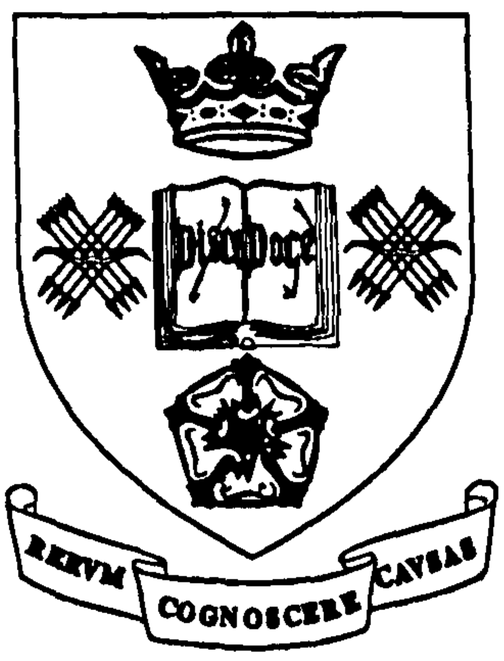
Department of
AUTOMATIC
CONTROL
and SYSTEMS
ENGINEERING
UNIVERSITY of SHEFFIELD

The Application of Adaptive Resonance Theory and Reinforcement Learning To Mapping and Control

Shaun Marriott, M.Sc.(Tech), B.A.(Hons.), Cert. Ed.



Department of Automatic Control and Systems Engineering
The University of Sheffield

Submitted in partial fulfilment of the requirements for admission to the degree of Ph.D.

September 1996

Acknowledgements

I dedicate this thesis to my grandparents Charles and Clarice Grattage without whose patience and love it would not have existed.

I would very much like to thank my supervisor, Robert Harrison, for his support and advice. His keen insight and thought-provoking questions have contributed greatly to the development of this work; his unerring sense of perspective has proved invaluable.

I would also like to thank my colleagues Chee Peng Lim and Zhe Ma for the many stimulating discussions which often led to new insights and clarifications.

I also acknowledge the support of Charles Whitworth whose friendship and sense of humour have contributed indirectly to this work.

My thanks go to the Engineering and Physical Sciences Research Council who have supported me financially throughout this work.

A word of mention must go to Mrs F Elliott for her encouragement and unwavering optimism.

Finally, I must mention that my early interest in reinforcement learning was increased through conversations with Richard 'Richie' Sutton at the Cambridge Neural Network Summer School.

The Application of Adaptive Resonance Theory and Reinforcement Learning To Mapping and Control

Shaun Marriott, M.Sc.(Tech), B.A.(Hons.), Cert. Ed.

Summary

In this thesis, the ideas of Adaptive Resonance Theory (ART) and Reinforcement Learning (RL) are applied to the problems of mapping and control. A neural architecture, fuzzy ARTMAP is considered as an alternative to standard feedforward networks for noisy mapping tasks. It is one of a series of architectures based upon ART. Fuzzy ARTMAP has advantages over feedforward networks—such as increased autonomy—and is especially suited to classification-type problems. Here it is used to estimate a continuous mapping from noisy data. Results show that properties useful for classification problems are not necessarily advantageous for noisy mapping problems. One particular feature is found to cause specialisation to the data. A modified variant is proposed which stores probability information in a sub-unit of the architecture. The proposed fuzzy ARTMAP variant is found to outperform fuzzy ARTMAP in a *mapping* task.

Another novel self-organising architecture, loosely based upon a particular implementation of ART, is proposed here as an alternative to the fixed state-space decoder in a seminal implementation of reinforcement learning. A well-known non-linear control problem is considered. Input / output pattern pairs, desired state-space regions and the network size / topology are not known in advance. Results show that, although learning is not smooth, the novel ART-based RL implementation is successful and develops a meaningful control mapping. The new decoder increases its information capacity as necessary and indicates that such a self-organising approach to control is viable. The self-organising properties of the new decoder allow the neurocontroller to retain previously learned information and to adapt to newly encountered states throughout its operation, on-line.

A fuzzy version of the original RL implementation is implemented to investigate the possibility of distributing control information across more than one state-space region. The fuzzy version is found to outperform the original RL implementation in a *control* task..

Contents

Chapter 5 Extending the Hybrid

.

Chapter 6 General Discussion and Conclusions

"...even to animals eventually capable of speech such as ourselves, the world is initially an unlabeled place."—Gerald M. Edelman, 1989

Chapter 1. Neural Networks, Mapping and Control

1.1 Motivation and Overview

One particular area of research of note in the technological arena lies in the development of more intelligent and autonomous systems. Although caution must be exercised when using the word 'intelligent', it is not difficult to grasp the intended meaning. There is a growing need for autonomous systems which act upon their environment with less pre-programmed rigidity and reliance on human intervention than many existing control solutions.

The work presented in this thesis is an investigation into two key aspects of machine intelligence which are intimately related viz. *mapping* and *control*. These are examined in some detail through both the study of existing artificial neural architectures and the introduction of three novel architectures. The new architectures do not lay claim to being universally applicable systems which circumvent all problems. Indeed, it is doubtful that such a universal system exists owing to the fundamental nature of the group of competing constraints involved in intelligent information processing; many of the constraints are mutually exclusive and compromise is the best that can be hoped for. The development of the novel architectures detailed in this thesis illustrates many of the issues involved in quantifying and implementing intelligent computing and control strategies.

The two main biologically-inspired areas of research covered in this thesis are *Adaptive Resonance Theory* (ART) and *reinforcement learning* (RL); both areas show future promise and interest in them is increasing. One of the new

architectures is a hybrid system with features taken from both theories and combined to give an autonomous self-organising control system.

The self-organising capabilities of ART-based architectures combine naturally with the reduced supervisory requirements of reinforcement learning systems. One of the motivating factors of this work is to investigate the feasibility of self-organising reduced-supervision systems for intelligent control.

The subject matter of this thesis falls naturally under the two headings of mapping and control. Adaptive resonance theory forms the basis for the new architectures developed in both areas of investigation. Reinforcement learning (and related areas) is of relevance only for the area of control. These factors make it more convenient to introduce background material as appropriate throughout the thesis. The structure divides the material conveniently into sets of related topics grouped within chapters. This is to prevent an oversized introduction consisting of a large number of preliminary topics grouped together out of context.

