-layer ANN to be used for automatic knowledge acquisition and deep knowledge modeling in a knowledge-based system used in traditional Chinese medicine for children's cough disease. RIALLE *et al*., 1991 worked on a loosely coupled system to interpret quantitative electromyographic data.

FU (1991) described a hybrid medical expert system for the diagnosis of jaundice disorders which consisted of conventional AI rule-based programming that cooperated with an ANN that revised the rule base. Fu was able to map the rule-based system into a neural architecture with his own mapping techniques (FU & FU, 1990) where variables and hypotheses were assigned to nodes and rules were mapped into the interconnections between them. Back-

propagation, a supervised connectionistic training algorithm, (RUMELHART, *et al*., 1986) was used to train 'non-conjunction layers' and an AI-based 'hill-climbing' algorithm was used with 'conjunction layers'. The system performed well with a 95.5 per cent accuracy.

In 1990, COHN, ROSENBAUM, FACTOR, & MILLER reported on a very practical expert system called DYNASCENE. Although the individual components of the system were not artificial neurons, per se, the system exhibited a connectionistic macrostructure. It was shown that DYNASCENE could recognize temporal sequences of vasoconstriction, vasodilation, hypotension, intravascular volume overload, and intravascular volume depletion and associate them with the corresponding disorders. The disorders that were recognized by DYNASCENE, along with the corresponding temporal sequences used to diagnose them, are listed:

1. Congestive Heart Failure With Fluid Overload:
   intravascular volume overload ® vasoconstriction ® hypotension

2. Congestive Heart Failure Due to Pump Failure:
   myocard. ischemia ® vasoconstriction ® hypotension ® intravascular vol overload

3. Hypovolemia:
   intravascular volume depletion ® vasoconstriction ® hypotension

4. Sepsis:
   vasodilation ® hypotension ® vasoconstriction

5. Cardiac Tamponade:
   increased pericardial pressure ® vasoconstriction ® hypotension

DYNASCENE was *explicitly programmed* to recognize these five sequences before being applied. As a result, it was limited to recognizing these five sequences only.

Objective

The overall objective of this study is to create a self-organizing hemodynamic pattern recognition system of several artificial neural networks capable of discovering, recognizing, and

predicting hemodynamic patterns in the ICU without being *explicitly programmed* to do so. After these hemodynamic patterns are discovered they can be symbolically labeled (calibrated) by the clinician according to their meaning (diagnosis) and assigned alarms to alert clinicians when they occur.

The objective is to discover unknown hemodynamic patterns in addition to well known patterns. The system will be explicitly programmed to recognize the instantaneous physiologic events that make up these hemodynamic patterns, but not the hemodynamic patterns themselves. Success will be evaluated according to its ability to recognize the same five hemodynamic sequences recognized by DYNASCENE: 1) congestive heart failure with fluid overload, 2) congestive heart failure due to pump failure, 3) hypovolemia, 4) sepsis, and 5) cardiac tamponade, given the ability to recognize the underlying instantaneous physiologic events.

At the very least, the system must be able to perform the following: 1) store current and previous physiologic events in short term memory (STM), 2) recognize temporal patterns of the physiologic events, and 3) predict future temporal patterns. Figure 1 shows the flow of data through each of these stages in addition to two other stages assumed to be operational: measurement and instantaneous physiologic pattern (event) recognition.

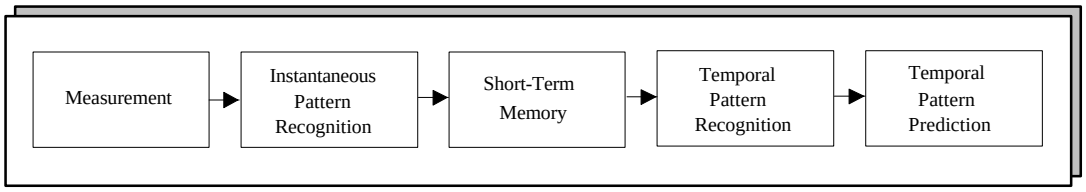


Fig. 1. Flow diagram for temporal pattern recognition and prediction.

Rationale

Automation of perfunctory tasks is one way of streamlining health care. Not only is automation an inevitable part of the future, but it is also cost effective when done correctly. As a result, any task that can be automated, should be automated, if it can reduce the workload of the

employees or reduce the time of stay of patients at a reasonable cost without jeopardizing the quality of the health care. In many instances, automation of perfunctory tasks should improve the quality of care. As one example, when a patient is admitted into the ICU, the clinicians are not usually aware of the chances of survival. As time passes, the clinicians become increasingly aware of the odds. If an automatic hemodynamic pattern recognition system could recognize patterns that indicate low odds of survival early in the treatment process, a considerable amount of time and money could be saved in addition to letting the patient live his last few days at home. Although such a system is not intended to replace intuition, it could enhance the ability of clinicians to make such critical decisions.

Self-organizing neural networks are well suited to the task of recognizing such temporal patterns. A large advantage of these networks is their lack of need for *a priori* information. In a clinical environment where raw data are much more abundant than expert diagnoses, this independence is an immediate advantage. In addition, unsupervised networks organize data according to the intrinsic salient features of the distribution that the data comes from. Due to the fact that these networks learn without supervision, they develop their own internal representations of the data and *discover* patterns *a posteriori*. As a result, they are able to find many patterns in the data, whether they are well known or not. It is expected that a system composed of such networks could advance medical knowledge and understanding by discovering new patterns.

Preview

In the pages that follow, the stages of discovering, recognizing, and predicting hemodynamic patterns are described. To begin with, a review of context in dynamic systems is given. The following chapter is on temporal pattern recognition which reviews several ANN models of short term memory and pattern recognition. Next, prediction is dealt with from a classical conditioning point of view as Hebbian learning is introduced along with the adaptive spectral

timing model. Following these chapters are the methods that were used to construct the hemodynamic pattern recognition and prediction system. Subsequent chapters present results of simulations, discuss methods and limitations, and offer recommendations and conclusions.

CONTEXT IN DYNAMIC SYSTEMS

When describing or tracking a dynamic process, a simple snapshot of the system is not always adequate. By itself, an instantaneous snapshot does not contain enough information to adequately define the state of the system. When listening to a song, for example, characteristics emerge from the collection of notes that are not evident in the individual notes. This result is due to the fact that dynamic systems often evolve along complex trajectories which, at any point in time, can be expressed in terms of location and direction. Since instantaneous measurements convey location information only, more than one state of the system can have the same set of instantaneous measurements. As a result, there is a need for obtaining directional information in order to evaluate the true state of the system. This contextual information must be extracted from time series of the instantaneous measurements.

Extracting contextual information from a deterministic time series, for example, requires that previous patterns be remembered. The purpose of this memory is to provide information about the direction of system which, in turn, can be used to expect future patterns. This process can be illustrated by the low-dimensional chaotic sequence, *AXBCXD*, generated by traversing the figure-eight as shown in figure 2. Knowing that the current location is X, neither the current

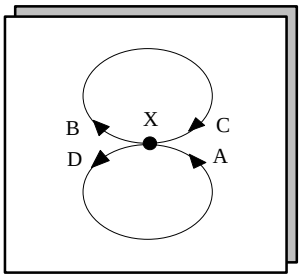


Fig. 2. Illustration of the need for contextual information. Moving in the direction shown on the figure-eight generates the sequence of instantaneous patterns, *AXBCXD*. Since *X* precedes *B* in one case and *D* in another, knowing the current location only does not allow for confident prediction of the next state. However, knowing the previous location too allows for the successful prediction of the next location.

state of the process nor the next location can be determined. In one instance, the following location is B and in the other it is D. Instantaneous information alone is simply not adequate. However, if the previous location is remembered too, then the next location can be determined. Knowing that the previous and current locations are A and X, respectively, location B can be predicted with success. The memory of the previous location provides context.

Biological systems exhibit much higher dimensional chaotic behavior than this simple deterministic system. Consequently, the events of a particular sequence may not always occur in a contiguous fashion. However, there is still a need for recognizing *order of occurrence* regardless of how closely the events occur. Recognizing hemodynamic patterns, for example, requires that different *permutations* of the same *combination* of events be discriminated. The intelligent cardiovascular monitoring system, DYNASCENE, was programmed to discriminate between congestive heart failure due to *fluid overload* versus congestive heart failure due to *pump failure* by recognizing different permutations of the same combination of three physiological events: intravascular volume overload, vasoconstriction, and hypotension. In both cases, all three events occurred, but in a unique order. In order to recognize these unique orders, some type of temporal pattern recognition system is needed.

TEMPORAL PATTERN RECOGNITION

The ultimate objective of temporal pattern recognition is to maximally activate one or more units in response to a particular temporal pattern which exists over space and time. In a sense, it is the *reduction* of a set of temporal events into a single event. Translated in terms of this study, temporal pattern recognition is the activation of a single unit in response to a hemodynamic pattern of a set of physiologic events.

Temporal pattern recognition requires: 1) a short term memory (STM) to represent the temporal information and 2) a means for categorizing or recognizing the state of the STM. As shown in figure 3, setting up the first task involves finding an optimum method for representing time *spatially*. The second task involves categorizing, or concentrating, the information in space.

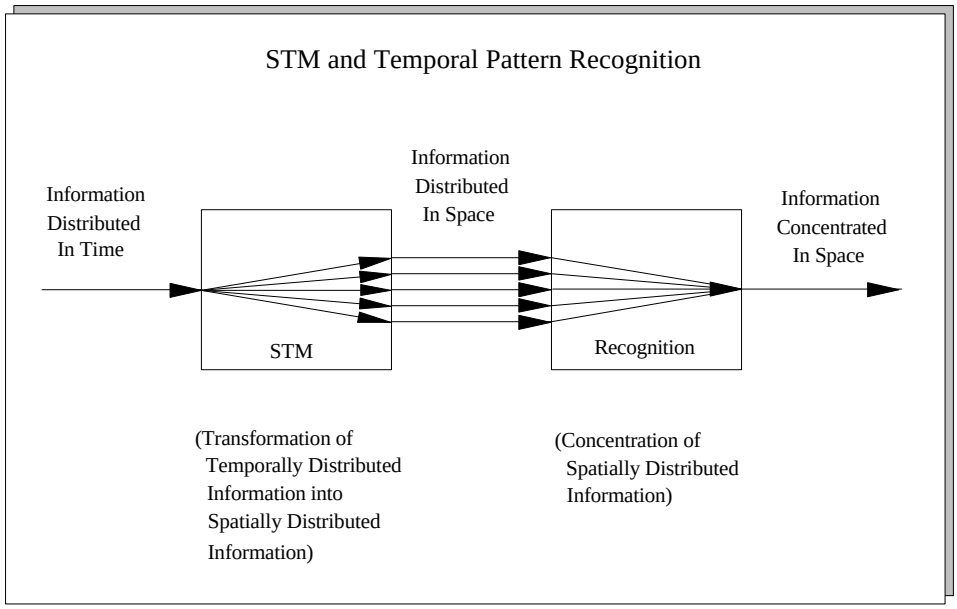


Fig. 3. The roles of STM and categorization in temporal pattern recognition. Information that is distributed in time is redistributed in space by STM. The resulting spatial information is recognized which constitutes a concentration, or reduction, of the information in space.

Many artificial neural networks have been proposed to recognize temporal sequences (HIRSCH 1989; WILLIAMS & ZIPSER, 1989; WERBOS 1990; COHEN 1992; SHIMOHARA *et al*., 1993); however, many of them employ supervised learning techniques which require a 'teaching output' for each input sequence. Although supervised learning is useful in a number of applications where the sequences to be learned are known *a priori*, it is not useful for *discovering* sequences where the sequences to be learned are not known before hand. However, supervised learning rules can be used for unsupervised learning by using the *input* as the '*teaching output*' (REISS & TAYLOR, 1991). This type of learning discovers transformations of input sequences that predict future patterns. Although such systems can be used to discover and recall sequences in an unsupervised manner, they do not *evaluate* or *categorize* the sequences according to their *inherent features*. As a result, the learned sequences can neither be reduced to nor expressed by individual units that can be semantically labeled. The position taken here is that in order to successfully discover and recognize sequences, they must be associated with single categories according to the degree of likeness between themselves and previously learned sequences. The advantage of this scheme lies in being able to monitor such categories for activity and sound alarms when they occur. This process requires: 1) representing the temporal sequences in STM in a continuous framework of measurement and 2) categorizing the sequences according to the information in STM.

Short Term Memories (STM)

In the neural network literature, STM is distinguished from long term memory (LTM) by the mechanisms used to store information. LTM is usually associated with slowly varying synaptic potentials, whereas STM is usually associated with quickly varying membrane potentials (CARPENTER, 1989). Several types of STM have been proposed and applied to information processing problems. Two of the most widely studied STMs are time-delay neural networks (TDNN) (McCLELLAND & ELMAN, 1986; TANK & HOPFIELD, 1987; ELMAN &

ZIPSER, 1988; KOHONEN, 1989; WAIBEL *et al*., 1989; LIPPMANN, 1989; de VRIES & PRINCIPE, 1992) and the self-exciting, leaky integrator (LI) (de VRIES & PRINCIPE, 1992; GJERDINGEN, 1992).

The simplest time-delay neural networks are essentially shift registers or first-in, first-out (FIFO) buffers. A simple TDNN stores a predetermined number of the most recent instantaneous values (snapshots of the dynamic system) as shown in figure 4. These values are

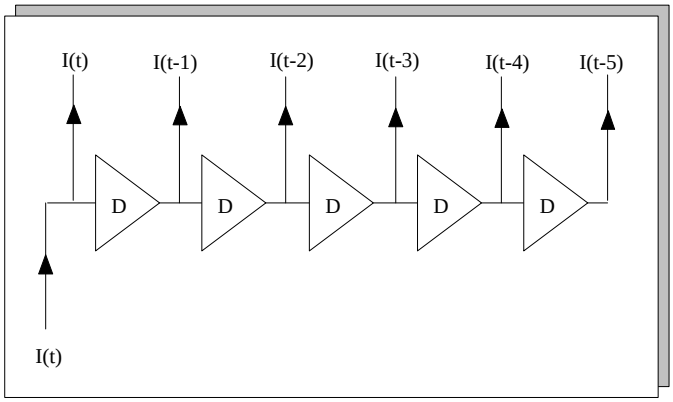


Fig. 4. A simple tapped delay line for remembering previous values of a single variable *I*.

shifted once per iteration, deleting the oldest value and storing the most recent value. Other neural networks can tap into this delay line and extract contextual information and for this reason, they are sometimes called 'tapped delay lines'. The advantage is their ability to retain primacy information in addition to recency information.

On the other hand, self-exciting, leaky integrators do not shift information from one node to another. Instead, a single LI receives input from its own external source as shown in figure 5 and continues to excite itself with a leaky reverberating connection. The LI remembers the *recency* of its external input. The advantage of the LI lies in the fact that information is not moved around, and therefore a single unit can be labeled according to its meaning in terms of the

external stimulus. And for the most part, the membrane potential generally follows a continuous time course of change unlike that of a tapped delay line unit.

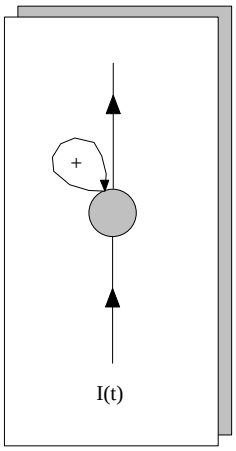


Fig. 5. A self-exciting leaky integrator for remembering the recent history of a single variable *I*.

When monitoring a multidimensional process where the number of variables to be monitored is greater than one, many LIs can be allocated; one for each dimension. The result is a STM vector which conveys recency information about the multidimensional process. This approach has been applied to music recognition and categorization (GJERDINGEN 1992).

Categorization of STM Patterns with Adaptive Resonance Theory (ART)

As mentioned earlier, temporal pattern recognition is the process of activating a single unit in response to a temporal sequence. This process involves reducing a spatiotemporal pattern to the activation of one or more units that are associated with different temporal patterns. For example, if a sequence is represented in STM, there must be some observing unit that is tuned to that particular sequence more than any other observing unit. Upon seeing that sequence, the unit must be maximally activated. This activation is referred to as recognition. In a sense, recognition is an analog to digital (A/D) conversion because it is transforming an event that occurs to a degree

into an event that occurs in binary. But it's more than this; it's also a *spatial bottleneck*. A more appropriate description of recognition might be 'distributed to concentrated' (D/C) conversion.

In any event, the temporal patterns in STM must be categorized according to the degree of likeness between themselves and other temporal patterns. Furthermore, this categorization must be unsupervised in order to discover frequently occurring patterns. One type of unsupervised categorization scheme known as the adaptive resonance theory (ART) (CARPENTER & GROSSBERG, 1987b) can do just this (GJERDINGEN 1992). Adaptive resonance theory networks categorize vectors according to similarity with other vectors. For the most part they are neural clustering techniques that partition the vector space by iteratively refining a set of exemplar representations, or categories.