Chapter 1 provides a brief survey of neural networks, mapping and control. This survey of relevant concepts and architectures comprises a backdrop for the thesis and motivates the development of the new architectures.

The first sections of Chapter 2 introduce Adaptive Resonance Theory and develop key themes in considerable detail before introducing a novel ART architecture called *PROBART*. PROBART is applied to mapping tasks and its performance is evaluated. Adaptive resonance theory is the common theme running throughout the thesis and provides a basis for all of the new architectures.

Chapter 3 is concerned with the reinforcement learning method. Reinforcement learning is discussed in the context of psychology and learning theory and forms the basis for the discussion of selected neurocontroller architectures. The seminal implementation of Barto, Sutton and Anderson (BSA) is then discussed in considerable detail and motivates the development of the second novel

architecture. Related ideas grouped together within this chapter include classical and operant conditioning, automata theory, and temporal difference learning.

Chapter 4 motivates the development of an ART-based self-organising architecture. The new architecture, called EUCART, forms a component of the EUCART-BSA hybrid neurocontroller. The hybrid neurocontroller is described in detail and its performance is evaluated with a number of simulations.

Chapter 5 investigates extensions to the ideas and architectures covered in the previous three chapters and introduces the area of fuzzy logic. A third novel architecture is described, called *FUZBOX* which illustrates some of these extensions.

Finally, Chapter 6 provides a general discussion and review of ideas covered in this thesis, draws some conclusions and indicates possible directions for further research.

1.2 Adaptive Behaviour

Humans and animals are able to adapt to changing conditions in the world around them. Successful adaptation is indicated by survival and by avoidance of discomfort. The main features of this adaptive behaviour are the prediction of certain environmental characteristics and the selection of appropriate actions from a repertoire including avoidance or control strategies. In general, learning is directed by reward and punishment stimuli acquired from an environment.

Prediction of temporal or spatial characteristics of an environment by an organism is a form of *system identification* (e.g. Norton, 1986; Söderström and Stoica, 1989); system identification—of whatever degree of sophistication—is a prerequisite for appropriate action if an organism is to adapt successfully. In living organisms, system identification often involves the formation of cognitive

maps (Walker, 1975) which represent pertinent information about the operating environment. Cognitive maps, or schemata (Howard, 1987) are selective abstractions of environmental features which allow behavioural adaptation (learning) by an organism or intelligent agent faced with a potentially confusing array of stimuli.

The simplest case of input-output map which does not require cognition is the stimulus-response map of *classical conditioning* (e.g. Barker, 1994). However, this is usually only appropriate in straightforward environmental situations involving a limited behavioural repertoire. In general terms identification may be applied to the environment and an appropriate action selected (indirect control) or by developing directly a control strategy (direct control). Either way, an internal representation of selected characteristics of the environment is acquired through time from the mass of available information.

The foregoing discussion may appear self-evident when thinking of living organisms. After all, these activities are carried out on a daily basis in the struggle for survival and, as such, proceed without reflection. It may appear to be a gross oversimplification of behaviour but even this analysis reveals several key points.

When attempting to develop artificial autonomous agents, certain important concepts become apparent and it transpires that the process of adaptation cannot be taken for granted after all. What is involved in the formation of an internal representation of the environment or a successful control strategy? How can information about the world be represented stored and retrieved? How is such information to be used? What is a "successful" control strategy? What does control involve? How can artificial autonomous agents be developed which will behave appropriately? Indeed, what is appropriate behaviour?

These questions and many more have arisen through the study of human and animal behaviour (e.g. Best, 1992; Carlson, 1994; Pinel,1993 ; Gellatly, 1986). This thesis will consider such issues from the point of view of *artificial*

intelligence (AI) with particular emphasis on the areas of *artificial neural systems* or *neural networks* and *autonomous learning systems*.

1.3 Mapping and Control: Interrelations

There are two major themes underlying adaptive behaviour which are implicit in the above discussion; these are mapping and control. These two themes are intimately related in that the mapping between an agent's actions and the subsequent environmental responses gives valuable information required to modify the action (behavioural) repertoire. In a limited sense, the behaviour of living organisms can be thought of as sequences of control actions (even actions involving flight or avoidance). Figure 1.1. shows a simple control (behaviour) loop involving mappings between environmental inputs and outputs and organism inputs and outputs; it illustrates schematically the relationship between an intelligent agent and its environment.

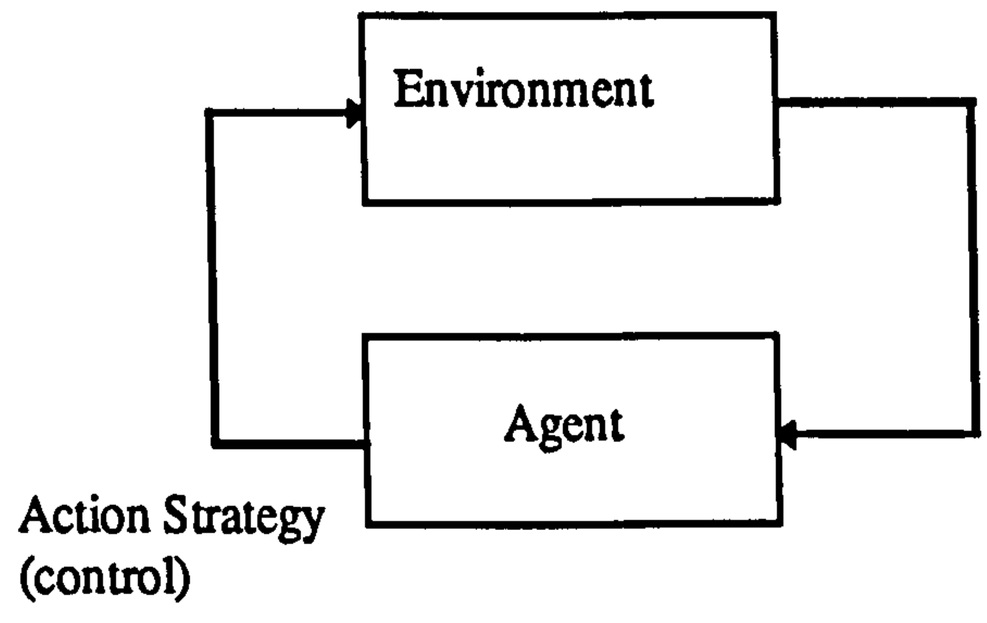


Figure 1.1. The relationship between an intelligent agent and its environment. The agent responds to information from the environment with an action or set of actions from its behavioural repertoire depending upon past experiences.

The biological inspiration giving rise to the two themes of mapping and control also provides inspiration for an implementable system suited to such intelligent tasks. To be more specific, this chapter will consider biologically inspired neural networks capable of implementing mappings and control strategies. Section 1.4 will introduce the idea of artificial neural networks and provide a background so that specific neural network architectures can be introduced in subsequent

sections. By considering these established architectures, this route will lead to the idea of artificial autonomous agents with much more flexibility.

1.4 Artificial Neural Networks

Evolution has resulted in the development of a highly adaptive command and control system in the human body. The *central nervous system* (CNS) consists of the brain and spinal chord. Communication with the rest of the body is via the *peripheral nervous system* (PNS). The main centre for information processing is the brain which consists of two main types of cells: *neurons* which form the excitable tissues, and *neuroglia*, which carry out a large number of important structural and maintenance functions. It is the neurons which provide biological inspiration for computing elements.

Biological neurons are modelled using simplified abstractions of key features to give artificial counterparts (e.g. Levine, 1991). An *artificial* neuron capable of learning, called an adaline, is discussed in Section 1.6. Artificial neural systems, or networks, constructed from comparatively simple elements form the subject of this thesis.

The field of *artificial neural networks* (ANNs) is also known as *connectionism*, *parallel distributed processing* (PDP), *neurocomputing* and *artificial neural systems* (ANS) (McClelland and Rumelhart, 1986; Simpson, 1990). A related field, *computational neuroscience* (Churchland and Sejnowski, 1992) encompasses both artificial and biological neural networks. Henceforth, the term "neural networks" will be used for convenience and will refer to biologically inspired artificial neural systems.

A relative newcomer in the history of science, neural networks comprises a highly interdisciplinary field devoted to developing new ways of processing information. By combining abstract processing elements, modelled on biological neurons, *emergent properties* arise from simulations of neural networks which emulate

some aspects of their biological counterparts. "Emergent properties" is a term used to describe the behaviour of systems whose individual subsystems may themselves have a very simple description, yet when taken together as a whole, can exhibit complex behaviour.

In broad terms a neural network learns to represent a subset of salient features of the environment. A set of appropriate or desirable responses is learned which can be elicited by the relevant environmental cues. The characteristic of generalisation is also present in many artificial neural systems which allows meaningful responses to stimuli not previously encountered. Good generalisation occurs when adequate responses are made to inputs of a given class when only a few exemplars or instantiations of that class have previously been processed. Generalisation is a consequence of the distributed representation of some artificial neural architectures.

Neural networks, within the limits of their structure, modify stored information through experience in contrast to systems which are pre-programmed with all information likely to be used during their operational lifetime. Performance on some task is improved with respect to some prescribed measure. Haykin defines neural network learning as:

> "...a process by which the free parameters of a neural network are adapted through a continuing process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place." (Haykin, 1994).

1.5 A Black Box Approach

For the present, neural networks can be treated as 'black boxes'. Any neural network can be viewed as a collection of input and output variables by the end user who, in many cases, is not concerned with the—possibly complex—internal structures or processes (Figure 1.2.). Like human and animal learning viewed

from a simplistic standpoint, data is presented to the neural network and is subsequently processed to produce information that is reflected in a change of internal *state* or external behaviour. Why is this any different from conventional data processing? Viewed at this superficial level, it appears that any computer program following a set of instructions, or *algorithm,* can be classed as a neural network; there is some truth in this, especially at the level of computer hardware, but there are fundamental differences in neural network and algorithmic information processing as will become apparent.

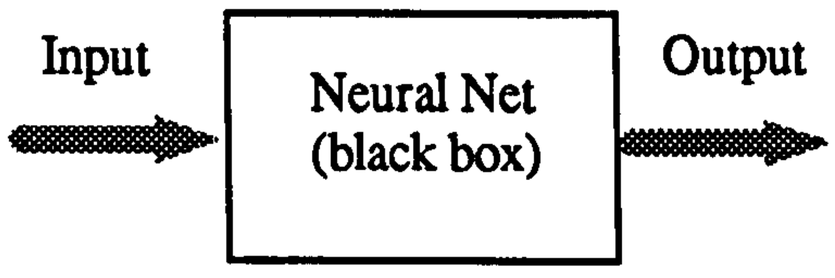


Figure 1.2. A neural net as a black box mapping

The key attribute of neural networks is their ability to learn from experience. Learning enables neural architectures to solve mapping and classification problems by adjusting a representation of the problem until a desirable solution has been found.

How data is used by an artificial network depends upon the learning method used. Within the field of artificial neural systems three broad classes of learning method can be distinguished: *supervised, unsupervised* and *reinforcement* learning (e.g. Haykin, 1994) The concepts associated with all three learning methods will recur throughout this thesis.

- *Supervised learning*: This form of learning involves pairs of patterns to be associated by a neural network; the pattern pairs, consisting of an input and a desired output, are pre-specified by an external teacher (Figure 1.3.). The set of pattern pairs is presented to the network until desired learning criteria are fulfilled. Note that the data have already been pre-processed by the user who has previously decided what data is relevant to the problem domain and which

patterns are to be associated; object classes are defined *a priori* to reflect a pre-defined structuring of the problem domain.