Adaptive resonance theory networks are capable of learning new patterns without compromising the integrity of the old ones. This ability is a large advantage that ART networks have over other self-organizing networks. Learning in ART networks is not distributed over neighborhoods like other self-organizing models (KOHONEN 1982). Instead, learning occurs one category at a time. As a result, only appropriate categories are modified to accommodate current inputs while other categories remain intact. This ability can be very useful in categorizing hemodynamic patterns where, for instance, there is a need for distinguishing between two different permutations of the same combination of events.

ART networks are generally not thought of as temporal pattern recognition networks, per se, because the words 'temporal pattern recognition' usually imply the process of storing temporal information in STM as well as actually recognizing it. Adaptive resonance theory networks, like many other static networks, recognize patterns in a flashcard fashion, i.e. they operate on one input vector at a time, each operation being independent of the previous ones. The reason that an ART network can be used to recognize temporal patterns is because the temporal information *already exists* in the STM vector that it categorizes. Consequently, the ART network is not

required to maintain an internal representation of time; it is only required to *categorize* representations that have already been formed.

TEMPORAL PATTERN PREDICTION AND CLASSICAL CONDITIONING

The second objective of the system is temporal pattern prediction. In this chapter, prediction models are reviewed from the perspective of classical conditioning. The most characteristic feature of this type of learning is the association of two events separated in time by an interstimulus interval (ISI). Because this type of learning *associates* two events separated in *time*, it might seem more appropriate to call it 'temporal associative learning'. However, the conditioned learning paradigms reviewed here fit well within the classical conditioning framework of experiments performed in the early 20th century and furthermore, the name 'temporal associative learning' has been reserved for the hetero-associative task of producing one sequence in response to another in (HERTZ, KROGH, & PALMER, 1991). Consequently, the name 'classical conditioning' is retained.

Hebbian Learning and Classical Conditioning

In 1949 Donald Hebb proposed a simple, self-organizing learning rule that is still ubiquitous today:

> "When an axon of cell *A* is near enough to excite a cell *B* and repeatedly or persistently takes place in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased." (Hebb, 1949)

With this statement, the self-similar characteristics of classical conditioning were cast into a neural framework which laid the foundation for neural network research for years to come. One of the most appealing propositions of the statement was the idea that synapses behave much like muscle, i.e. they grow stronger with use. Or more correctly, the synapses grow stronger if they are active simultaneously with downstream neuron *B*. Due to the fact that neuron *B* has a threshold for firing, the synapse may be inadequate for exciting neuron *B*. In this case, although

the synapse is active, the Hebbian condition is not satisfied and the synapse does not grow stronger. But if a large number of synapses which innervate neuron *B* are stimulated simultaneously, neuron *B* may start firing. In this case, the Hebbian condition is satisfied and these synapses are rewarded with trophic factor. Consequently, these synapses grow while less active synapses atrophy.

The ideas of classical conditioning fit well within the Hebbian framework. To demonstrate, let neuron *i* recognize event I_{CS}, the conditioned stimulus (CS), and let neuron *j* recognize event I_{US}, the unconditioned stimulus (US). Furthermore, let neuron *j* produce the unconditioned response (UR) when adequately stimulated. This arrangement is illustrated in figure 6. When I_{CS} occurs, neuron *i* fires and when I_{US} occurs, neuron *j* fires. In the beginning of the learning, z_{ij} is too weak for neuron *i* to activate neuron *j* by itself. This result is due to the fact that $f(x_i)z_{ij}$ is too small to push neuron *j* over its threshold. But if the experiment is repeated several times, z_{ij} grows to a point where $f(x_i)z_{ij}$ is great enough to activate neuron *j without* I_{US}. At this point, a temporal association between I_{CS} and I_{US} has been encoded and I_{CS} alone, is adequate to produce UR.

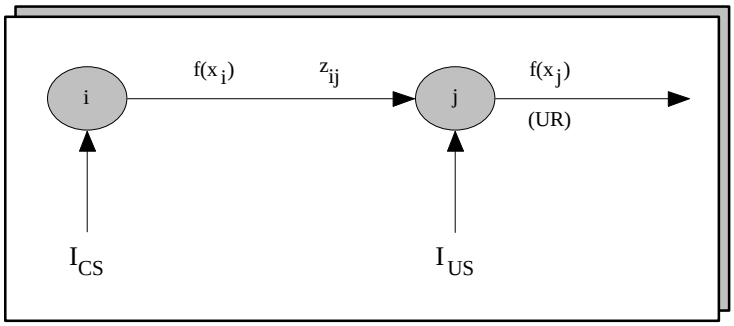


Fig. 6. Two neurons connected by a Hebbian synapse in the spirit of classical conditioning.

In terms of Pavlov's salivating dog, I_{CS} represents the whistle, I_{US} represents the food, and the response of neuron *j*, $f(x_j)$, represents salivation (PAVLOV, 1929). However, in Pavlov's

experiment the whistle and food were not presented simultaneously. Instead, the food followed the whistle by some non-zero ISI. Consequently, the Hebbian model in figure 6 misses the temporal relationship between I_{CS} and I_{US} because the signal from neuron *i* arrives *instantaneously* at neuron *j* at the onset of I_{CS}. Put another way, the activation rate of synapse z_{ij} is infinite. A synapse with a slower activation rate is needed. If chosen to match the ISI between I_{CS} and I_{US}, a new activation rate, α, slows the propagation of the signal from neuron *i* to neuron *j* so that synapse z_{ij} is activated at the same instant that I_{US} occurs and the Hebbian condition is satisfied.

It should be noted that this model does not subsume Pavlov's experiment entirely because even the slowest activation of synapses is too quick to correlate with long ISIs (GROSSBERG & SCHMAJUK, 1989). However, as with many other self-similar structures, some of the same characteristics that emerge at one level of organization, emerge at another. And since the objective in this study is to apply the information processing capabilities of such networks, this model is interesting.

Adaptive Spectral Timing

An important extension of the classical conditioning model is the adaptive spectral timing model (GROSSBERG & SCHMAJUK, 1989). In the classical conditioning example, the activation rate, α, was chosen *a priori* to agree with the ISI. However, when the ISI is unknown, α cannot be chosen *a priori*. Instead of having one synapse and one activation rate, a *set* of pathways with a *spectrum* of activation rates can be envisioned between neuron *i* and neuron *j* as shown in figure 7. This forms the basis of the adaptive spectral timing (AST) model. Denoting the pathways between the *i*th and *j*th neurons by z_{ijk} where $1 \leq k \leq K$, each z_{ijk}

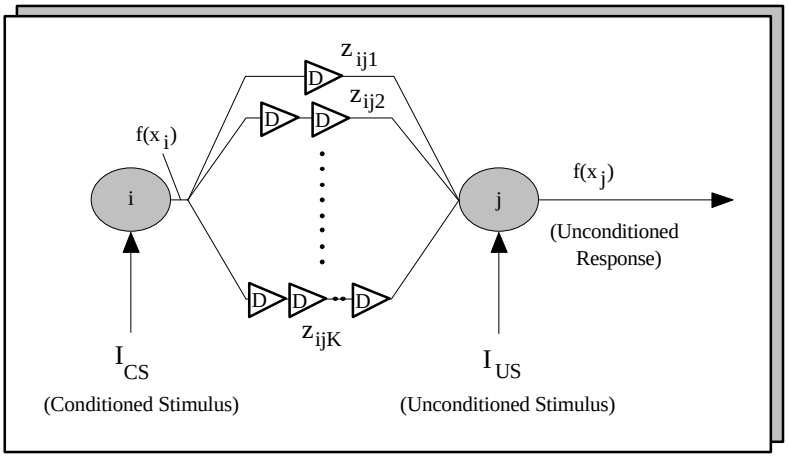


Fig. 7. Adaptive spectral timing pathways from one neuron to another.

has an activation rate proportional to 1/*k*. Consequently, small indices correlate with short ISIs and large indices correlate with long ISIs.

The continuous version of the adaptive spectral timing equations are shown in equations (1-6) for a single adaptive pathway, *k*, from the neuron where the conditioned stimulus, I_{CS}, is sensed to the neuron where the unconditioned stimulus, I_{US}, is sensed. In addition to the axon coming from the neuron where I_{US} is sensed, the adaptive pathway is actually made up of more than one interneuron. The membrane potential, x_k, of first interneuron in the pathway is governed by the shunting activation equation:

$$\frac{dx_k}{dt} = a_k[-Ax_k + (1 - Bx_k)I_{CS}(t)] \qquad (1)$$

At the onset of I_{CS}, the membrane potential charges from its passive saturation potential of zero to its active excitatory saturation potential at a rate, α_k, which is inversely proportional to *k*. As x_k charges up, it begins to inactivate process y_k, activating the neurotransmitter at the synaptic cleft as expressed by the transmitter gate equation:

$$\frac{dy_k}{dt} = C(1-y_k) - Df(x_k)y_k \tag{2}$$

The degree to which the membrane potential activates neurotransmitter y_k is governed by the monotonically increasing sigmoidal function given by:

$$f(x_k) = \frac{x_k{}^n}{b^n + x_k{}^n} \tag{3}$$

As x_k is charging up, y_k is dumping neurotransmitter from a limited source into the synaptic cleft. There exists a point in time, depending on the activation rate α_k where the postsynaptic knob is activated maximally. Beyond this point, less neurotransmitter is available for release and the prereleased neurotransmitter is being diffused away by processes responsible for reclaiming it. The LTM trace of the synapse z_k learns at a rate that is determined partly by the product of $f(x_k)$ and y_k as shown by:

$$\frac{dz_k}{dt} = Ef(x_k)y_k[-z_k + I_{US}(t)] \tag{4}$$

If I_{US} occurs when the product of $f(x_k)y_k$ is maximal, then the efficacy of synapse z_k is maximally increased. As a result, z_k becomes stronger when the ISI between I_{CS} and I_{US} and the activation period agree. Finally, the response of the neuron that senses I_{US}, is given by:

$$R = [\sum_k f(x_k)y_k z_k - q_k]^+ \tag{5}$$

where $[w]_+ = \max(w,0) = w$ if $w>0$, 0 otherwise (6)

METHODS

From the models presented thus far, a self-organizing hemodynamic pattern recognition and prediction system was created and simulated. It consisted of three layers: 1) short term memory layer, 2) temporal pattern recognition (categorization) layer, and 3) temporal pattern prediction layer. Each layer is described below.

Short Term Memory

After recognizing the seven physiologic conditions, they were stored in a STM, consisting of seven neurons as shown in figure 8. Each neuron corresponded to one of the seven physiologic conditions, denoted by I_p as follows: 1) intravascular volume overload (I_0), 2) intravascular volume depletion (I_1), 3) increased pericardial pressure (I_2), 4) vasoconstriction (I_3), 5) vasodilation (I_4), 6) hypotension (I_5), and 7) hypertension (I_6).

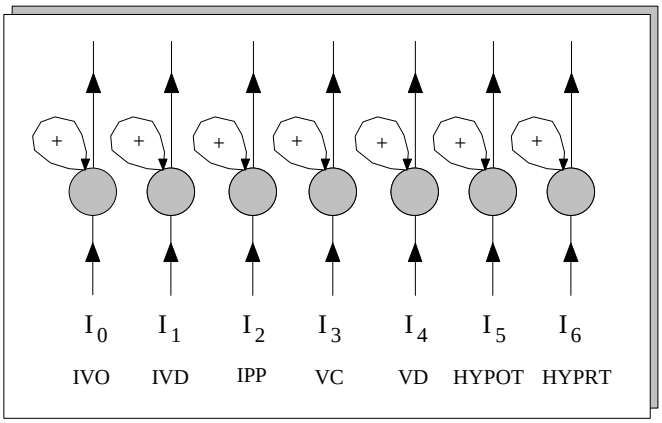


Fig. 8. Short term memory of leaky integrators for remembering recent occurrences of the seven physiologic conditions.

Inspired by the same dynamic STM used in (GJERDINGEN, 1992) the equation used to govern the STM potentials was derived from a typical shunting activation equation:

$$\frac{ds_p}{dt} = -As_p + (B - s_p)\sum I_{exc} - (C + s_p)\sum I_{inh} \tag{7}$$

where s_p was the variable membrane potential of the pth STM node. Shunting activation equations are discussed in appendix C. Rewriting (7) in terms of a difference equation yielded:

$$s_p(t+1) - s_p(t) = -As_p(t) + (B - s_p(t))\sum I_{exc} - (C + s_p(t))\sum I_{inh} \tag{8}$$

$$s_p(t+1) = (1-A)s_p(t) + (B - s_p(t))\sum I_{exc} - (C + s_p(t))\sum I_{inh} \tag{9}$$

Letting $\Sigma I_{exc} = I_p(t)$ and $\Sigma I_{inh} = 0$ in (9) yielded:

$$s_p(t+1) = (1-A)s_p(t) + (B - s_p(t))I_p(t) \tag{10}$$

Due to the fact that difference equations operate in discrete time and no passive decay occurs between iterations, a sustained input of $I_p(t)=1$ to equation (10) produces oscillations that do not occur in continuous time. One way to remedy this situation was to make $I_p(t)$ smaller than unity and perform many 'subiterations' to drive the node to saturation. Another way is to replace the second term of (10) with $[B-(1-A)s_p(t)]I_p(t)$ to model passive decay between iterations. The last alternative was chosen because it did not require additional subiterations. This replacement prevents artificial oscillations from occurring when the membrane potential is driven to saturation:

$$s_p(t+1) = (1-A)s_p(t) + [B - (1-A)s_p(t)]I_p(t) \tag{11}$$

where $0 < A < 1$. When process p occurred, $I_p(t)=1$ and $s_p(t)$ was driven to saturation in one iteration. The membrane potential $s_p(t)$ remained saturated until process p ceased to exist, at

which time it began to decay exponentially. As an example, the most recent process(es) usually had an activation level of unity, the second most recent had an activation level of (1-*A*), the third had an activation of $(1\text{-}A)^2$, and so on.

Using the activations of all STM nodes, a *P*-dimensional, unnormalized STM vector $\mathbf{s_u(}t\mathbf{)}$ was created:

$$\mathbf{s_u}(t) = \left[s_0(t) \;\; s_1(t) \;\; s_2(t) \;\; \ldots\ldots\ldots \;\; s_{(P-1)}(t) \right] \tag{12}$$

The seven-dimensional vector of STM traces reflected the temporal order in which the corresponding physiologic events occurred. A typical STM configuration is shown in figure 9.

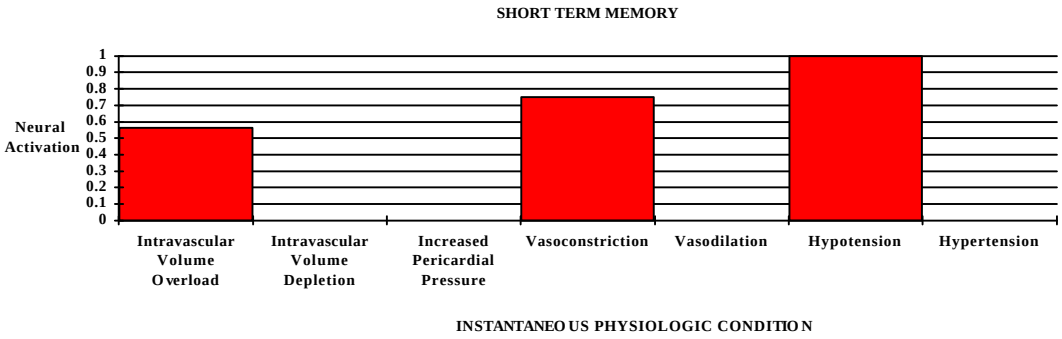


Fig. 9. A typical state of the STM. As shown by the neural activation levels, intravascular volume overload was encountered first, followed by vasoconstriction, and then hypotension.

A (*P*+1)-dimensional, normalized STM vector $\mathbf{s(}t\mathbf{)}$ was created by augmenting $\mathbf{s_u(}t\mathbf{)}$ with a normalization value $s_P(t)$ and dividing by a slow varying parameter $R(t)$:

$$\mathbf{s}(t) = \frac{\left[s_0(t) \;\; s_1(t) \;\; s_2(t) \;\; \ldots\ldots\ldots \;\; s_{(P-1)}(t) \;\; s_P(t) \right]}{R(t)} \tag{13}$$

where $R(t)$ was 1.1 times the maximum $\left| \mathbf{s_u(}t\mathbf{)} \right|$ encountered up to time *t* as shown by:

$$R(t) = 1.1*\max(\left| \mathbf{s_u(}t\mathbf{)} \right|, \; t \in \{1,2,3,.....t\}) \tag{14}$$

and $s_P(t)$ was calculated by:

$$s_P(t) = \sqrt{R(t)^2 - \sum_{p=0}^{P-1} s_p(t)^2} \tag{15}$$

In the limit, as $t \to \infty$, $R(t)$ became a slow parameter or constant. This type of alternative normalization served to retain the absolute values of each STM trace without compromising normality.