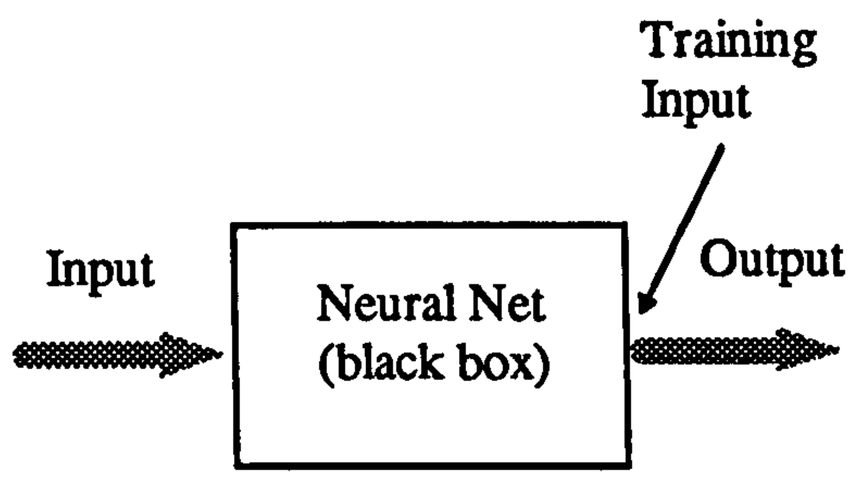


Figure 1.3 Supervised learning (training) of a neural net.

- *Unsupervised learning* (self-organisation): Here there is no external teacher and, thus, no pre-specified organisation of the domain (Figure 1.4.). The neural network has autonomously to organise the input data into structures and to find regularities within the input space. Any *a priori* information is usually in the form of constraints governing the similarity or "closeness" of data items. Self-organising systems, through experience, develop an internal structure that reflects the ordering of information in the environment.

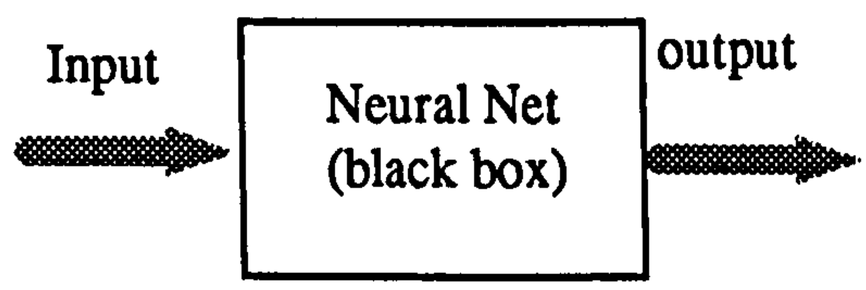


Figure 1.4 Unsupervised learning in a neural net. Self-organised categories are taken to be the outcome of the learning process.

- *Reinforcement Learning*: this learning method has evolved from consideration of aspects of psychology. A goal or "end state" is specified but no direction or method of attaining the goal is given. Learning is by trial and error with successful changes being rewarded by a non-specific reward signal; similarly, unsuccessful changes are penalised. The non-specific signal is generated by a *critic* network that only indicates success or failure and not the direction of

future changes (Figure 1.5.). Unlike the supervised learning method, the learning task is not solved beforehand by a teacher and pattern pairs are not specified, only the end-state is specified together with a punishment schedule. Also, reinforcement learning is distinct from unsupervised learning owing to the adaptive critic which analyses the performance of the system with respect to a goal or task. Unsupervised systems do not have feedback of this type as there is no teacher or critic. Reinforcement learning can be thought of as supervised learning but not in the strictest sense because the crude reinforcement signal does not give the desired output but only a crude measure of success or failure.

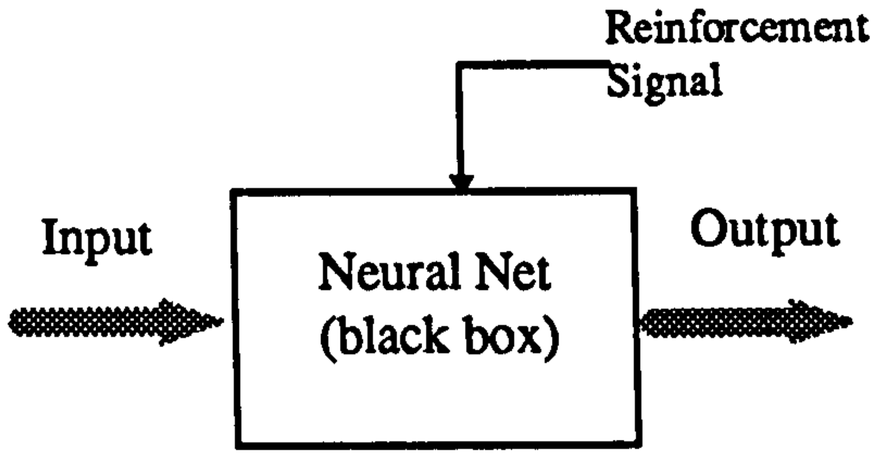


Figure 1.5. Reinforcement learning applied to a neural net.

1.6 Neural Network Architectures: a Critical Review

One of the earliest useful neural network architectures is the *adaline* or *AD*Aptive *LIN*ear *E*lement(Widrow and Hoff, 1960; Widrow and Smith, 1963; Widrow, 1987). The adaline is a binary classifier which implements linear discriminant analysis of decision theory. It is capable of learning from experience and converges incrementally to a hyperplanar decision boundary through the data. The final solution gives the best (least squares) approximation to a suitable (possibly non-linear) decision boundary.

Adaline-type elements have been applied to weather forecasting, speech recognition, cardiogram analysis and also to control engineering where an adaline was used to model an existing controller for the cart-pole problem discussed in section 3.3.2 (Widrow and Smith, 1963; Widrow, 1987);

The performance of the adaline is measured by a function of the adjustable parameters, $\mathbf{w} = (w_0, w_1, \ldots w_m)$ known as *weights*. The function, denoted here by $E(\mathbf{w})$, defines an *error energy surface* parameterised by the weight vector. The optimal solution is represented by the set of weights for which $E(\mathbf{w})$ attains the global minimum of the error surface which is found by adapting the weights and is analogous to learning.

The task of modifying the adaline weight vector is called *training*. A training file is used which contains a set of pattern pairs, $\{\mathbf{x}_p, d_p\}$ consisting of an input pattern and its desired class output. This training file is read sequentially for as many times as are necessary during the supervised learning procedure.

The weight-update rule for adapting the adaline weights is the *Widrow-Hoff* learning rule which implements the *gradient descent* procedure in which the weight vector is moved in the direction of steepest descent in the error energy space.

The general form of the gradient descent approach is given in vector form by

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla_{\mathbf{w}} E \qquad (1.1)$$

In terms of a discrete time interval, t, where η is a constant governing the learning rate and . $\nabla_{\mathbf{w}} E$ is the gradient of the error energy surface with respect to the weights.

One thing to note is that the adaline will always find a solution; the linearity restriction for the adaline simply means that the best (LMS) linear solution will be found. As far as non-linear decision boundaries are concerned, however, all is not lost. If the input data can be transformed from the original space into a space where the classes are more readily linearly separable, then a solution can be found using the adaline; the technique of transforming the data into a new space is known as *pre-processing* which involves transforming the desired input-output

mapping into a form which is linear-in-the-parameters. Pre-processing is required for the solution of non-linear mapping or decision problems by a linear network.

The main problem with using pre-processing is that the non-linear transform must be known *a priori*. In a few simple cases where data can be visualised easily, it may be possible to assume a transform. In most cases, the data is complex and a suitable transform is not identifiable readily. However, the need for pre-processing may be circumvented by building non-linearity into a network.

The importance of pre-processing will be seen in Chapters 3, 4 and 5 where methods of pre-processing (state-space) are considered as part of a control problem; automated methods to render problems more tractable are highly desirable.

Two possible ways of constructing non-linear neural networks are:

(i) to use a set of non-linear basis functions to construct a mapping or decision boundary which is linear-in-the-parameters or,

(ii) to use a layered *feedforward* network of fixed dimensions consisting of cascaded non-linearities—non-linear-in-the-parameters.

Examples of (i) include the *functional link network* (Pao, 1989) and the *radial basis function network* (Powell, 1987; Broomhead and Lowe, 1988; Moody and Darken, 1989; Girosi and Poggio, 1990) which may be constructed incrementally. An example of (ii) is the *multilayer Perceptron* (Rumelhart, Hinton and Williams, 1986).

The adaline can be used to construct multilayer networks. However, multilayer networks of these elements are not necessarily useful. For the adaline, a multilayer network has a single layer equivalent and is thus restricted to

implement a *linear* function. If the adaline is used in multiple layers with its post-processing step function—as was originally conceived (Widrow and Hoff, 1960)—then major difficulties arise in the development of adaptation algorithms. These have been addressed by Widrow (Widrow and Hoff, 1960; Widrow and Lehr, 1990) but such methods have not been widely adopted.

In general, a multilayer network which uses gradient descent has the following requirements:

- a *non-linear* activation function to prevent the single layer equivalence problem,

- a method of *credit assignment* to distribute the error at the output throughout the network, and

- a *differentiable* activation function to enable gradient-based adaptation laws to be used.

If non-linear continuous processing units are used for the nodes comprising the multilayer network, the result is a multilayer Perceptron (MLP)(Rumelhart, Hinton and Williams, 1986). The use of non-linear sigmoidal (usually logistic) activation functions precludes the existence of an equivalent single layer network and thus implements a non-linear mapping. Logical questions now arise regarding the form of a possible learning law for such a network and what can be approximated by it.

The learning law is a generalisation of the gradient descent rule of equation (1.1). The continuous activation function allows errors in the output layer to be *back-propagated* to the input layer so that *all* the network weights can be updated.

Subject to some conditions, arbitrary functions can be approximated using a polynomial basis (Scarborough, 1966; Timan, 1994). That is, component polynomials can be used to construct approximations to a given function. Here, the MLP uses sigmoid functions. A result by G. Cybenko (Cybenko, 1989) shows

that, under some mild conditions, any continuous function of N real variables may be approximated by an MLP of a single hidden layer (layer between input and output layers). Other, independent, results concur with this result (Hornik *et al*, 1989, Funahashi, 1989) and new results have extended the original scope (e.g. Hornik, 1993). These theorems, however, give little, if any, practical guidance as to the network size and configuration for a given mapping task; they are existence theorems, and some networks may have to be of impractical dimensions to achieve the desired approximation. In the MLP the pre-processing is an intrinsic property of the architecture where the hidden layer(s) distort the incoming signal through "squashing" by the sigmoidal activation functions.

The adaline is restricted to linear solutions which can be ascertained *a priori* (e.g. Kohonen, 1989; Haykin, 1994) using the techniques of linear algebra. This obviates the use of a linear neural network.

Although, the MLP has good representational properties, optimisation is non-linear and non-convex leading to potential difficulties in training. When using supervised learning to train a network to represent a non-linear mapping between input and output space, the problem of *local minima* may be encountered (e.g. Gallant, 1993; Haykin, 1994). For example, as with a single layer non-linear network the MLP error energy function has the possibility of many local (false) minima. Figure 1.6 illustrates that following the steepest descent rule does not always result in the minimum error energy.

A possible result of encountering a local minimum, in terms of pattern classification, is shown in Figure 1.7.

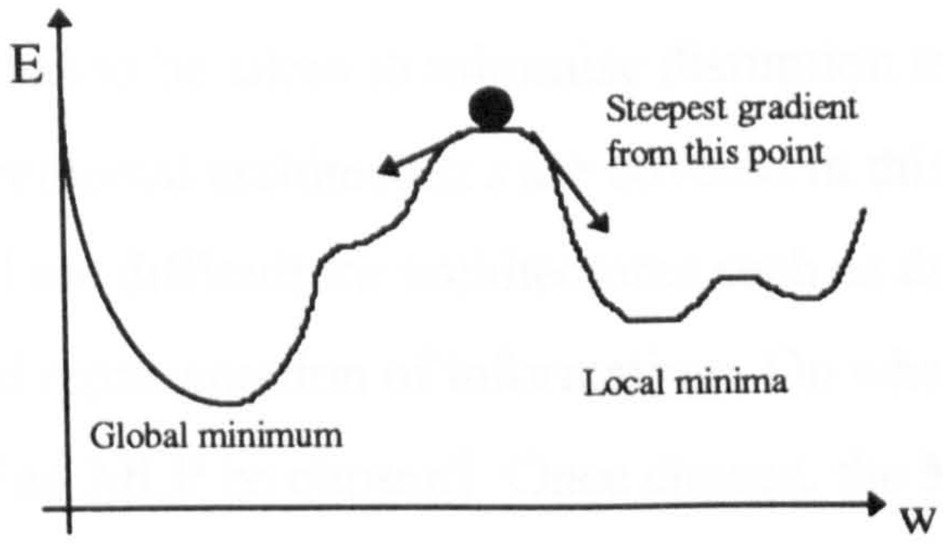


Figure 1.6. The problem of local minima.