This short term memory was chosen for three reasons: 1) its ability to perform current time processing, 2) its ability to encode temporal information in a continuous framework, and 3) its bias toward recency of events. Current-time processing is the exclusive processing of current measurements only at any given instant in time (deVRIES & PRINCIPE, 1992). The STM composed of leaky integrators operates on current inputs only during a single iteration. This constraint *forces* the short term memory to have some *dynamic* representation of time and obviates the need for predetermining an arbitrary number of previous values to remember.

The next advantage of this STM is the ability to encode temporal information in a *continuous framework* of representations. Since the LIs always receive input from a single external source, they can be semantically labeled according to their meaning in terms of that source. The tapped delay line, although useful in many instances, cannot claim this advantage because data are not assigned to time independent locations. For instance, tapped delay lines index their contents once per iteration regardless of the state of the input. As a result, the activation levels of a large number of locations change instantaneously. This discontinuous, high frequency change is built into the model. Due to the global shift in information, the semantic attribute (meaning) of each location in the memory changes frequently or does not exist. For supervised learning purposes, where the only objective is to produce a *unique* representation for each state of the process, the tapped delay line has its advantages. But for unsupervised, self-organizing learning, where information about degree of similarity is an integral part of the

organization process, they fall short. This result is simply due to the fact that one state of the STM describing the state of the process at one point in time may be very different than another state of the STM describing a very similar state. To put the situation in perspective, consider the following analogy. If grains of sand are added to a pile, one at a time, with the number of grains in the pile being recorded and assigned to a random symbol each time a new one is added, then the categorization of the resulting symbols would only succeed in discriminating between two different piles. No information about *degree of similarity* would be conveyed. As a result, the pair of symbols associated with two piles of 999 and 1000 grains would be as different as the pair of symbols associated with two piles of one and 1000 grains. The same holds true for monitoring dynamic systems. When all is said and done, the state of the STM at time *t* should not be much different than the state of the STM at time *t*+1. If not, then it cannot be categorized in a continuous framework of measurement according to its inherent features.

Temporal Pattern Recognition

A continuous-valued adaptive resonance theory (ART2) network (CARPENTER & GROSSBERG, 1987a) was used for temporal pattern recognition. It categorized the normalized STM vector discussed previously. Since the nodes were mutually exclusive, the state of the STM at any instant in time was represented by a single hemodynamic pattern category. Figure 10 is a rough, symbolic representation of the network.

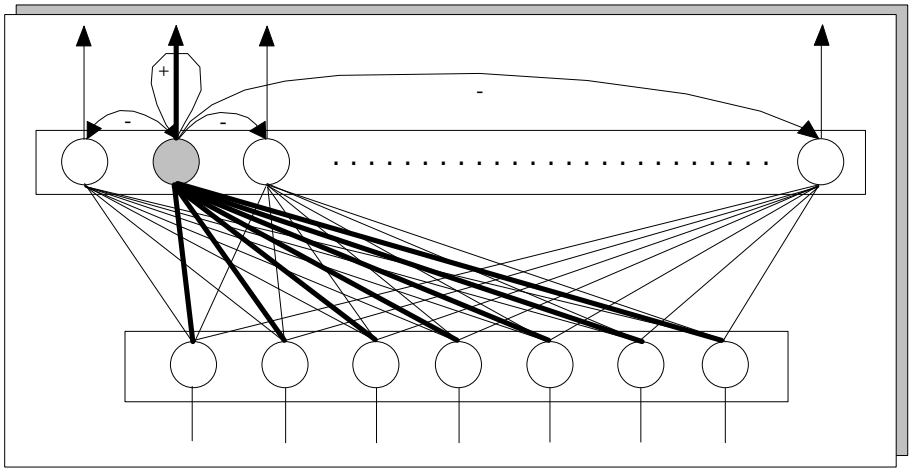


Fig. 10. Symbolic representation of the ART2 network for categorizing STM patterns. The bottom layer represents the STM and the top layer represents the temporal pattern categories. The darkened node in the top layer is shown to be laterally inhibiting the other nodes due to a strong activation from the STM layer.

The equations used in the simulations were not the actual ART2 equations, but they did have a similar effect. The STM vector ***s*** was categorized by calculating the degree to which it excited each node in the ART2 network using the dot product and choosing the one with the maximum activation $x_i(t)$. The index of the winning node $i^*(t)$ was chosen according to:

$$i^*(t) = i \quad \text{if} \quad x_i(t) = \max(x_i(t), \forall i) \tag{16}$$

When node $i^*(t)$ was determined to be the winner, it laterally inhibited all other nodes in the ART2 layer and rotated its own instar vector toward ***s***. As a result, the ART2 nodes self-organized to recognize the STM patterns conveyed by ***s*** over time. A more complete set of the equations used in the simulations are included in appendix D.

Temporal Pattern Prediction

For prediction purposes, a set of neurons with adaptive spectral timing connections was created on a one-to-one basis with the each ART2 node as shown in figure 11. In addition to being

connected to a single ART2 node, each node in the adaptive spectral timing layer was connected to every other node in the same layer by 2*K* synapses (*K* synapses in each direction).

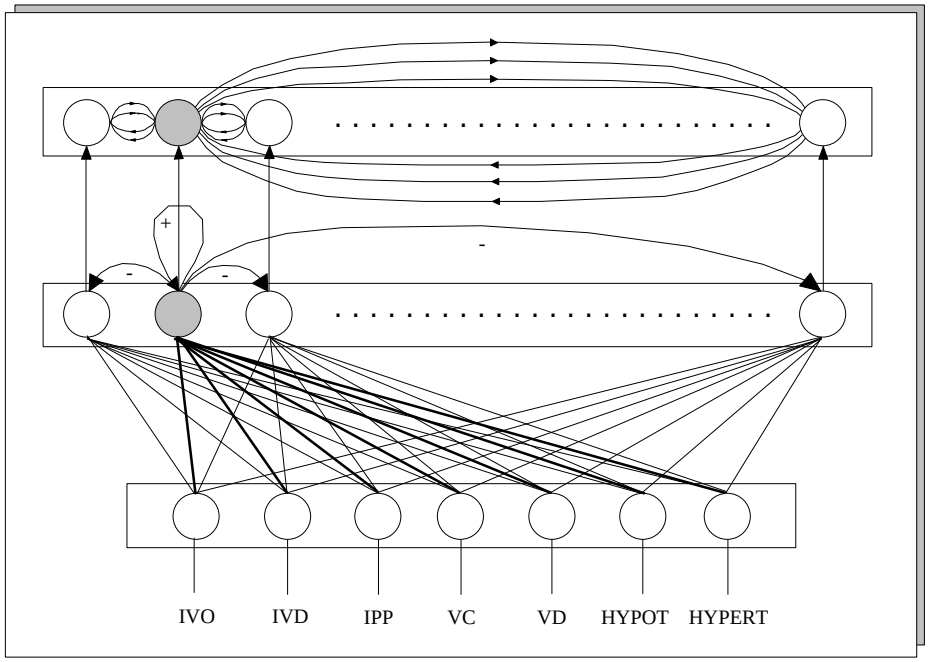


Fig. 11. The complete model. Three distinct layers are present: 1) short term memory, 2) temporal pattern recognition, and 3) temporal pattern prediction.

As shown by figure 12, the general synapse was denoted by z_{ijk}, for $1 \le i \le M$, $1 \le j \le M$, and $1 \le k \le K$ where M was the number of ART2 categories and K was the number of discrete time units into the future that one node predicted future hemodynamic patterns.

Discrete versions of the adaptive spectral timing equations were used in the simulations. In discrete time, the pathways functioned as follows. When the *i*th hemodynamic pattern occurred and neuron *i* fired at time *t*, synapse z_{ij1} was activated maximally at time t+1, z_{ij2} was activated maximally at time t+2, and so forth. Depending on when the *j*th hemodynamic pattern occurred, individual synapses were strengthened.

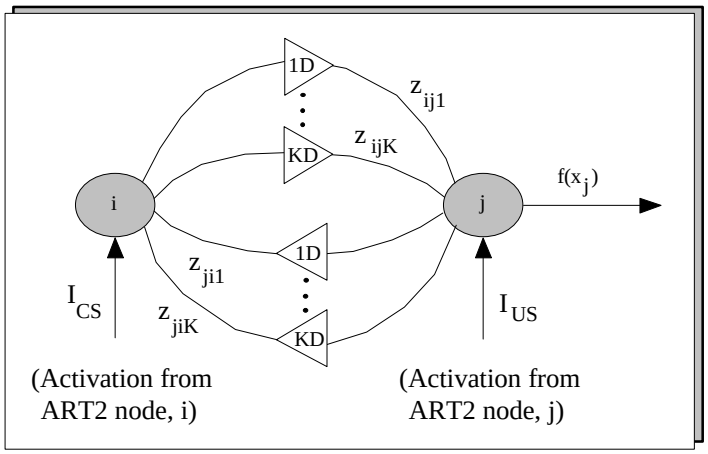


Fig. 12. A set of adaptive spectral timing pathways between the *i*th and *j*th neurons.

The adaptive spectral timing model was able to adaptively find the ISI between two hemodynamic patterns as long as it was not longer than the predictive reach of the model, *K*. For example, if the *j*th hemodynamic pattern usually occurred *k* time units after the *i*th hemodynamic pattern and $k<K$, then the efficacy of synapse z_{ijk} increased over time while the other synapses atrophied. However, if the *j*th hemodynamic pattern occurred $K+1$ time units after the *i*th hemodynamic pattern, no synapses were strengthened and no temporal relationship was encoded.

The adaptrode synapse model (MOBUS 1990) served as the actual LTM trace of the spectral timing equations. It served two purposes: 1) to protect the synapse against transient (local) temporal associations and 2) to drive the synaptic potential to saturation when it experienced prolonged reinforcement. The adaptrode synapse modeled long term potentiation processes such as those which increase the availability of chemical substrates required for the production of neurotransmitter during prolonged periods of stimulation. Essentially, these long term processes acted as *mass* or *capacitance* which introduced phase delays in the response of the synapse to external forces. Consequently, the synapse tended to remain in a steady state in a varying environment until or unless forced to change by a new persistent force (or lack thereof). Without the adaptrode synapse, a temporal relationship that would have formed over a long period

of time would have been quickly forgotten when the relationship ceased to occur. In other words, the adaptrode synapse *decoupled* the AST network from local inconsistencies.

The adaptrode synapse equation used in the place of the LTM trace of the adaptive spectral timing equations is given by:

$$z_{ijk}^{(l)}(t+1) = z_{ijk}^{(l)}(t) + g_{ijk}^{(l)}(t)\varepsilon_k^{(l)}[z_{ijk}^{(l-1)}(t)-z_{ijk}^{(l)}(t)] - \gamma_k^{(l)}[z_{ijk}^{(l)}(t)-z_{ijk}^{(l+1)}(t)] \quad (17)$$

where $z_{ijk}^{(l)}$ represented the lth potential of the kth synapse from the ith neuron to the jth neuron for $1 \le i \le M$, $1 \le j \le M$, $1 \le k \le K$, and $1 \le l \le L$. The first potential $z_{ijk}^{(1)}$ represented the strength of synapse as a whole and served as the LTM trace z_{ijk} of the adaptive spectral timing equations. Therefore $z_{ijk} = z_{ijk}^{(1)}$. The remaining potentials were used to calculate the first potential. For the first and last potentials, 'pull up' and 'pull-down' reference values were defined respectively by:

$$z_{ijk}^{(l-1)}(t) = \left[z^{(max)}\right] \text{ if } l=0, \left[z_{ijk}^{(l-1)}(t)\right] \text{ otherwise} \quad (18)$$

$$z_{ijk}^{(l+1)}(t) = \left[z^{(min)}\right] \text{ if } l=L, \left[z_{ijk}^{(l+1)}(t)\right] \text{ otherwise} \quad (19)$$

where $z^{(min)} = 0$ and $z^{(max)} = 1$.

The learning constants (constant with respect to time) were defined to be functions of k as given by:

$$\varepsilon_k^{(l)} = \exp(-C(k-1))(D-El) \quad (20)$$

$$\gamma_k^{(l)} = D-El \quad (21)$$

The excitatory learning constant, $\varepsilon_k^{(l)}$, was made to obey a decaying exponential with respect to index k, multiplied by a linear function of index l with decreasing slope. As a result of the exponential, learning followed a recency gradient where successful predictions based on recent

temporal patterns were learned more rapidly than successful predictions based on earlier temporal patterns. The linear factor (*D-El*) made $\varepsilon_k^{(l)}$ decrease with increasing *l*, thus modeling the increasing excitatory time constants of long term potentials. The same linear factor was used to define $\gamma_k^{(l)}$, thus modeling the increasing passive decay times of the same long term potentials.

The learning of each potential was gated by the binary signal, $g_{ijk}^{(l)}(t)$. For l=1, $g_{ijk}^{(l)}(t)$ was set to unity only if the synapse fired at the same time the local hemodynamic pattern occurred. Consequently, $g_{ijk}^{(1)}$ was determined by the agreement between the ISI between the occurrence of the *i*th and *j*th hemodynamic patterns and the activation rate of the synapse as expressed by:

$$g_{ijk}^{(l)}(t) = \left[F_{ijk}(t)G_j(t)\right] \text{ if } l=1, \left[1\right] \text{ otherwise} \qquad (22)$$

where $F_{ijk}(t)$ and $G_j(t)$ were binary variables which represented the prediction and occurrence of the *j*th hemodynamic pattern, respectively. The first binary variable, $F_{ijk}(t)$, represented the prediction of the *j*th hemodynamic pattern by the *i*th hemodynamic pattern if it occurred *k* time units earlier as shown by:

$$F_{ijk}(t+1) = \left[F_{ijk}(t) + \delta_k(t-t_c);\ t_c=t+k\right] \text{ if } i = a^*, \left[F_{ijk}(t)\right] \text{ otherwise} \qquad (23)$$

where the function, $\delta_k(t-t_c)$, represented the Kronecker Delta which equals unity when its argument equals zero as shown by:

$$\delta_k(t-t_c) = 1 \text{ if } t=t_c, 0 \text{ otherwise} \qquad (24)$$

The second binary variable, $G_j(t)$, represented the occurrence of the *j*th hemodynamic pattern at time *t*:

$$F_j(t) = 1 \text{ if } j = i^*(t), 0 \text{ otherwise} \qquad (25)$$

where $i^*(t)$ represented the winning hemodynamic pattern at time t. When both variables, $F_{ijk}(t)$ and $G_j(t)$ equaled unity, the Hebbian condition was satisfied and the binary gate, $g_{ijk}^{(l)}(t)$, was opened (set to unity). As an example, when the jth hemodynamic pattern occurred then $G_j(t)=1$. If this pattern had been predicted k time units earlier by the ith hemodynamic pattern, then $F_{ijk}(t)=1$. As a result, $g_{ijk}^{(1)}=1$ and $z_{ijk}^{(1)}$ increased. On the other hand, if the activation rate did not agree with the local ISI, then $g_{ijk}^{(1)}=0$ and no increase in $z_{ijk}^{(1)}$ occurred.

The adaptrode synapse served to drive the first potential to saturation when the Hebbian condition was satisfied frequently. If the adaptrode model had not been used, z_{ijk} would have never reached its saturation value regardless of how much it was stimulated. Instead, it would have saturated at a value of $\varepsilon_k^{(1)}/(\varepsilon_k^{(1)+}\gamma_k^{(1)})$.

In order to predict future trends, the incoming signals to each neuron in the adaptive spectral timing network were summed $t_p=1$ time units in advance as given by:

$$y_j(t) = \sum_{i=1}^{M}\sum_{k=1}^{K}[E_{ijk}(t+t_p)\, z_{ijk}(t)] \tag{26}$$

Equation (26) is assumed to be an original contribution of the author. For each neuron, $y_j(t)$ represented the probability that the jth hemodynamic pattern would occur at $t=t+t_p$. The maximum $y_j(t)$, for $1 \le j \le M$, indicated the most probable hemodynamic pattern j^* at $t=t+t_p$ as shown by:

$$j^* = j \quad \text{if} \quad y_j(t) = \max(y_j(t), \forall_j) \tag{27}$$

RESULTS

The system was simulated using two sets of data: 1) a fabricated calibration set and 2) real clinical data. For the purpose of recognizing instantaneous physiologic conditions, four physiologic variables were measured (clinical data set) or fabricated (calibration set):

1. Heart rate (HR) (BPM)
2. Systolic arterial pressure (SAP) (mmHg)
3. Diastolic arterial pressure (DAP) (mmHg)
4. Diastolic pulmonary artery pressure (PAD) (mmHg)

The instantaneous values and rates of change of these variables were used to recognize seven physiologic events:

1. Intravascular volume overload
2. Intravascular volume depletion
3. Increased pericardial pressure
4. Vasoconstriction
5. Vasodilation
6. Hypotension
7. Hypertension

Of these events, six were the same as those recognized by DYNASCENE. The equations used to calculate or estimate intermediate variables such as systemic vascular resistance (SVR), pulse pressure (PP), stroke volume (SV), cardiac output (CO), central venous pressure (CVP), pulsus paradoxus, and reverse pulsus paradoxus can be found in appendix B.