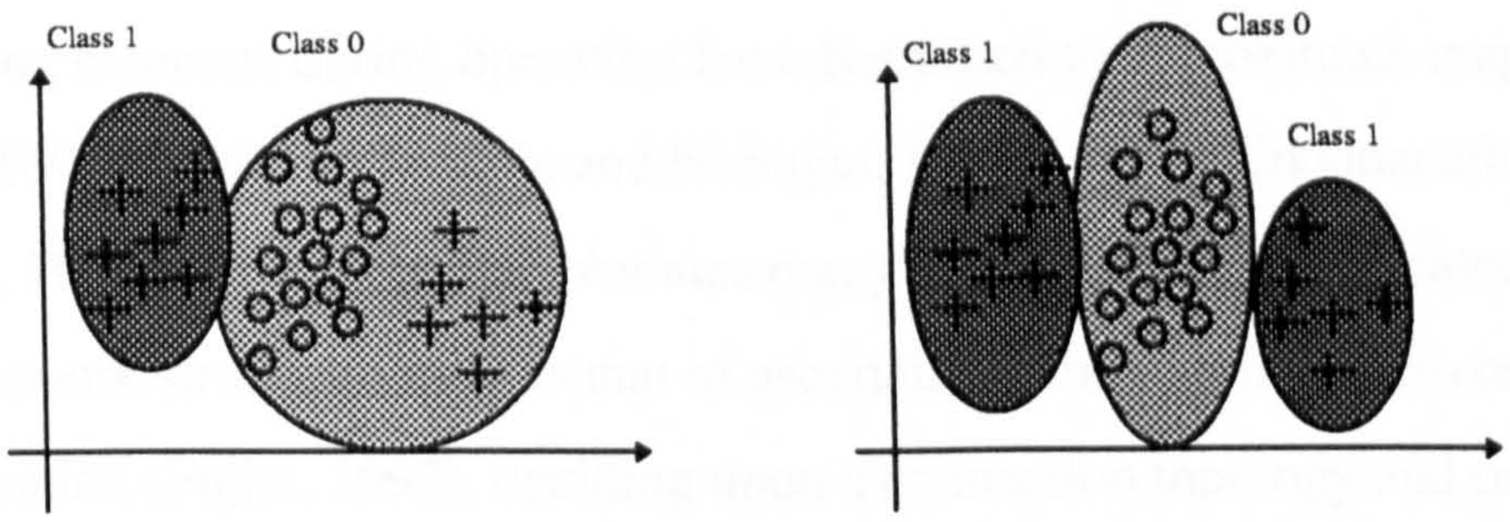


Figure 1.7. The effect of local minima on classification. (a) A local minimum leads to the misclassification of a group of patterns. (b) A possible MLP classification scheme without the local minimum of (a).

In practical terms, the higher the dimension of weight space, the more chance there is of escaping from a local minimum by following one of the weight space dimensions. The location and depth of local minima depend upon the number of layers and the number of nodes in each layer. The choice of network size and configuration for optimal learning is still a large area for research and depends mainly on heuristic (trial and error) methods but see Vapnik (1995) for optimal structure selection in feedforward networks for pattern classification.

The last point raises another issue; that of fixed vs. free network topologies. For some networks the topology (size and configuration), and hence the information capacity, is fixed and so may not provide the best representation for a particular data set. For a fixed topology, the network parameters are adjusted to optimise the representation of a given data set. If the fixed topology is inadequate then, despite much training, the representation may always remain poor.

Variable topologies have the advantage of flexibility in that nodes may be added or removed but care has to be taken to minimise disruption to the representation. Incremental and decremental architectures are covered in this thesis. Node addition and removal are difficult for architectures such as the MLP owing to the non-linear distributed representation of information. On what basis can the size and configuration of an MLP be chosen? Once chosen, the MLP architecture is usually fixed *a priori* regardless of the training set.

Artificial neural networks which learn incrementally by adding new nodes or processing elements during operation have been used to approximate mappings (Platt, 1991; Kandirkamanathan and Niranjan, 1992; Liu, Kadirkamanathan, and Billings, 1994). This technique obviates many of the problems associated with fixed network structures such as that of ascertaining the optimum network size / configuration (Fujita, 1992), deciding upon a connection topology and providing sufficient information capacity (complexity) for adequate representation of the problem domain.

Incremental learning is especially useful in situations where information is gathered and used on-line. In many situations, it is not enough simply to train a neural network on a given collection of data and leave it to operate without further adjustment through experience. What if conditions arise which have not yet been encountered by a trained network? Does new information necessitate retraining? What happens to the existing body of information represented by the network if new information is incorporated? Some fixed network structures suffer the double problems of requiring off-line retraining to deal with new conditions and *catastrophic forgetting* where an established mapping is replaced by a new one (Sharkey and Sharkey, 1994).

The addition of new processing units requires the detection of novel information which cannot be incorporated into the existing structure. For the *resource allocation network* (RAN) of Platt (1991), the addition of new processing units depends upon a two part novelty condition. The first part deals with the input vector of a pattern pair. A pattern is novel if the input lies beyond a specified

distance from the nearest stored exemplar (centre). In layer 1 the specified limit decays with time to a resolvable minimum exemplar separation. The second part of the novelty condition deals with the output vector and states that a pattern is novel if the difference between the network output and the actual output exceeds a set limit. Initially the representation of the function is coarse; as learning proceeds, the allocated units have a reducing width until both novelty conditions are fulfilled.