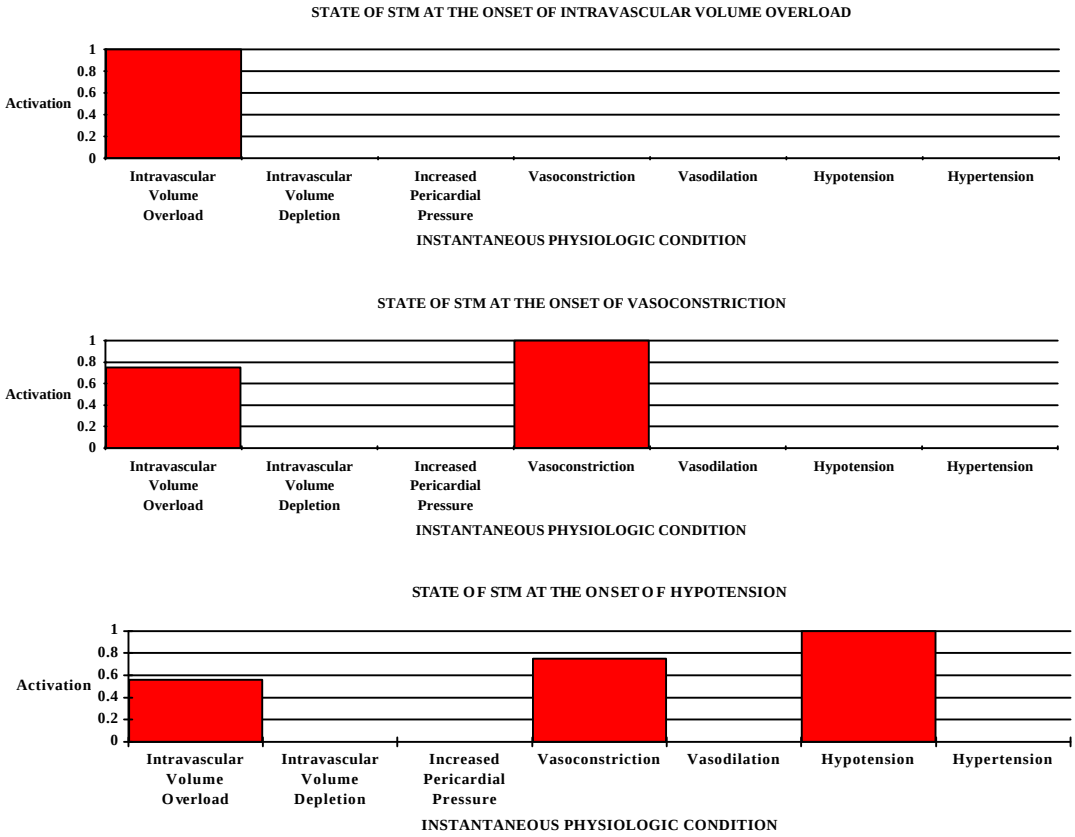
Calibration

For calibration purposes, the model was trained on the same five hemodynamic sequences recognized by DYNASCENE: 1) congestive heart failure with intravascular volume overload, 2)

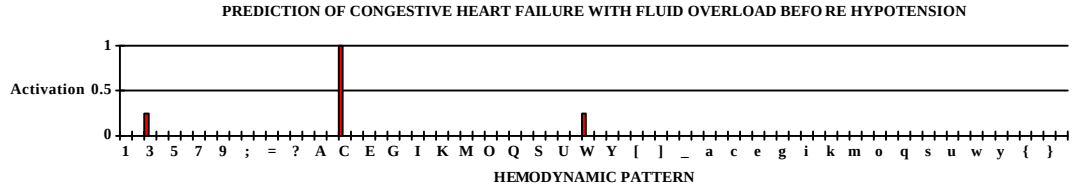
congestive heart failure due to pump failure, 3) hypovolemia, 4) sepsis, and 5) cardiac tamponade. Raw data was fabricated to produce the required sequence of events. The main parameters of the simulation were: A=.25, B=1, C=.5, D=.11, E=.01, K=32, L=3, and M=78.

Part of the results of the calibration are illustrated in figures 13 and 14. These graphs show how the system reacted to two hemodynamic sequences: congestive heart failure with fluid overload and congestive heart failure due to pump failure. Of the five sequences used for calibration, these two were chosen because they represent two different permutations of the same combination of physiologic events.

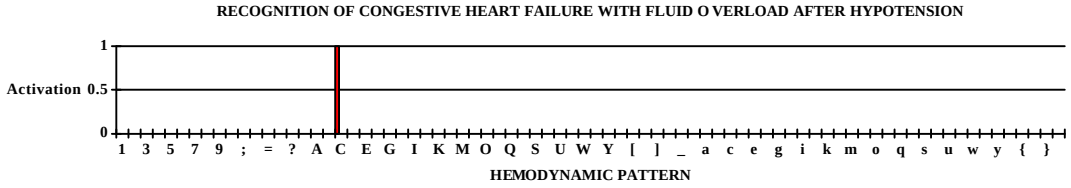
In part (a) of figure 13 the progressive states of the STM during the unfolding of congestive heart failure with fluid overload are shown. The first graph in (a) shows the occurrence of intravascular volume overload after a long period of seeing no events. The second graph shows the occurrence of vasoconstriction. Notice that intravascular volume overload is no longer occurring; therefore the activation of the first neuron has decayed by a small amount. The third graph shows the occurrence of hypotension. Notice that the memory traces of intravascular volume overload and vasoconstriction have decayed by amounts that correlate with the elapsed time since their occurrence. As a result, the state of the STM conveys the order in which the events occurred. Part (b) shows the prediction of hemodynamic pattern C after seeing only the first two events. Hemodynamic pattern C was one of the temporal patterns which represented congestive heart failure with fluid overload. Part (c) shows the eventual recognition of hemodynamic pattern C after seeing the third and final event. Figure 14, which shows how the system reacted to congestive heart failure due to pump failure, is structured in the same manner as figure 13. Notice that in parts (b) and (c) the hemodynamic pattern that was predicted and eventually recognized was not hemodynamic pattern C, although the same combination of physiologic events occurred.



(a)

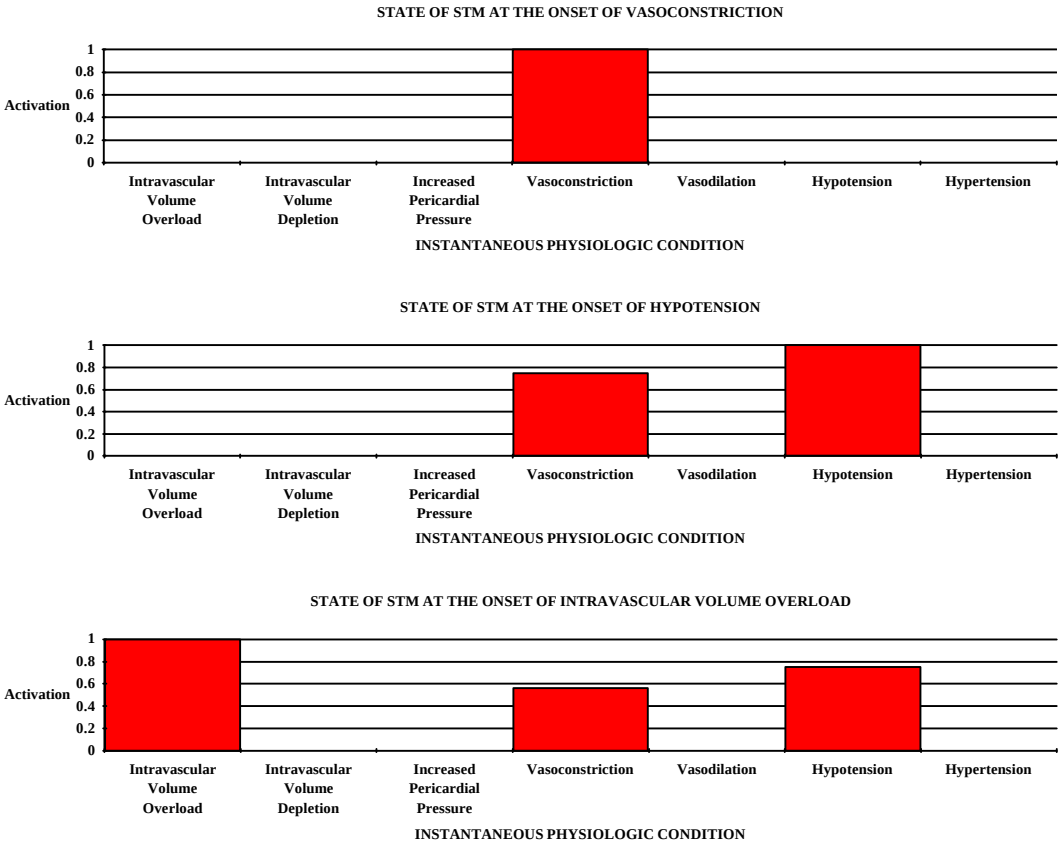


(b)

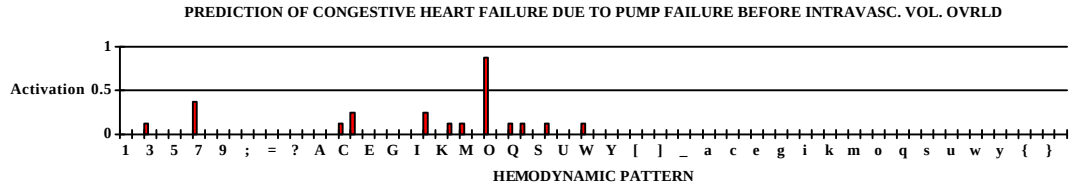


(c)

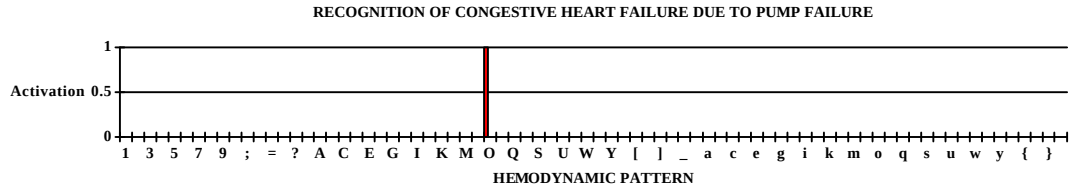
Fig. 13. Prediction and recognition of congestive heart failure with fluid overload. In (a) the progression of the state of the STM is shown. After seeing only the first two events, the system predicted hemodynamic pattern C as shown in (b). After seeing the third condition, the system recognized hemodynamic pattern C in (c) which was one of the categories which represented congestive heart failure with fluid overload. The x-axis in (b) and (c) represents all possible hemodynamic patterns.



(a)



(b)



(c)

Fig. 14. Prediction and recognition of congestive heart failure due to pump failure. In (a) the progression of the state STM is shown. After seeing only the first two events, the system predicted hemodynamic pattern O as shown in (b). After seeing the third condition, the system recognized hemodynamic pattern O in (c) which was one of the categories which represented congestive heart failure due to pump failure. The x-axis in (b) and (c) represents all possible hemodynamic patterns.

Application

The system was tested on real clinical data measured at five minute intervals using the same parameters: *A*=.25, *B*=1, *C*=.5, *D*=.11, *E*=.01, *K*=32, *L*=3, and *M*=78. A full 24 hours of data (288 five-minute samples) from a cardiac patient was analyzed. As expected with five-minute interval samples, a large amount of variability was observed from one sample to the next. Three events occurred repeatedly: intravascular volume overload, vasodilation, and hypertension. One interesting sequence of events is shown in figure 15. Notice that hypotension occurred only once. From this sequence several interesting subpatterns were discovered:

1. Vasodilation & Hypotension → Hypertension → Intravascular Volume Overload
2. Hypotension → Hypertension → Intravascular Volume Overload & Vasodilation
3. Hypotension → Hypertension → Vasodilation → Intravascular Volume Overload
4. Hypotension → Vasodilation → Intravascular Volume Overload & Hypertension
5. Hypotension → Vasodilation → Hypertension → Intravascular Volume Overload
6. Hypotension → Hypertension → Intravascular Volume Overload → Vasodilation

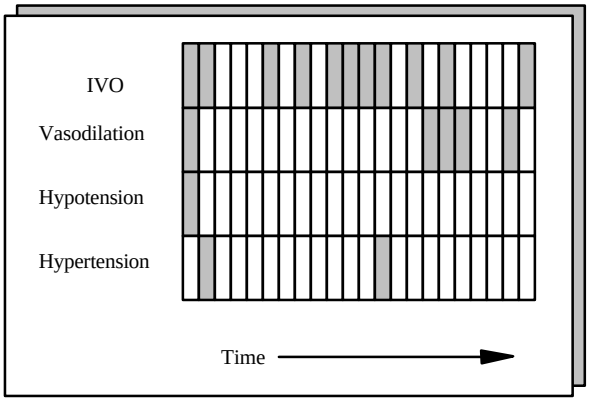


Fig. 15. Timing graph of physiologic conditions recognized in actual clinical data.

Although the events of these sequences occurred in the orders indicated, many did not occur in a contiguous fashion. Instead, they were mixed in with other conditions. As a result,

more than one ART2 node could represent the same hemodynamic pattern. Variations of single permutations of physiologic events were represented by *sets* of ART2 nodes. Although not shown in the list, the time-between-event information was retained by each hemodynamic pattern category, such that two hemodynamic patterns with identical permutations but different ISIs would still be distinguished. For example, hemodynamic pattern 6 in the list shows four distinct physiologic conditions occurring in serial order. The occurrence of hypotension and hypertension were separated from intravascular volume overload and vasodilation by several time units. This can be seen by the actual synaptic vector of hemodynamic pattern 6 as shown in figure 16. Notice the difference in heights between intravascular volume overload and hypertension. The difference between activations indicates that these two sets of events were separated by more than a single time unit. If the contiguous version of same sequence had occurred, the STM vector would have been different enough from this vector to justify a new category, thus distinguishing between the two variations of the same sequence.

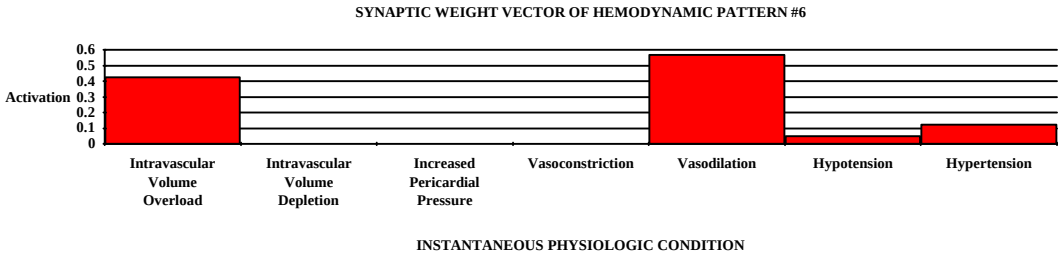


Fig. 16. Synaptic weight vector of hemodynamic pattern #6. This pattern category shows the sequence: hypotension → hypertension → intravascular volume overload → vasodilation with a lag of more than one time unit between the first two conditions and the last two.

DISCUSSION

A very interesting property of the interconnected adaptive spectral timing network emerged in the calibration process: the ability to predict sequences based on *rhythm*. At first, this property was alarming, but upon further reflection it seems natural for such a network that is so deeply rooted in discovering interstimulus intervals, to be 'lulled into a routine'. If hemodynamic sequence *B* was frequently made to occur *k* time units after hemodynamic sequence *A*, the AST network learned to predict sequence *B* based on seeing sequence *A*. If hemodynamic sequence *B* was suddenly removed from the input stream altogether, it was still predicted by the recognition of sequence *A*. Even though the first couple of events of sequence *B* did not occur, the network still predicted the last events of sequence *B*. This result is interesting because the system predicted sequence *B without seeing any of the events in sequence B*. If sequence *B* represented congestive heart failure, for example, then the system would have predicted congestive heart failure without seeing the events associated with congestive heart failure, even though the state of the STM had completely decayed since the onset of sequence *A*. Essentially, the network detached itself from cues of reality in confidence that sequence *B* would continue to occur after sequence *A*. Of course the system never *recognized* sequence *B*, it just *predicted* it. This feature could prove very useful in discovering hemodynamic sequences that are longer than the those simulated in this study.

Although this 'biorhythmic-like' property could be very functional in practice, it was not allowed to be used as a 'crutch' in the calibration process. Each sequence was separated by 40 time units of normal condition (no physiologic events) in order to keep the network from learning to predict the sequences based on rhythm. Separating the sequences by more than the predictive reach of the network forced the network to recognize the sequences based on the inherent features of the sequences *themselves*, rather than man-made periodicity. This decoupling scheme worked as long as the ART2 network was not too vigilant, in which case the winning hemodynamic pattern category (ART2 node) changed frequently and the predictive reach of the network was extended.

Another interesting property of the predictive network was its ability to predict events that occurred at varying frequencies. For varying ISIs between two events, the first derivative of the predictive potentials $y_j(t)$ with respect to time sometimes correlated more closely with the probability of occurrence than did absolute magnitude. Using the previous example, if sequence *B* usually followed sequence *A* 22-28 time units later, then the predictive potentials of the nodes that recognized variations of sequence *B* tended to *grow* during the 22-28 time unit window. It is expected that this feature would be extremely useful for discovering real world hemodynamic patterns in the clinic.

LIMITATIONS

As previously mentioned, several assumptions and estimations were made for some variables. Central venous pressure, for example, was estimated at 12.0 mmHg and a simplistic linear relationship between pulse pressure and stroke volume was used. Although not an exact estimate, SV increased when PP increased and decreased when PP decreased.

These estimated variables were used to recognize the seven physiologic events. Consequently, the instantaneous recognition of the physiologic conditions was also just an estimate. For calibration purposes, the question of how the physiologic conditions were recognized was not an issue. As long as the model *thought* it was recognizing the events in the desired order, the goal of calibrating the temporal pattern recognition and prediction processes was accomplished.

The analysis of the clinical data was a little different. Because neither pulsus paradoxus nor reverse pulsus paradoxus were being recognized by automatic data collection devices available at the time of the simulations, both conditions were assumed to be absent. And as it turned out, intravascular volume overload was recognized frequently, partly because pulsus paradoxus was assumed to be absent. Since the criteria for recognizing increased pericardial pressure was the same as that for recognizing intravascular volume overload with the exception of the presence of pulsus paradoxus, it was unknown which condition really occurred. Consequently, the occurrence of intravascular volume overload should be taken lightly. The analysis of the clinical data only demonstrated that the model was capable of recognizing various permutations of physiologic events in general. This system should be integrated with more robust methods, such as Kalman filtering for recognizing the fundamental physiologic events before being applied in the clinical environment.

CONCLUSIONS AND RECOMMENDATIONS

The collection of artificial neural networks presented in this study are able to detect hemodynamic patterns without being programmed to do so explicitly. It is expected that such a system, combined with robust methods for preprocessing and cross verifying raw data, can be operated in the clinical environment in real time to assist clinicians with caring for critically ill patients. Because the system learns self-organized representations of input data, it requires no expert knowledge *a priori*, other than the preprocessing required to recognize fundamental physiologic events. As a result, it is able to discover unknown hemodynamic patterns in addition to ones that are well known.

In addition to the ability to recognize hemodynamic patterns, the model is able to predict them before they completely unfold in time. A hemodynamic pattern that is defined by a permutation of several events, for example, can be predicted after the first couple of events. Other hemodynamic patterns that might occur regularly, or even semi-regularly, can be predicted if they fall within the predictive reach of the network.

In the simulations, a K=78 hemodynamic pattern limit was imposed. For calibration and application purposes, this limit was very reasonable. However, in the clinical setting, it is recommended that this limit be extended, depending on how many different hemodynamic patterns it is expected to learn. If the system is to be trained on data from a limited number of patients, then K=78 might be adequate. However, if the system is intended to continuously monitor many patients, K would most likely need to be increased.

REFERENCES

Anderson, J.A. and Rosenfeld, E. (1988) *Neurocomputing: Foundations of Research*, MIT Press, Cambridge, MA.

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

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

Carpenter, G.A. and Grossberg, S. (1988) The ART of adaptive pattern recognition by a self-organizing neural network, *Computer*, 21, 77-88.

Carpenter, G.A. (1989) Neural network models for pattern recognition and associative memory, *Neural Networks*, 2, 243-257.

Carpenter, G.A. and Grossberg, S. (1991) *Pattern Recognition by Self-Organizing Neural Networks*, MIT Press, Cambridge, MA.

Cohen, M.A. (1992) The construction of arbitrary stable dynamics in nonlinear neural networks, *Neural Networks*, 2, 5, 331-349.

Cohn, A.I., Rosenbaum, S., Factor, M., and Miller, P.L. (1990) DYNASCENE: An approach to computer-based intelligent cardiovascular monitoring using sequential clinical "scenes", *Methods of Information in Medicine*, 29, 122-131.

de Vries, B. and Principe, J.C. (1992) The gamma model - A new neural model for temporal processing, *Neural Networks*, 5:565-576.

Eberhart, R.C. and Dobbins, R.W. (1991) Proc. 13th Ann. Int. Conf. IEEE Eng. in Med. & Biol. Soc., Section 4, 1470-1471.

Elman, J.L. and Zipser, D. (1988) Learning the hidden structure of speech, *Journal of the Acoustical Society of America*, 83, 1615-1626.

Fu, L. (1991) A hybrid medical expert system, Proc. 13th Ann. Int. Conf. IEEE Eng. in Med. & Biol. Soc., Section 4, 1290-1291.

Gallant, S.I. (1988) Connectionist expert systems, *Communications of the ACM*, 2, 31, 152-169.

Gjerdingen, R.O. (1992) Learning syntactically significant temporal patterns of chords: A masking field embedded in an ART3 architecture, *Neural Networks*, 5, 551-564.

Grossberg, S. (1968) Some nonlinear networks capable of learning a spatial pattern of arbitrary complexity, *Proceedings of the National Academy of Sciences USA*, 59, 368-372.

Grossberg, S. (1972) Neural expectation: Cerebellar and retinal analogs of cells fired by learnable or unlearned pattern classes, *Kybernetik*, 10, 49-57.

Grossberg, S. (1982) *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*, Boston: Reidel Press.

Grossberg, S. (1988) Nonlinear neural networks: Principles, mechanisms, and architectures, *Neural Networks*, 1, 17-61.

Hebb, D.O. (1949) *The organization of behavior*, New York: Wiley.

Hertz, J., Krogh, A., & Palmer, R.G. (1991). *Introduction to the Theory of Neural Computation*, Santa Fe Institute: Studies in the Sciences of Complexity, New York: Addison-Wesley Publishing Company.

Hirsch, M.W. (1989) Convergent activation dynamics in continuous time networks, *Neural Networks*, 2, 5, 331-349.

Hodgkin, A.L. and Huxley, A.F. (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve, *Journal of Physiology*, 117, 500-544.

Hopfield, J.J. (1982) Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences USA*, 79, 2554-2558.

Hopfield, J.J. (1984) Neurons with graded responses have collective computational properties like those of two-state neurons, *Proceedings of the National Academy of Sciences, USA*, 81, 3088-3092.

Kasabov, N.K. (1990) Hybrid connectionist rule-based systems, in: Ph. Jorrand and S. Segurev (Eds.), *Artificial Intelligence IV: Methodology, Systems, Applications* (Elsevier Science Pub., North-Holland, 227-235.

Kohonen, T. (1982) Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, 43, 59-69.

Kohonen, T. (1989) Self-organization and associative memory (3rd ed.), Berlin: Springer-Verlag.

Kosko, B. (1987) Constructing an associative memory, *Byte*, September, pp. 137-144.

Kosko, B. (1988) Bidirectional associative memories, *IEEE Transactions on Systems, Man, and Cybernetics*, 18, 1, January-February 1988, 141-152.

Kosko, B. (1992) *Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence*, Prentice Hall, New Jersey.

Kung, S.Y. (1993) *Digital Neural Networks*, PTR Prentice Hall, New Jersey.

Liow, R. and Vidal, J.J. (1991) A dual network expert system, *IEEE*, 1670-1674.

Lippmann, R.P. (1989) Review of neural networks for speech recognition, *Neural Computation*, 1, 1-38.

McClelland, J.L. and Elman, J.L. (1986) Interactive processes in speech perception: The TRACE model, In *Parallel Distributed Processing*, vol. 2, chap. 15.

McCulloch, W.S. and Pitts, W. (1943) A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, 9, 127-147.

Miller, A.S., Blott, B.H., and Hames, T.K. (1992) Review of neural networ applications in medical imaging and signal processing, *Medical & Biological Engineering & Computing*, 30, 449-464.

Mobus, G.E. (1990) The adaptrode learning model: applications in neural network computing. *Technical Report* CRPDC-90-5, Center for Parallel and Distributed Computing, U. of North Texas, Denton.

Mobus, G.E. (1992) Toward a theory of learning and representing causal inferences in neural networks, In *Neural Networks for Knowledge Representation and Inference*, D.S. Levine & M. Aparicio (Eds.), Lawrence Erlbaum, Inc., Hillsdale, New Jersey.

Papadourakis, G.M., Gaga, E., Vareltzis, G., and Bebis, G. (1992) Use of artificial neural networks for clinical decision-making (maldescensus testis), *IEEE*, 159-164.

Pavlov, I.P. (1927) *Conditioned reflexes*, London: Oxford University Press.

Poli, R., Cagnoni, S., Livi, R., Coppini, G., and Valli, G. (1991) A neural network expert system for diagnosing and treating hypertension, IEEE Computer, March, 64-71.

Reiss, M. and Taylor, J.G. (1991) Storing temporal sequences, *Neural Networks*, 4, 773-787.

Rialle, V., Ohayon, M., Amy, B., and Bessiere, P. (1991) Medical knowledge modeling in a symbolic-connectionist perspective, Proc. 13th Ann. Int. Conf. IEEE Eng. in Med. & Biol. Soc., Section 4, 1109-1110.

Rosenblat, F. (1958) The perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review*, 65, 386-408.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986) Learning internal representations by error propagation, In D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructures of cognitions*, I, Cambridge, MA: MIT Press.

Shimohara, K., Uchiyama, T., and Tokunaga, Y. (1993) Subconnection neural network for event-driven temporal sequence processing, *Neural Networks*, 6, 709-718.

Stubbs, D.F. (1990) Multiple neural network approaches to clinical expert systems, *SPIE Appl. of Artif. Neural Networks*, 1294, 433-441.

Tank, D.W. and Hopfield, J.J. (1987) Concentrating information in time: Analog neural networks with application to speech recognition problems, In *IEEE First International Conference on Neural Networks* (San Diego 1987), eds. M. Caudill and C. Butler, New York: IEEE, vol. IV, 455-468

Tian He and Tai Juwei (1989) Connectionist traditional Chinese medicine expert system - NCCS, in: B. Barber, D. Cao, D. Qin, and G. Wagner (Eds.), *Proc. MEDINFO 89 - Sixth Int. Conf. on Medical Informatics* (North-Holland, Amsterdam, Netherlands), 1186.

Tesauro, G. (1986) Simple neural models of classical conditioning, *Biological Cybernetics*, 55, 187-200.

Waibel, A. (1989) Modular construction of time-delay neural networks for speech recognition, *Neural Computation*, 1, 39-46.

Werbos, P.J. (1990) Backpropagation through time: what it does and how to do it, *Proc. of the IEEE*, 87, 10.

Widrow, B. and Hoff, M.E. (1960) Adaptive switching circuits, 1960 IRE WESCON Convention Record, part 4, 96-104.

Williams, R.J. and Zipser, D. (1989) A learning algorithm for continually running fully connected networks, *Neural Computation*, 1, 270-280.

APPENDIX A

LIST OF VARIABLES, PARAMETERS, AND FUNCTIONS

System Variables, Parameters, Vectors, and Functions

A	global decay rate of STM nodes
B	global excitatory saturation level of STM nodes
C	global decay rate constant of excitatory learning rate constant of adaptrode-based synapses
D	global y-intercept of inhibitory learning rate constant function of adaptrode-based synapses
E	global slope of inhibitory learning rate constant function of adaptrode-based synapses
$F_{ijk}(t)$	binary occurrence variable of the prediction of the jth hemodynamic pattern by the ith hemodynamic pattern at time t, k time units ago
$G_j(t)$	binary occurrence variable of the jth hemodynamic pattern at time t
$g_{ijk}^{(l)}$	binary gating signal of the lth potential of the kth predictive pathway from the ith to the jth AST node
i	index of the ith ART2 or AST node
$i^*(t)$	index of winning ART2 node (hemodynamic pattern) at time t
j	index of the jth ART2 or AST node
k	index of the kth predictive pathway from one AST node to another
K	maximum number of AST pathways from one predictive node to another
l	index of the lth potential of a predictive pathway from one AST node to another
L	maximum number of potentials per adaptrode synapse
$I_p(t)$	binary occurrence variable of the pth physiologic condition

M maximum number of ART2 and AST nodes (each)

p index of the pth instantaneous physiologic condition

P maximum number of instantaneous physiologic conditions

$s_p(t)$ activation of the pth instantaneous physiologic condition node in STM

$y_j(t)$ probability activation of the jth hemodynamic pattern

$z_{ijk}(t)$ strength of the kth predictive pathway from the ith to the jth AST node (this variable is equal to $z_{ijk}^{(1)}(t)$)

$z_{ijk}^{(l)}(t)$ lth potential of the kth predictive pathway from the ith to the jth AST node

$z^{(max)}$ global positive saturation level of the first potential of a predictive pathway from one AST node to another

$z^{(min)}$ global negative saturation level of the first potential of a predictive pathway from one AST node to another

$e_k^{(l)}$ global excitatory learning constant of the lth potential of the kth predictive pathway from one AST node to another

$g_k^{(l)}$ global inhibitory learning constant of the lth potential of the kth predictive pathway from one AST node to another

$\delta_k(t\text{-}t_c)$ Kronecker delta function

$r(t)$ global vigilance variable of ART2 categorization

APPENDIX B

INSTANTANEOUS RECOGNITION OF PHYSIOLOGIC CONDITIONS

In order to recognize the seven physiologic conditions: 1) intravascular volume overload, 2) intravascular volume depletion, 3) increased pericardial pressure, 4) vasoconstriction, 5) vasodilation, 6) hypotension, and 7) hypertension, several variables were needed: heart rate (HR), mean arterial pressure (MAP), pulmonary artery diastolic pressure (PAD), cardiac output (CO), systemic vascular resistance (SVR), pulsus paradoxus, and reverse pulsus paradoxus. Intravascular volume overload was recognized when pulsus paradoxus was absent and PAD was rising and greater than or equal to 25 mmHg. Intravascular volume depletion was recognized when reverse pulsus paradoxus was present and PAD was falling and less than or equal to 15 mmHg. Increased pericardial pressure was recognized when pulsus paradoxus was present and PAD was rising and greater than or equal to 25 mmHg. Vasoconstriction was recognized when SVR was rising, HR was greater than or equal to 90 BPM, and CO was less than 4 L/beat. Vasodilation was recognized when CO was rising and greater than or equal to 7 L/beat, SVR was falling, and HR was greater than or equal to 90 BPM. Hypotension was recognized when MAP was falling and less than 70 mmHg and HR was rising. Hypertension was recognized when MAP was rising and greater than 100 mmHg and HR was falling.

In order to calculate systemic vascular resistance (SVR), the following quantities were needed: 1) MAP, 2) central venous pressure (CVP), and 3) CO. None of these values were being measured by automatic devices available at the time of the simulation so they were calculated where possible and estimated when not. MAP was calculated from pulse pressure (PP) and diastolic arterial pressure (DAP) by:

$$MAP = DAP + .333(PP) \tag{B1}$$

where *PP* was calculated by:

$$PP = SAP\text{-}DAP \quad \text{(B2)}$$

SVR was calculated from *MAP*, *CVP*, and *CO* by:

$$SVR = 79.9*(MAP\text{-}CVP)/CO \quad \text{(B3)}$$

where *CVP* was estimated to be equal to its average value of 12.0 mmHg and CO was calculated from heart rate (HR) and stroke volume (SV):

$$CO = .001*HR*SV \quad \text{(B4)}$$

where *SV* was estimated from *PP* using a simple linear relationship:

$$SV = 1.4*PP \quad \text{(B5)}$$

Although a simplistic assumption, *SV* behaved like *PP*, increasing when *PP* increased and decreasing when *PP* decreased.

APPENDIX C
ARTIFICIAL NEURAL MODELS

During the past decade, artificial neural networks have been recognized for their extraordinarily surprising abilities in cognition, perception, organization, classification, feature extraction, pattern recognition, associative memory, data compression, dimensionality reduction, projection, and prediction (McCULLOCH & PITTS, 1943; ROSENBLATT, 1958; WIDROW & HOFF, 1960; HOPFIELD, 1982, 1984; KOHONEN, 1982; RUMELHART, *et al*., 1986; CARPENTER & GROSSBERG, 1986, 1987; KOSKO, 1987, 1988, 1992). Inspired by real biological neurons and interconnections, researchers have developed a cornucopia of ANN models that mimic biological functions and perform intelligent tasks. These recent developments have attracted interest from such diverse fields as physics, engineering, medicine, mathematics, computer science, biology, economics, and psychology.