Two supervised networks, ARTMAP and fuzzy ARTMAP, based on adaptive resonance theory (discussed in Chapter 2) have two part novelty conditions similar to those of the RAN but are not based upon Euclidean distance. The first part governs the allocation of new nodes to cover regions of the input space. The second condition deals with incorrectly predicted outputs and triggers corrective activity which may include the allocation of new nodes. Adaptive resonance theory offers a sophisticated and flexible approach to both mapping and pattern clustering.

Another problem that can occur with networks such as the MLP is *overtraining*. If an MLP is trained for too long, it can learn to reproduce the training data to a high degree of accuracy but fail to generalise to the underlying function. To prevent this, another data set is required to *validate* the training and to help in making the decision on whether to train for a shorter or longer period next time. There are no hard and fast rules and much experience is needed; the avoidance of overtraining is a large area for research. Using a *cost function* consisting of the error energy and a *regularization* term may help to overcome this (Bishop, 1995). The regularization term restricts the amount of curvature of the fitted function so that rapid changes (which allow tracking of noise) are avoided. However, regularization introduces *a priori* assumptions about the form of the underlying mapping and requires the choice of extra parameters.

Many neural networks, including the adaline and MLP, use supervised learning. One issue which will be considered in much more detail throughout this thesis is the availability of desired output patterns for supervised learning. Supervised learning requires that output labels or actions are available during training and this is not always the case, especially with dynamical systems. Sometimes only initial and final states are known but not a specified intermediate trajectory.

An alternative to supervised learning where pattern classes are not pre-specified is *pattern clustering* which involves sorting input patterns into groups without predefining a set of such groups; i.e. unsupervised learning. Members of the same group will have several features in common, that is, they will be "close together", in some sense, in input space. The groups or clusters of vectors in input space can be represented in many ways. A convenient way is to use a prototype or exemplar which represents the cluster as a set of abstract features, i.e. an 'average' example. For pattern classification, class labels may be added retrospectively if required.

The notion of "closeness" depends very much on how the input data is coded and what *metric* or *distance measure* is used. A common metric is Euclidean distance used in many neural networks (e.g. Haykin, 1994, Kohonen, 1989, 1995). Euclidean distance is not the only possible measure; networks based upon adaptive resonance theory use the sum of the components of a difference vector (see Chapter 2).

The self-organising map (SOM) of Kohonen (1989, 1995) consists of a number of nodes arranged on a two-dimensional lattice. Each node stores an *exemplar vector* which is representative of a local cluster of inputs. It is an unsupervised network which operates by allowing the nodes to compete for activation when an input is presented. The node with an exemplar nearest to the input is chosen as the winner and updated. The nodes of a neighbourhood around the winning node are also updated. The SOM competitive network can be applied to the process of *vector quantization* which involves the unsupervised compression and storage of input information by finding a set of exemplar vectors that represents the input

space in the most efficient way. The resultant representation tessellates the input space with a set of irregular convex polygons (regions) delineated by a set of intersecting hyperplanes; these hyperplanes represent the decision boundaries between neighbouring nodes when the choice of winning node (exemplar) is based upon the Euclidean distance between the input vector and all of the stored exemplars. The tessellation, known as the *Voronoi tessellation,* is illustrated in Figures 1.8. and 1.9. (Kohonen, 1989, 1995; Hertz, *et al*, 1991).

Figure 1.8. Selection of winning nodes based upon Euclidean distance leads to hyperplanar decision boundaries between nodes. The intersection of these hyperplanar boundaries defines the tessellation shown in Figure 1.9.

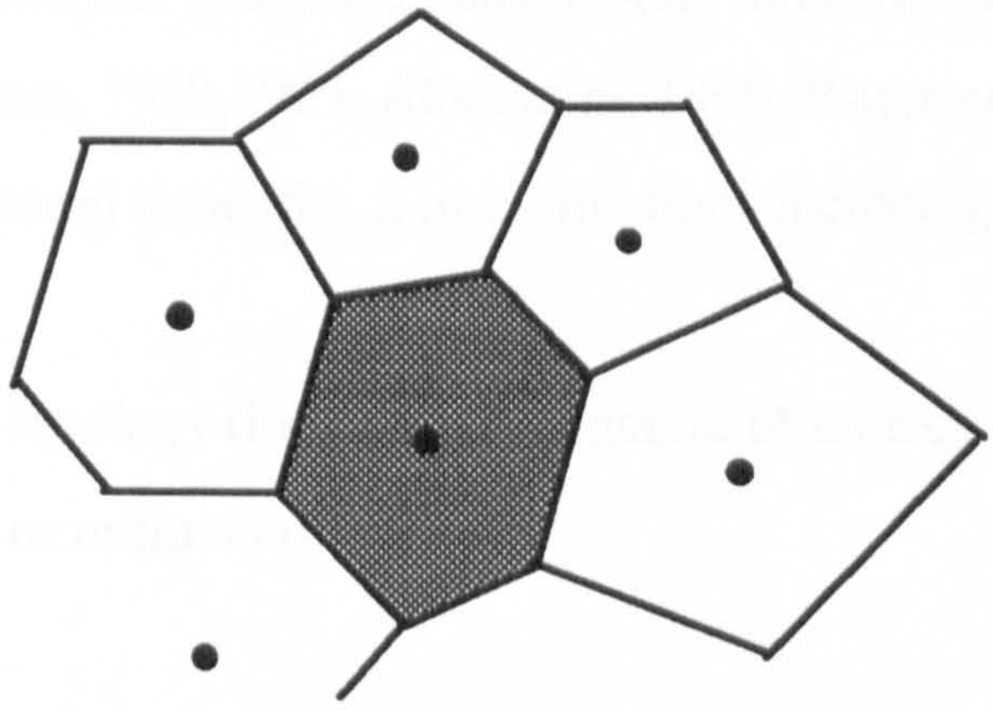


Figure 1.9. Using Euclidean clustering with winner-takes-all dynamics results in a partitioning of the input space that consists of irregular convex regions. This partitioning is known as a *Voronoi tessellation*

The set of all input vectors belonging to the same partition of the Voronoi tessellation is known as a *Voronoi set.* For the SOM winning node, the individual exemplar (weight) vectors, $\mathbf{w}_J$, will move in the direction of the difference vector $\mathbf{I} - \mathbf{w}_J(t)$ towards an input vector, $\mathbf{I}$, and come to represent nearby clusters of inputs to which they respond maximally. This is shown schematically in Figure 1.10.