The Artificial Neuron

Illustrated in figure C1 is a typical artificial neuron that appears frequently in the literature. The simplest neural networks employ these units at the most fundamental level of processing (McCULLOCH & PITTS, 1943; ROSENBLATT, 1958; WIDROW & HOFF, 1960, KOHONEN, 1981, RUMELHART, *et al*., 1986). The model shown in figure C1 has grown out of a simple model introduced by McCULLOCH & PITTS in 1943.

Each neuron possesses an *m*-dimensional *instar* vector $\boldsymbol{w}$ (GROSSBERG, 1972) which represents the efficacies of its incoming synapses (connections) and an *outstar* vector $\boldsymbol{v}$ (GROSSBERG, 1968). Incoming signals, represented by vector $\boldsymbol{S}$, are amplified or attenuated according to corresponding weights in $\boldsymbol{w}$ and summed to produce a membrane potential, or activation, x_j. The membrane potential x_j is processed by a monotonically increasing output function $f(x_j)$ which activates $\boldsymbol{v}$ accordingly. The neuron essentially *imposes* $\boldsymbol{v}$ on downstream neurons to a degree determined by the activation through $\boldsymbol{w}$. As a result, there is an association

between vectors ***w*** and ***v***. Sometimes ***v*** is one-dimensional, representing the scalar value of the output of the neuron.

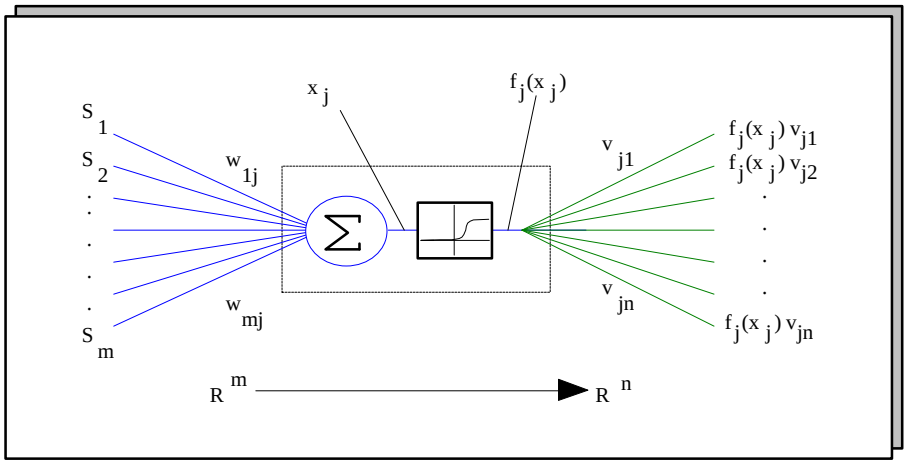


Fig. C1. An artificial neuron

Additive and Shunting Activation Equations

Probably the simplest equations describing the membrane potential of the artificial neuron are the additive activation equations (GROSSBERG 1988). A general form of the additive equation is given by:

$$\frac{dx_j}{dt} = -x_j + \sum_{\forall i} S_i w_{ij} \qquad \text{(C1)}$$

where x_j is the activation of the jth neuron, w_{ij} is the ith component of the input vector of the jth neuron, and S_i is the ith component of the input vector, ***S***. The first term of equation (C1) represents self-excitation which allows for slow, exponential decay of the activation level in the absence of stimulation. The second term represents positive and negative stimulation from other neurons. The second term is nothing more than the dot product between the input vector ***S*** and instar vector ***w***. The more the two vectors point in the same direction, the greater the result. If the

two vectors point in the same direction, the result is unity (for normalized vectors). Conversely, if the two vectors are orthogonal, the result is zero. In effect, ***w*** *filters* ***S***. And since ***w*** is allowed to change over time, ***w*** is said to be an *adaptive filter* (CARPENTER 1989).

The simplest additive activation equation with infinitely leaky activation dynamics is described by:

$$x_j = \sum_{\forall i} S_i w_{ij} \qquad \text{(C2)}$$

The activation x_j is frequently 'squashed' by a continuous sigmoidal output function given by:

$$f_j(x_j) = \frac{1}{1+\exp(-l(x_j - q_j))} \qquad \text{(C3)}$$

which represents the mean firing rate of the neuron. Equation (C3) is a monotonically increasing function of the activation value, x_j, minus a threshold value, Q_j. The function saturates at both ends, thus 'squashing' or bounding the input. This output equation is just one type of squashing function. Other, more hard-limiting functions can result as $\lambda \rightarrow \infty$.

In the additive activation equation (C1), both excitatory and inhibitory inputs are added together to produce a *net* input. This net input affects the membrane potential in an unbounded fashion without automatic gain control. An alternative model, the shunting model, (GROSSBERG, 1968) separates excitatory and inhibitory inputs, bounds their effect according to fixed saturation levels, and adds automatic gain control:

$$\frac{dx_j}{dt} = -Ax_j + (B - x_j)\sum I_{exc} - (C + x_j)\sum I_{inh} \qquad \text{(C4)}$$

where the first term represents self-excitation, the second term represents active excitation from external sources, and the third term represents active inhibition from external sources. Constants *A*, *B*, and *C* represent the passive decay rate, positive saturation level, and negative saturation

level, respectively. Constants B and C bound x_j (for small inputs). The shunting model is closely aligned with the Nobel prize winning membrane potential equation of neurophysiology (HODGKIN & HUXLEY, 1952).

APPENDIX D
CONTINUOUS-VALUED ADAPTIVE RESONANCE THEORY EQUATIONS

The equations used in the simulations were not the actual ART2 equations, but they did have a similar effect. The STM vector $\boldsymbol{s}$ was categorized by calculating the degree to which it excited each neuron in the network and choosing the one with the maximum activation $x_i(t)$. The index of the winning node $i^*(t)$ was chosen according to:

$$i^*(t) = i \quad \text{if} \quad x_i(t)=\max(x_i(t), \forall i) \tag{D1}$$

where the activation $x_i(t)$ was obtained by calculating the dot product between $\boldsymbol{s(t)}$ and each normalized instar vector, $\boldsymbol{w_i(t)}$, as given by:

$$x_i(t) = \boldsymbol{s(t)} \cdot \boldsymbol{w_i(t)} = \sum_{\forall i} s_p(t)w_{ip}(t) = |\boldsymbol{s(t)}||\boldsymbol{w(t)}| \cos \theta_{sw}(t) \tag{D2}$$

where $\theta_{sw}(t)$ represented the angle between vectors $\boldsymbol{s(t)}$ and $\boldsymbol{w_i(t)}$. Notice that since vectors $\boldsymbol{s(t)}$ and $\boldsymbol{w_i(t)}$ were both normalized, the magnitudes, $|\boldsymbol{s(t)}|$ and $|\boldsymbol{w(t)}|$ both equaled unity. Since vector $\boldsymbol{s(t)}$ was constant throughout the competition, equation (D2) depended only on the *direction* of vector $\boldsymbol{w(t)}$ with respect to $\boldsymbol{s(t)}$. When $\theta_{sw}(t)=0$, the result was maximal and when $\theta_{sw}(t)=\pi/2$ radians, the result was minimal, with varying degrees in between. The magnitude of the original STM vector $\boldsymbol{s_u(t)}$ played no part in the competition. As a result, the competition was protected against magnitude biases. In terms of pattern recognition, a high degree of match between the directions of $\boldsymbol{s(t)}$ and $\boldsymbol{w_i(t)}$ constituted a recognition of pattern $\boldsymbol{s(t)}$. Consequently, vector $\boldsymbol{w_i(t)}$ *adaptively filtered* $\boldsymbol{s(t)}$.

After the winning node was chosen, it underwent a vigilance test to determine if $\boldsymbol{s(t)}$ was close enough to $\boldsymbol{w_i(t)}$ for $\boldsymbol{w_i(t)}$ to admit $\boldsymbol{s(t)}$. If the node passed the test then it remained as the winning category and its membership magnitude $m_{i^*(t)}(t)$ was incremented as shown by:

$$i^*(t) = i \text{ if } \max(x_i, \forall i) \geq r(t), \ b \text{ otherwise} \quad \text{(D3)}$$

$$m_{i^*(t)}(t+1) = m_{i^*(t)}(t) + 1 \text{ if } \max(x_i, \forall i) \geq r(t), \ 1 \text{ otherwise} \quad \text{(D4)}$$

The instar vector $\boldsymbol{w_{i^*(t)}(t)}$ of the winning node was then modified by:

$$\boldsymbol{w_{i^*(t)}(t+1)} = \frac{\boldsymbol{s(t)} + \boldsymbol{w_{i^*(t)}(t)} * (m_{i^*(t)}(t+1)-1)}{m_{i^*(t)}(t+1)} \quad \text{(D5)}$$

If the node failed the vigilance test an idle node *b* was recruited to represent the novel pattern:

$$\boldsymbol{w_b(t+1)} = \boldsymbol{s(t)} \quad \text{(D6)}$$

and its membership magnitude $m_b(t)$ was set to unity.

Automatic Vigilance Control

Because ART networks are very sensitive to the vigilance parameter, $r(t)$ was changed into a variable and controlled automatically using the general shunting equation:

$$\frac{dr_j}{dt} = -Ar_j + (B - r_j)\sum I_{exc} - (C + r_j)\sum I_{inh} \quad \text{(D7)}$$

Rewriting (C7) in terms of a difference equation yielded:.

$$r_j(t+1) - r_j(t) = -Ar_j(t) + (B - r_j(t))\sum I_{exc} - (C + r_j(t))\sum I_{inh} \quad \text{(D8)}$$

$$r_j(t+1) = (1 - A)\, r_j(t) + (B - r_j(t))\sum I_{exc} - (C + r_j(t))\sum I_{inh} \quad \text{(D9)}$$

Substituting A=0, B=1, C=0, ΣI_{exc} = .01, and ΣI_{inh} = .05d, where d is a binary input that reflected the event of an STM pattern being forced into the closest ART2 category, yielded the automatic vigilance control equation:

$$r_j(t+1) = r_j(t) + (.01)(1 - r_j(t)) - (.05\delta)\, r_j(t) \qquad \text{(D10)}$$

If $r(t)$ would have not been automatically controlled, the categorization process could have been distorted. For example, if the network would have been too vigilant, it would have used all of its nodes to represent meaningless hemodynamic patterns in the beginning of training, leaving no nodes to represent interesting patterns later. Although provisions were made for reclaiming idle nodes, this would have happened at a relatively slow pace, during which time the salient STM patterns would have been distorted. On the other hand, if the network was not vigilant enough, it would have taken too long to learn and distorted the patterns as well. A happy medium was achieved with equation (D10).

APPENDIX E
STAND-ALONE EQUATIONS OF THE ADAPTRODE-BASED SYNAPSE

The adaptrode synapse, is a dynamic model of the synapse designed to encode the temporal relationship between two stimuli across multiple time scales (MOBUS, 1990). Unlike many simplistic models of the synapse, the adaptrode model is composed of a *set* of potentials which charge up at different rates. In the same way that muscles build up a resistance to atrophy during long periods of use, the adaptrode synapse builds up a 'bank' of potential which can support the synapse during periods of infrequent stimulation. In terms of biological plausibility, the adaptrode synapse models slower biological processes such as long term accumulation of substrates which facilitate short term chemical reactions. The result is that the synapse is protected against transient changes in the environment.

To be exact, the adaptrode synapse is divided into a series of processes as shown in figure E1. The series of processes can be envisioned as a series of coupled capacitors with increasing time constants where the leftmost capacitor charges the quickest and the rightmost capacitor charges the slowest. Each capacitor obeys a shunting equation with *variable saturation levels* modulated by neighboring potentials:

$$z_{ijk}^{(l)}(t+1) = z_{ijk}^{(l)}(t) + g_{ijk}^{(l)}(t)\varepsilon^{(l)}\left[z_{ijk}^{(l-1)}(t) - z_{ijk}^{(l)}(t)\right] - \gamma^{(l)}\left[z_{ijk}^{(l)}(t) - z_{ijk}^{(l+1)}(t)\right] \qquad \text{(E1)}$$

where $z_{ijk}^{(l)}$ represents the lth potential of the kth synapse from the ith neuron to the jth neuron for $1 \le i \le M$, $1 \le j \le M$, $1 \le k \le K$, and $1 \le l \le L$. The quickest capacitor is denoted by $z_{ijk}^{(1)}$, the next quickest is denoted by $z_{ijk}^{(2)}$, and so forth. Each charge affects the rate at which its neighboring capacitors charge and discharge. For example, the second capacitor from the left in figure E1 is charged by the first capacitor while simultaneously being discharged by the third

capacitor. Essentially, the second capacitor is 'pulled up' by the first capacitor and 'pulled down' by the third capacitor. Consequently, the charge on the second capacitor can neither exceed the charge on the first capacitor nor subtend the charge on the third capacitor.

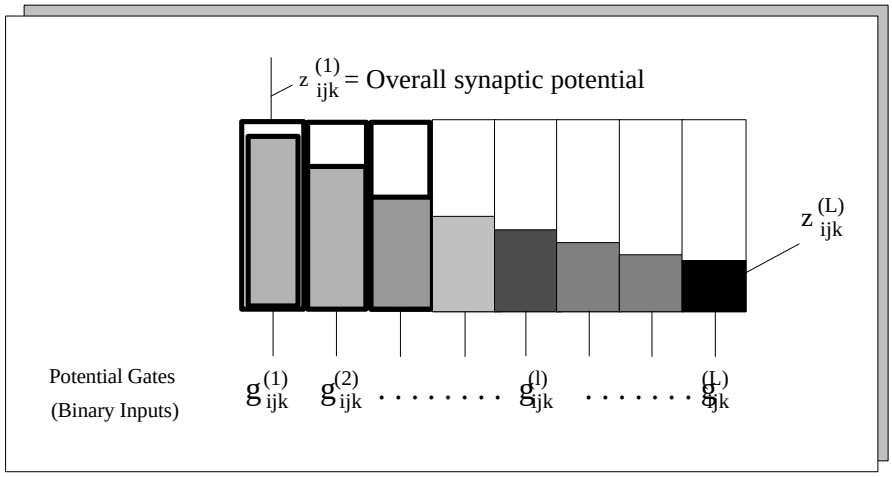


Figure E1. Potentials and gating signals of the adaptrode-based synapse.

Since $z_{ijk}^{(l-1)}(t)$ for l=1 and $z_{ijk}^{(l+1)}$ for l=L do not exist, maximum and minimum saturation constants, are defined as follows:

$$z_{ijk}^{(l-1)}(t) = z^{(max)} \quad \text{if } l=1,\ z_{ijk}^{(l-1)}(t) \text{ otherwise} \tag{E2}$$

$$z_{ijk}^{(l+1)}(t) = z^{(min)} \quad \text{if } l=L,\ z_{ijk}^{(l+1)}(t) \text{ otherwise} \tag{E3}$$

where $z^{(max)}$ and $z^{(min)}$ are usually set to unity and zero, respectively.

The learning of each potential is *gated* by the binary learning signal, or gate, $g_{ijk}^{(l)}(t)$. When $g_{ijk}^{(l)}(t)$ equals unity, $z_{ijk}^{(l)}(t)$ is modulated according to its neighboring potentials. When $g_{ijk}^{(l)}(t)$ equals zero, $z_{ijk}^{(l)}(t)$ experiences passive decay without excitation. The first gate $g_{ijk}^{(1)}$ is intended to reflect the occurrence of a successful event (1 for success and 0 for failure). Usually, $g_{ijk}^{(1)}$ recognizes a successful recognition of a temporal relationship between two events. The gates of the other potentials, $g_{ijk}^{(l)}(t)$, for $2 \le l \le L$, are set according to arbitrary criteria. The

excitatory and inhibitory learning rates are denoted by $\varepsilon^{(l)}$ and $\gamma^{(l)}$, respectively. Each learning rate decreases with increasing l.

Although the adaptrode synapse is made up of many potentials, the first potential is the only one that represents the strength of the synapse as a whole. The other potentials $z_{ijk}^{(l)}$ for $2 \leq l \leq L$, are only used to *calculate* the first potential.

APPENDIX F

SIMULATION SOURCE CODE

```
/* Declare global constants ------------------------------------------------ */

  #include <sys/types.h>
  #include <sys/time.h>
  #include <stdlib.h>
  #include <stdio.h>
  #include <math.h>
  #include <curses.h>
  #include <fcntl.h>

/* Declare global constants ------------------------------------------------ */

  #define R          11        /* number of raw measurements                */
  #define P           7        /* Number of physiological processes         */
  #define A          78        /* max number of clusters per ART2 network   */
  #define K          32        /* Predictive capacity (time units into future)*/

/* Declare global logical variables ---------------------------------------- */

  short Done;                  /* logical program done flag                 */

/* Declare global variables, vectors, and arrays --------------------------- */

  short I[P];                  /* physiological process vector              */
  short p_offset=1;            /* predictive offset index                   */
  short mode[3];               /* program mode vector                       */
  short alarm[A+1];            /* ART2 trend recognition alarms             */
  short winner;                /* ART2 winning category                     */
  short iter_size=3;           /* iteration vector size                     */
  short info_size=2;           /* information vector size                   */
  short alarm_size=A+1;        /* ART2 trend recognition alarm vector size  */
  short t_size=A+1;            /* nodal time stamp vector size              */
  short s_size=P+1;            /* short term memory vector size             */
  long r_size=(R+1)*2;         /* input array size                          */
  long w_size=(P+1)*(A+1);     /* ART2 synapse array size                   */
  long m_size=A+1;             /* ART2 pattern count vector size            */
  long e_size=(A+1)*(A+1)*(K+1);   /* ASTEM event array size                */
  long z_size=(A+1)*(A+1)*(K+1)*5; /* ASTEM synapse array size              */
  long iter[3];                /* iteration numbers                         */
  long t[A+1];                 /* ART2 nodal time stamp                     */
  long m[A+1];                 /* ART2 category magnitude (relative freq)   */
  long e[A+1][A+1][K+1];       /* prediction map event words                */
  long bitcon[K+1];            /* bit constants                             */
```

```
   long histo[20];                /* predictive success histogram vector       */
   long numdone,total_items;      /* misc read/write variables                 */
   float info[2];                 /* misc information                          */
   float r[R+1][2];               /* raw measurement input vector              */
   float s[P+1];            /* short term memory vector                        */
   float w[P+1][A+1];             /* ART2 category vector matrix               */
   float z[A+1][A+1][K+1][5];     /* Adaptive spectral timing/Adaptrode synapses*/
   double vigilance=1;            /* ART2 vigilance parameter                  */
   char te[30];             /* dummy character vector                          */
   FILE *INFILE;                  /* Input data file                           */
   FILE *MSGFILE;                 /* Message file                              */
   FILE *WEIGHTS;                 /* Synaptic weights file                     */

/* Declare global functions ------------------------------------------------ */

   short modulo();                /* modulo function                           */
   short btu();                   /* bipolar to unipolar conversion function   */
   double zero_check();           /* zero checking function                    */
   void recruit_node();           /* recruit ART2 node function                */
   void modify_node();            /* modify ART2 node function                 */
   void jobbar();                 /* job bar update function                   */
   void saveweights();            /* save weights function                     */

/* Declare global windows -------------------------------------------------- */

   WINDOW *MSG;                   /* message window                            */
   WINDOW *JOBBAR;                /* job bar window                            */
   WINDOW *INFOWIN;               /* Information window                        */
   WINDOW *MENU;                  /* Menu window                               */
   WINDOW *HISTOGRAM;             /* ART pattern histogram window              */
   WINDOW *ARTWIN;                /* ART activation window                     */
   WINDOW *STMWIN;                /* Short term memory window                  */
   WINDOW *ASTEM[2];              /* ASTEM window                              */
   WINDOW *ISIDIST;               /* Interstimulus interval distribution       */

/* Main -------------------------------------------------------------------- */

main ()
{/* Declare variables ------------------------------------------------------ */

    register short i,j,k,l,p,x,vp,hp;
    short node[3];

 /* Declare functions ------------------------------------------------------ */

    void kboard();                /* get kboard input function                */
    void read_vector();           /* read input vector function               */
    void ART2();                  /* ART2 function                            */
```

```
  void stm();              /* short term memory update function        */
  void histogram();              /* ART category histogram display function   */
  void astem();                  /* adaptive short term expectation mem functn */

/* Initialize variables ---------------------------------------------------- */

  mode[0]=1;                     /* learning mode or not                  */
  mode[1]=1;                     /* read old weights or not               */
  mode[2]=0;                     /* viewing mode                         */
  total_items=iter_size+r_size+w_size+m_size+alarm_size;
  total_items+=e_size+z_size+info_size+t_size+s_size;

/* Initialize bit constant vectors: bitcon[1]=100...00, bitcon[T]=00...0001 */

  for(k=1;k<=K;k++) {bitcon[k]=(long)pow(2.0,(double)(K-k));}
  bitcon[1]=1; for(k=1;k<=K-1;k++) {bitcon[1]*=2;}

/* Open message file */

  MSGFILE=fopen("avq.log","a");

/* Initialize screen & windows */

  initscr();
  INFOWIN=newwin(1,80,0,0);
  MSG=newwin(3,30,7,24);              box(MSG,0,0);
  JOBBAR=newwin(3,60,14,10);          box(JOBBAR,0,0);
  MENU=newwin(15,40,5,20);            box(MENU,0,0);
  STMWIN=newwin(11,80,12,00);         box(STMWIN,0,0);
  ARTWIN=newwin(11,80,01,00);         box(ARTWIN,0,0);
  HISTOGRAM=newwin(11,80,12,00); box(HISTOGRAM,0,0);
  ASTEM[0]=newwin(11,80,01,00);       box(ASTEM[0],0,0);
  ASTEM[1]=newwin(11,80,12,00);       box(ASTEM[1],0,0);
  ISIDIST=newwin(22,80,01,00);        box(ISIDIST,0,0);

/* Build STMWIN window */

  for(hp=1;hp<7*11;hp++) {mvwaddch(STMWIN,8,hp,ACS_HLINE);}
  for(hp=11;hp<=7*11;hp+=11)
    {for(vp=1;vp<=11;vp++) {mvwaddch(STMWIN,vp,hp,ACS_VLINE);}
     mvwaddch(STMWIN,0,hp,ACS_TTEE);
     mvwaddch(STMWIN,8,hp,ACS_PLUS);
     mvwaddch(STMWIN,10,hp,ACS_BTEE);
    }
  mvwaddch(STMWIN,8,00,ACS_LTEE); mvwaddch(STMWIN,8,7*11,ACS_RTEE);
  sprintf(te,"    IVO   ");          mvwaddstr(STMWIN,9,1,te);
  sprintf(te,"    IVD   ");          mvwaddstr(STMWIN,9,12,te);
  sprintf(te,"    IPP   ");          mvwaddstr(STMWIN,9,23,te);
```

```
   sprintf(te,"Vasocnstrn");       mvwaddstr(STMWIN,9,34,te);
   sprintf(te,"Vasodilatn");       mvwaddstr(STMWIN,9,45,te);
   sprintf(te,"Hypotenson");       mvwaddstr(STMWIN,9,56,te);
   sprintf(te,"Hypertensn");       mvwaddstr(STMWIN,9,67,te);

/* Build MENU window */

   sprintf(te,"MENU                          ");          mvwaddstr(MENU,1,2,te);
   sprintf(te," <1> Trend Recognition & STM  ");          mvwaddstr(MENU,3,2,te);
   sprintf(te," <2> Trend Recognition & Histogram  ");    mvwaddstr(MENU,4,2,te);
   sprintf(te," <3> Trend Prediction (ASTEM)  ");         mvwaddstr(MENU,5,2,te);
   sprintf(te," <4> Alarm Assignments          ");        mvwaddstr(MENU,6,2,te);
   sprintf(te," <i> Information Panel          ");        mvwaddstr(MENU,8,2,te);
   sprintf(te," <m> Menu                       ");        mvwaddstr(MENU,9,2,te);
   sprintf(te," <n> Next                       ");        mvwaddstr(MENU,10,2,te);
   sprintf(te," <s> Save                       ");        mvwaddstr(MENU,11,2,te);
   sprintf(te," <e> Save & Exit                ");        mvwaddstr(MENU,12,2,te);

/* Initialize or read arrays */

   sprintf(te,"      Initializing ...      "); mvwaddstr(MSG,1,1,te);
   touchwin(MSG); wnoutrefresh(MSG); doupdate();
   if(!mode[1])
     {for(i=1;i<=A;i++)
        {for(j=1;j<=A;j++)
           {for(k=1;k<=K;k++)
              {e[i][j][k]=0;
               z[i][j][k][0]=1.0; for(l=1;l<=4;l++) {z[i][j][k][l]=0;}
              }
           }
        }
      iter[0]=1; iter[1]=1; iter[2]=1;                  /* reset iteration counts         */
      info[1]=0; info[0]=zero_check(0.0); /* reset normalization magnitudes */
      for(i=1;i<=A;i++) {alarm[i]=0;}                   /* Reset all alarms               */
     }
   else /* Read weights from binary file ---------------------------------- */
     {sprintf(te,"     Reading Arrays ...     ");
      mvwaddstr(MSG,1,1,te); touchwin(MSG); wnoutrefresh(MSG); doupdate();
      if((WEIGHTS = fopen("/scratch/rgs0497/weights","r+b")) != NULL)
        {numdone=0; total_items=(long)zero_check((double)total_items);
         numdone+=fread((char *)iter,sizeof(long),iter_size,WEIGHTS); jobbar();
         numdone+=fread((char *)info,sizeof(float),info_size,WEIGHTS);jobbar();
         numdone+=fread((char *)r,sizeof(float),r_size,WEIGHTS); jobbar();
         numdone+=fread((char *)w,sizeof(float),w_size,WEIGHTS); jobbar();
         numdone+=fread((char *)m,sizeof(long),m_size,WEIGHTS); jobbar();
         numdone+=fread((char *)e,sizeof(long),e_size,WEIGHTS); jobbar();
         numdone+=fread((char *)z,sizeof(float),z_size,WEIGHTS); jobbar();
         numdone+=fread((char *)t,sizeof(long),t_size,WEIGHTS); jobbar();
```

```
        numdone+=fread((char *)s,sizeof(float),s_size,WEIGHTS); jobbar();
        numdone+=fread((char *)alarm,sizeof(short),alarm_size,WEIGHTS); jobbar();
        fclose(WEIGHTS);
       }
     }

 /* MAIN LOOP ------------------------------------------------------------- */

    Done=0;
    if((INFILE = fopen("input.txt","r+t")) != NULL)
      {read_vector();        /* read first input vector                  */
       do{kboard();                  /* read the keyboard                      */
          for(p=0;p<P;p++)
            {sprintf(te,"%1d",I[p]); mvwaddstr(INFOWIN,0,68+p,te);
            }
          stm();                     /* update STM                             */
          ART2();                    /* instantaneous STM pattern recognition  */
          histogram();               /* histogram of instantaneous STM pattern recs */
          astem();                   /* adapt. spect. timing expectation memory */
          iter[0]++;                 /* increment prediction iteration number  */
          iter[0]=(long)modulo(iter[0],1,K); /* limit iteration number         */
          if(mode[0]) {iter[1]++;}   /* training iteration number              */
          iter[2]++;                 /* absolute iteration number              */
          read_vector();      /* read next input vector                  */
          doupdate();                /* update screen                          */
         }
       while(!feof(INFILE) && !Done);
       fclose(INFILE);
      }

 /* End program ----------------------------------------------------------- */

    saveweights();                  /* save weights to file                   */
    attrset(A_NORMAL);              /* reset video attributes to normal       */
    fclose(MSGFILE);                /* close message file                     */
    endwin();                       /* end windows mode                       */
    exit(1);                        /* exit program                           */
}

/* Zero check function ------------------------------------------------------ */

   double zero_check(value)
   double value;
   {if((double)abs((int)(value*1000))*.001<.001) {value=1;}
    return(value);
   }

/* Bipolar to Unipolar Conversion Function ---------------------------------- */
```

```
   short btu(s)
   short s;
   {return((1+s)/2);
   }

/* Modulo Function ------------------------------------------------------- */

   short modulo(value,min,max)

   short value,min,max;
   {if(value>max) {do {value-=(max-min+1);} while (value>max);}
    if(value<min) {do {value+=(max-min+1);} while (value<min);}
    return(value);
   }

/* Read Input Vector ----------------------------------------------------- */

   void read_vector()

   {register short i,p;
    float HR[2];          /* heart rate                        (BPM)        */
    float MAP[2];         /* mean arterial pressure             (mmHg)       */
    float SAP[2];         /* systolic arterial pressure         (mmHg)       */
    float DAP[2];         /* diastolic arterial pressure        (mmHg)       */
    float MPA[2];         /* mean pulmonary arterial pressure   (mmHg)       */
    float PAS[2];         /* systolic pulmonary arterial pressure (mmHg)     */
    float PAD[2];         /* current diastolic pulm arterial pressure (mmHg) */
    float CVP[2];         /* central venous pressure            (mmHg)       */
    float PP;             /* pulsus paradoxus                   (none)       */
    float RPP;            /* reverse pulsus paradoxus           (none)       */
    float SV[2];          /* stroke volume                      (mL/beat)    */
    float CO[2];          /* current cardiac output             (L/min)      */
    float SVR[2];         /* current systemic vascular resistance (dynes*sec/cm^5) */
    char string[7],*char_result;

    for(i=1;i<=R;i++)
      {r[i][1]=r[i][0];     /* move current to previous */
       char_result=fgets(string,6+1,INFILE);
       r[i][0]=(float)abs((int)(atof(string)/.0001))*.0001;
      }

    if(!feof(INFILE))
     {/* Calculations */

        HR[0]=r[1][0];  HR[1]=r[1][1];          /* heart rate                */
        MAP[0]=r[2][0]; MAP[1]=r[2][1];         /* mean arterial pressure    */
        SAP[0]=r[3][0]; SAP[1]=r[3][1];         /* systolic arterial pressure */
```

```
   DAP[0]=r[4][0]; DAP[1]=r[4][1];          /* diastolic arterial pressure   */
   PAD[0]=r[8][0]; PAD[1]=r[8][1];          /* prev diastolic pulm art press */
   CVP[0]=r[9][0]; CVP[1]=r[9][1];          /* central venous pressure       */
   PP=r[10][0];                             /* pulsus paradoxus              */
   RPP=r[11][0];                            /* reverse pulsus paradoxus      */
   SV[0]=1.4*(SAP[0]-DAP[0]);               /* estimate current stroke vol   */
   SV[1]=1.4*(SAP[1]-DAP[1]);               /* estimate previous stroke vol  */
   CO[0]=HR[0]*SV[0]*.001;                  /* estimate current card output  */
   CO[1]=HR[1]*SV[1]*.001;                  /* estimate previous card output */
   SVR[0]=79.9*(MAP[0]-CVP[0])/CO[0]; /* estimate curr sys vasc resist */
   SVR[1]=79.9*(MAP[1]-CVP[1])/CO[1]; /* estimate prev sys vasc resist */

/* Look for physiologic conditions */

   for(p=0;p<P;p++) {I[p]=0;}         /* reset all conditions          */
   if(PAD[0]>=25 && PAD[0]>PAD[1] && PP==0)
     {I[0]=1;                               /* intravascular volume overload */
     }
   if(PAD[0]<=15 && PAD[0]<PAD[1] && RPP==1)
     {I[1]=1;                               /* intravascular volume depletion */
     }
   if(PAD[0]>=25 && PAD[0]>PAD[1] && PP==1)
     {I[2]=1;                               /* increased pericardial pressure */
     }
   if(SVR[0]>SVR[1] && HR[0]>=90 && CO[0]<4)
     {I[3]=1;                               /* vasoconstriction */
     }
   if(CO[0]>=7 && CO[0]>CO[1] && SVR[0]<SVR[1] && HR[0]>=90)
     {I[4]=1;                               /* vasodilation */
     }
   if(MAP[0]<70 && MAP[0]<MAP[1] && HR[0]>HR[1])
     {I[5]=1;                               /* hypotension */
     }
   if(MAP[0]>100 && MAP[0]>MAP[1])
     {I[6]=1;                               /* hypertension */
     }

/* Print to screen */

   sprintf(te,"Iteration:  %5d",iter[2]); mvwaddstr(INFOWIN,0,2,te);
   touchwin(INFOWIN); wnoutrefresh(INFOWIN);

/* Print to log file */

   fprintf(MSGFILE,"HR %5.1f CO %5.1f MAP %5.1f PAD %5.1f SVR %6.1f PP %1d
RPP %1d
%1d%1d%1d%1d%1d%1d%1d%1d\n",HR[0],CO[0],MAP[0],PAD[0],SVR[0],(short)PP,(sh
ort)RPP,I[0],I[1],I[2],I[3],I[4],I[5],I[6],I[7]);
```

```
    }
  return;
  }

/* STM Update ------------------------------------------------------------ */

  void stm()

  {register short p,x,vp,ht=7;
   long pc;
   double norm_sqr=0;

   for(p=0;p<P;p++)
     {s[p]=.75*s[p]+(1-.75*s[p])*I[p];
      norm_sqr+=pow((double)s[p],2.0);
      for(vp=ht;vp>0;vp--)
        {for(x=0;x<10;x++)
           {if((float)vp>(float)ht-(s[p]*(float)ht)) {pc=ACS_CKBOARD;}
            else {pc=ACS_BULLET;}
            mvwaddch(STMWIN,vp,11*p+x+1,pc);
           }
        }
     }
   if((float)sqrt(norm_sqr)>info[0]) {info[0]=(float)sqrt(1.01*norm_sqr);}
   s[P]=(float)sqrt(pow((double)info[0],2.0)-norm_sqr);
   info[0]=(float)zero_check(info[0]);
   if(mode[2]==0) {touchwin(STMWIN); wnoutrefresh(STMWIN);}
   return;
  }

/* Recruit ART2 node ----------------------------------------------------- */

  void recruit_node(n)

  short n;
  {register short i,j,k,l,p;
   flash();

   for(p=0;p<=P;p++) {w[p][n]=s[p]/info[0];}     /* recruit neuron           */
   m[n]=1; if(m[n]>m[0]) {m[0]=m[n];}            /* reset category magnitude  */
   for(k=1;k<=K;k++)
     {for(i=1;i<=A;i++)                          /* reset all instar vectors  */
        {e[i][n][k]=0;
         z[i][n][k][0]=1.0; for(l=1;l<=4;l++) {z[i][n][k][l]=0;}
        }
      for(j=1;j<=A;j++)                          /* reset all outstar vectors */
        {e[n][j][k]=0;
         z[n][j][k][0]=1.0; for(l=1;l<=4;l++) {z[n][j][k][l]=0;}
```

```
        }
      }
    t[n]=iter[1];
    return;
   }

/* Modify ART2 node ---------------------------------------------------- */

   void modify_node(n)

   short n;
   {register short p;

    for(p=0;p<=P;p++)                          /* synaptic vector learning     */
      {w[p][n]=((s[p]/info[0])+w[p][n]*(float)m[n])/((float)m[n]+1);
      }
    m[n]++; if(m[n]>m[0]) {m[0]=m[n];}         /* increment category magnitude */
    t[n]=iter[1];                              /* time stamp the learning      */

    return;
   }

/* ART2 Pattern recognition, prediction, and learning ---------------------- */

   void ART2()

   {register short a,p,x,vp;
    short min_count,idle_node,ht=8;
    float net[A],max_net=(-999),delta;

    /* calculate neural activation */

      winner=999;
      for(a=1;a<=info[1];a++)
        {net[a]=0;
         for(p=0;p<=P;p++) {net[a]+=(s[p]/info[0])*w[p][a];}
         if(net[a]>max_net) {max_net=net[a]; winner=a;}
        }

    /* synaptic vector learning */

      if(mode[0])
        {if(max_net>=vigilance) {modify_node(winner);}
         else                                  /* look for idle node */
           {if(info[1]<A)                      /* if we still have unrecruited nodes */
              {                                /* recruit previously unrecruited node */
                info[1]++;
                winner=info[1];
```

```
            recruit_node(winner);
          }
         else
          {                                  /* look for idle recruited node */
            min_count=999;
            for(a=1;a<=info[1];a++)
              {if(((double)m[a]/(double)m[0])<1.0/(double)A && (iter[1]-t[a])>500 &&
m[a]<min_count)
                {min_count=m[a];
                 idle_node=a;
                }
              }
            if(min_count<999)              /* if we found an idle node */
              {winner=idle_node;
               recruit_node(winner);
              }
            else                       /* we found no idle node so modify best match */
              {modify_node(winner);
              }
          }
        }
      }

   /* Update screen */

     for(a=1;a<=A;a++)
       {if(a==winner){mvwaddch(ARTWIN,ht+1,a,a+48|A_REVERSE|A_BLINK);}
        else {mvwaddch(ARTWIN,ht+1,a,a+48);}
        if(a>info[1]) {net[a]=0;}
        for(vp=1;vp<=ht;vp++)
          {if((double)vp>(double)ht-(((double)(net[a]-.9)*(double)ht)/.1))
            {mvwaddch(ARTWIN,vp,a,ACS_CKBOARD);
            }
           else
            {mvwaddch(ARTWIN,vp,a,ACS_BULLET);
            }
          }
       }
     if(mode[2]==0 || mode[2]==1) {touchwin(ARTWIN); wnoutrefresh(ARTWIN);}

   /* Automatic vigilance control */

     if(max_net<vigilance) {delta=.05;} else {delta=0;}
     vigilance+=.01*(1-vigilance)-delta*vigilance;
     sprintf(te,"%9.4f %9.4f    ",max_net,vigilance);
     mvwaddstr(INFOWIN,0,25,te);

   /* Sound alarm if set */
```

```
    if(alarm[winner]) {for(x=1;x<=1000;x++) {flash();}}

  return;
 }

/* Update ART2 category magnitude (relative frequency) ---------- */

 void histogram()

 {register short vp,hp;
  short ht=8,width=A;
  double max=(double)m[0];
  for(hp=1;hp<=width;hp++)
    {if(hp==winner){mvwaddch(HISTOGRAM,ht+1,hp,hp+48|A_REVERSE|A_BLINK);}
     else {mvwaddch(HISTOGRAM,ht+1,hp,hp+48);}
     max=zero_check(max);
     for(vp=ht;vp>0;vp--)
       {if((double)vp>(double)ht-(((double)m[hp]*(double)ht)/max))
         {mvwaddch(HISTOGRAM,vp,hp,ACS_CKBOARD);
         }
        else
         {mvwaddch(HISTOGRAM,vp,hp,ACS_BULLET);
         }
       }
    }
  if(mode[2]==1) {touchwin(HISTOGRAM); wnoutrefresh(HISTOGRAM);}
  return;
 }

/* Adaptive Spectral Timing Expectation Memory (ASTEM) ---------- */

 void astem()

 {register short vp,hp,i,j,k,l,x;
  short ht=8,width=A,hopeful;
  static short curr=1,prev;
  long count=0,pc;
  float g,c[4],d[4],sum=0;
  double y[A+1][2],max_hope=(-999);
  double ap[K+1],max_ap_local;
  static double max_net=(-999),max_ap_global=.001;

  /* Initialize Variables */

    c[1]=.10; c[2]=.08; c[3]=.06;
    d[1]=.05; d[2]=.06; d[3]=.04;
```

```
   /* Synaptic reinforcement */

      if(mode[0])
       {if(winner!=999)
         {for(i=1;i<=A;i++)
            {for(j=1;j<=A;j++)
               {for(k=1;k<=K;k++)
                  {g=(float)exp(-.5*(k-1));
                   if(e[i][j][k]&bitcon[iter[0]])
                     {e[i][j][k]^=bitcon[iter[0]]; /* reset bit */
                      if(mode[0])
                        {if(j==winner) {c[1]=.08;} else {c[1]=0;}
                         for(l=1;l<=3;l++)
                            {z[i][j][k][l]+=c[l]*g*(z[i][j][k][l-1]-z[i][j][k][l])-d[l]*(z[i][j][k][l]-
z[i][j][k][l+1]);
                            }
                        }
                     }
                   if(z[i][j][k][1]>=0 && z[i][j][k][1]<=.05)
                     {histo[0]++;
                     }
                   for(x=1;x<=19;x++)
                     {if(z[i][j][k][1]>x*.05 && z[i][j][k][1]<=(x+1)*.05)
                        {histo[x]++;
                        }
                     }
                  }
               }
            }
         }

   /* Let winning node predict future recognitions */

      if(winner!=999)
       {for(j=1;j<=info[1];j++)
          {for(k=1;k<=K;k++)
             {if(j!=winner)
                {e[winner][j][k]|=bitcon[modulo(iter[0]+k,1,K)];
                }
             }
          }
       }

   /* Integrate impulses from neighboring neurons */

      curr*=(-1); prev=curr*(-1);
      for(j=1;j<=A;j++)
```

```
      {y[j][btu(curr)]=0;
       for(i=1;i<=A;i++)
          {for(k=1;k<=K;k++)
             {if(e[i][j][k]&bitcon[modulo(iter[0]+1,1,K)])
                {y[j][btu(curr)]+=z[i][j][k][1];
                }
             }
          }
       if(y[j][btu(curr)]>max_hope)
         {max_hope=y[j][btu(curr)];
          hopeful=j;
         }
       if(y[j][btu(curr)]>max_net) {max_net=y[j][btu(curr)];}
      }
   if(alarm[hopeful]) {for(x=1;x<=1000;x++) {flash();}}

/* Calculate "predictive confidence" */

   for(x=0;x<=19;x++)
      {sum+=(float)(x+1)*.05*(float)histo[x];
       count+=histo[x];
       histo[x]=0;
      }
   if(count!=0)
     {sprintf(te,"%9.7f",sum/(float)count); mvwaddstr(INFOWIN,0,50,te);
     }

/* Update screen */

   max_net=zero_check(max_net);
   for(hp=1;hp<=width;hp++)
      {if(hp==winner) {pc=hp+48|A_REVERSE|A_BLINK;} else {pc=hp+48;}
       mvwaddch(ASTEM[0],ht+1,hp,pc); mvwaddch(ASTEM[1],ht+1,hp,pc);
       for(vp=ht;vp>0;vp--)
          {if((double)vp>(double)ht-((y[hp][btu(curr)]*(double)ht)/max_net))
             {mvwaddch(ASTEM[0],vp,hp,ACS_CKBOARD);
             }
           else {mvwaddch(ASTEM[0],vp,hp,ACS_BULLET);}
           if((double)vp>(double)ht-((y[hp][btu(curr)]*(double)ht)/(max_net*.33)))
             {mvwaddch(ASTEM[1],vp,hp,ACS_CKBOARD);
             }
           else {mvwaddch(ASTEM[1],vp,hp,ACS_BULLET);}
          }
      }

if(mode[2]==2)
  {touchwin(ASTEM[0]); wnoutrefresh(ASTEM[0]);
   touchwin(ASTEM[1]); wnoutrefresh(ASTEM[1]);
```

```
    }

  /* Update interstimulus interval distribution screen */

    if(mode[2]==3)
      {max_ap_global=zero_check(max_ap_global);
       max_ap_local=0; ht=20;
       for(k=1;k<=K;k++)
          {ap[k]=0;
           for(i=1;i<=A;i++) {for(j=1;j<=A;j++) {ap[k]+=z[i][j][k][1];}}
           ap[k]/=(double)(A*A);
           if(ap[k]>max_ap_local) {max_ap_local=ap[k];}
           for(vp=1;vp<=ht;vp++)
             {for(x=1;x<=2;x++)
                {if((double)vp>(double)ht-((ap[k]*(double)ht)/max_ap_global))
                  {mvwaddch(ISIDIST,vp,2*(k-1)+x,ACS_CKBOARD);
                  }
                 else
                  {mvwaddch(ISIDIST,vp,2*(k-1)+x,ACS_BULLET);
                  }
                }
             }
          }
       if(max_ap_local>max_ap_global) {max_ap_global=max_ap_local;}
       touchwin(ISIDIST); wnoutrefresh(ISIDIST);
      }

  return;
 }

/* Job Bar Update Function ---------------------------------------------- */

 void jobbar()
 {short y;
  for(y=1;y<=58;y++)
    {if((double)y<=58.0*(double)numdone/(double)total_items)
      {mvwaddch(JOBBAR,1,y,ACS_CKBOARD);
      }
     else {mvwaddch(JOBBAR,1,y,32);}
    }
  touchwin(JOBBAR); wnoutrefresh(JOBBAR); doupdate();
  return;
 }

/* Poll Keyboard for User Input ------------------------------ */

 void kboard()
```

```
{short a,p,x,cc,vp,ht=7;
 short UserDone=0;
 fd_set readfds;
 struct timeval tv;
 static short once=0;
 if(!once) {FD_ZERO(&readfds); once=1;}
 tv.tv_sec=0;       /* stdin (keyboard) timeout value (seconds)      */
 tv.tv_usec=10;     /* stdin (keyboard) timeout value (microseconds) */
 FD_SET(0,&readfds);
 if((select(1,&readfds,NULL,NULL,&tv))==1)
   {do{switch (cc=getchar())
       {case 27:  switch(cc=getchar())
                  {case 65: if(p_offset<(K-1)) {p_offset++;} break;
                   case 66: if(p_offset>1) {p_offset--;} break;
                   case 91: switch(cc=getchar())
                            {case 65: if(p_offset<(K-1)) {p_offset++;}
                                      break;
                             case 66: if(p_offset>1) {p_offset--;}
                                      break;
                            }
                  }
                 UserDone=1;
                 break;
        case 49:  mode[2]=0; UserDone=1; break;
        case 50:  mode[2]=1; UserDone=1; break;
        case 51:  mode[2]=2; UserDone=1; break;
        case 52:  mode[2]=3; UserDone=1; break;
        case 53:  for(a=1;a<=A;a++)
                   {sprintf(te," Pattern:     <s>et <r>eset ");
                    mvwaddstr(MSG,1,1,te);
                    mvwaddch(MSG,1,11,a+48|A_REVERSE);
                    if(alarm[a])
                      {mvwaddch(MSG,1,16,115|A_REVERSE);
                       mvwaddch(MSG,1,22,114);
                      }
                    else
                      {mvwaddch(MSG,1,16,115);
                       mvwaddch(MSG,1,22,114|A_REVERSE);
                      }
                    touchwin(MSG); wnoutrefresh(MSG);
                    for(p=0;p<P;p++)
                       {for(vp=ht;vp>0;vp--)
                          {for(x=0;x<10;x++)
                             {if((float)vp>(float)ht-(w[p][a]*(float)ht))
                                {mvwaddch(STMWIN,vp,11*p+x+1,ACS_CKBOARD);
                                }
                              else
                                {mvwaddch(STMWIN,vp,11*p+x+1,ACS_BULLET);
```

```
                        }
                      }
                    }
                  }
                 touchwin(STMWIN); wnoutrefresh(STMWIN); doupdate();
                 switch (cc=getchar())
                   {case 110: break;
                    case 114: alarm[a]=0; break; /* reset alarm */
                    case 115: alarm[a]=1; break; /* set alarm */
                   }
                 }
               UserDone=1;
               break;
          case 101: UserDone=1; Done=1; break;
          case 105: touchwin(INFOWIN); wrefresh(INFOWIN); break;
          case 109: touchwin(MENU); wrefresh(MENU); break;
          case 110: UserDone=1; break;
          case 115: UserDone=1; saveweights(); break;
          default:  Done=0;
                    UserDone=1;
         }
         FD_SET(0,&readfds);
        }
      while (!UserDone || select(1,&readfds,NULL,NULL,&tv)==1);
     }
    return;
   }

/* Save Weights To DOS File ---------------------------------- */

   void saveweights()

   {sprintf(te,"         Saving ...         "); mvwaddstr(MSG,1,1,te);
    touchwin(MSG); wnoutrefresh(MSG); doupdate();
    if((WEIGHTS = fopen("/scratch/rgs0497/weights","w+b")) != NULL)
      {numdone=0; total_items=(long)zero_check((double)total_items);
       numdone+=fwrite((char *)iter,sizeof(long),iter_size,WEIGHTS); jobbar();
       numdone+=fwrite((char *)info,sizeof(float),info_size,WEIGHTS); jobbar();
       numdone+=fwrite((char *)r,sizeof(float),r_size,WEIGHTS); jobbar();
       numdone+=fwrite((char *)w,sizeof(float),w_size,WEIGHTS); jobbar();
       numdone+=fwrite((char *)m,sizeof(long),m_size,WEIGHTS); jobbar();
       numdone+=fwrite((char *)e,sizeof(long),e_size,WEIGHTS); jobbar();
       numdone+=fwrite((char *)z,sizeof(float),z_size,WEIGHTS); jobbar();
       numdone+=fwrite((char *)t,sizeof(long),t_size,WEIGHTS); jobbar();
       numdone+=fwrite((char *)s,sizeof(float),s_size,WEIGHTS); jobbar();
       numdone+=fwrite((char *)alarm,sizeof(short),alarm_size,WEIGHTS); jobbar();
      }
    fclose(WEIGHTS);
```

```
 return;
}
```

VITA

(August 1994)

In 1986 Ronald Glen Spencer entered GMI Engineering and Management Institute with sponsorship from the General Motors Fairfax assembly plant in Kansas City, KS. For the four and a half years he alternated between school and work, spending twelve weeks in each place. At school he studied electrical engineering and at the plant he worked in several departments doing a variety of technical jobs including designing and implementing large-scale management information systems (MIS), computer aided design (CAD), troubleshooting manufacturing automation and computer systems, programming logic controllers, computer programming in 'C' and UNIX, and simulating product mixing algorithms.

In June of 1991 Ron earned a Bachelor of Science in Electrical Engineering with a minor in Business Management from GMI. At the same time Ron was hired as an associate systems engineer at the Fairfax plant. He continued his undergraduate thesis work (stochastic throughput bottleneck identification) and continued to learned about intelligent machines in the plant. Over the next several years he became interested in natural intelligence and neural networks.

In January of 1993 Ron entered the graduate Bioengineering program at Texas A&M University to study bioengineering and neuroscience. With this thesis Ron will complete a Master of Science in Bioengineering in August 1994.

Ron's current major field of study is cognitive and neural systems. He plans to study VLSI Neural Systems in the Electrical Engineering department at Texas A&M. Specific interests include biological signal and information processing, brain modeling, machine intelligence, speech recognition, visual processing, chaos theory, and neurological disorders.