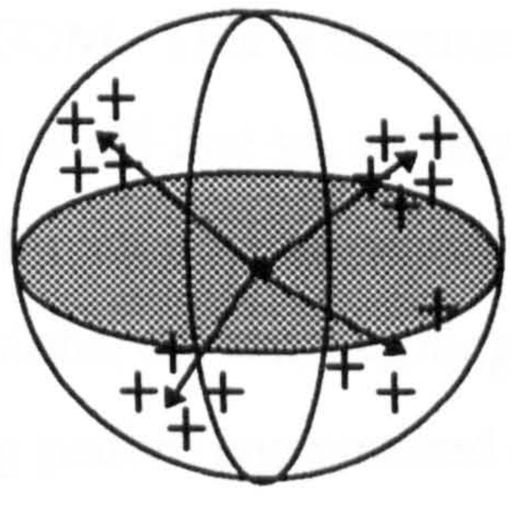


Figure 1.10. An illustration of clustering in three dimensions. The vectors represent the centres-of-mass (centroids) of the clusters.

The neighbourhood update mechanism allows topological relationships within the input space to be conserved within the lattice i.e. data items "close together" in input space will be stored close together in the lattice nodes (Kohonen, 1989, 1995). For discrete simulations, a continuous neighbourhood can approximated by updating all neighbours within a given region only. The SOM is a very effective learning system which has generated considerable interest in the neural network community (Kohonen, 1989,1995; Ahalt e*t al*, 1990; Ritter *et al* 1992). However, like all neural networks, it has limitations including:

- a *fixed network topology* (including 2-D nature of lattice) which does not allow for the addition or removal of nodes,

- *Rare data cases may be swamped* (Kohonen, 1995) which means that small statistical frequencies are not allotted any territory in the SOM,

- neighbourhood and learning rate *shrinkage schedules are arbitrary* and have no basis other than empirical judgement.

Other limitations such as lack of a well-defined cost function or absence of guaranteed convergence are mentioned by authors including Bishop (Bishop, 1995).

Self-organising Architectures based upon Adaptive Resonance Theory overcome some of the limitations of the SOM. This is discussed in Chapter 2 onwards. Another important issue is that of off-line vs. on-line learning, i.e acausal learning vs. causal learning. A decision must be made between these two options for any application. For some applications, off-line learning makes better use of the training data because a training pattern encountered early on during training may have a different significance later on; this is not possible with on-line learning and information may only be used once and discarded. Learning may be dependent upon the order of presentation for on-line learning problems.

One advantage of on-line learning is its flexibility. Take, for example, a control problem such as is considered in this thesis. With off-line learning, data is gathered and used to train a neural network controller (see section 1.7). Following training, the network is fixed and can only work within the bounds of its training experience. What if conditions change? How will a controller learn new strategies? With on-line learning, a controller may assimilate new information as and when it arises. The off-line vs. on-line dilemma also arises in the unsupervised learning case. For example the SOM may be trained using either mode but catastrophic forgetting may occur if the underlying statistics of the problem change.

1.7 Neurocontrollers

There exists a large body of knowledge regarding control engineering theory and techniques. Conventional techniques often involve *linear* control theory which is well established. Controller operating regions are chosen and linearised around a given set point. State-space methods can be applied to systems with models which have been simplified by linearising them. The linearity assumption is fulfilled because control objectives are to keep signals small but unexpected disturbances

can violate the linearity assumption by forcing operation out of the linear region; therefore a non-linear or piecewise linear approach is required.

Conventional controller design relies upon knowledge of the plant formulated in a plant model. The plant model is an input-output mapping which represents a plant's dynamical characteristics. Prior to controller design, the desired plant behaviour is formulated for comparison with the actual behaviour. A compensator is then designed to alter the open-loop plant characteristics resulting in desired plant behaviour (closed-loop). The controller is almost always fixed and often involves standard proportional-integral-differential (PID) control methods (e.g. Banks, 1986)

Obtaining a plant model is an important part of the controller design process and involves the techniques of system identification applied to model fitting to input-output data or to derive a mathematical model from first-principles using physical laws. Conventional control methods have been successful up to a point. However, the plant dynamics are often complex or little-known and, thus, may require more sophisticated control techniques than classically derived linear controllers. Where a system model is available, it may not necessarily have an inverse and neither may the real system; this makes model-based control more difficult.

Neural networks provide a flexible approach and are especially suited to mapping problems. The application of neural network methods to control is known as *neurocontrol* and involves the development of neural network based controllers or neurocontrollers. State-space can be quantised or represented in a smooth manner depending upon the neurocontroller architecture. Learning can be carried out off-line or on-line depending upon the situation. On-line learning is usually known as *adaptive control* and allows neurocontrollers to adapt continuously to changing plant conditions or parameters.

The plant can be treated as a "black box" if no model is available and a neural network can be used to model the plant using the plant's input-output

relationships. If the desired controller actions are known—a controller input-output relationship has been found either implicitly or explicitly— this information can be used to train a neurocontroller using supervised learning. The trained neurocontroller will represent a control mapping. The on-line establishment of control mappings will be covered in more detail in sections 1.8 and 1.9.

Application of neural networks to control problems must be done with caution. Using neural networks may not necessarily be the best approach. The "model" obtained may be no more useful than the set of input-output relationships specified in the training data or obtained during plant operation. The mapping specified by the neural network may not be transparent or give a parsimonious representation of the plant. Model complexity depends upon many factors and the choice of a representative model class may beg the question by requiring knowledge of plant dynamics of the type being sought.

In addition to the problem of over-complex models with too many parameters, there is the problem of adequate network complexity. A neural network may have insufficient "capacity" to represent a dynamical system; this under-parameterisation may lead to instability and inadequate control because some of the important dynamical modes may not be represented by the model.

A more fundamental problem concerns the availability of adequate training data in the first place. For a system identification or parameter estimation task, information regarding desired plant behaviour is required to "tune" the neurocontroller during learning; this feedback signal allows the neurocontroller to associate control actions with the input data (Figure 1.11).

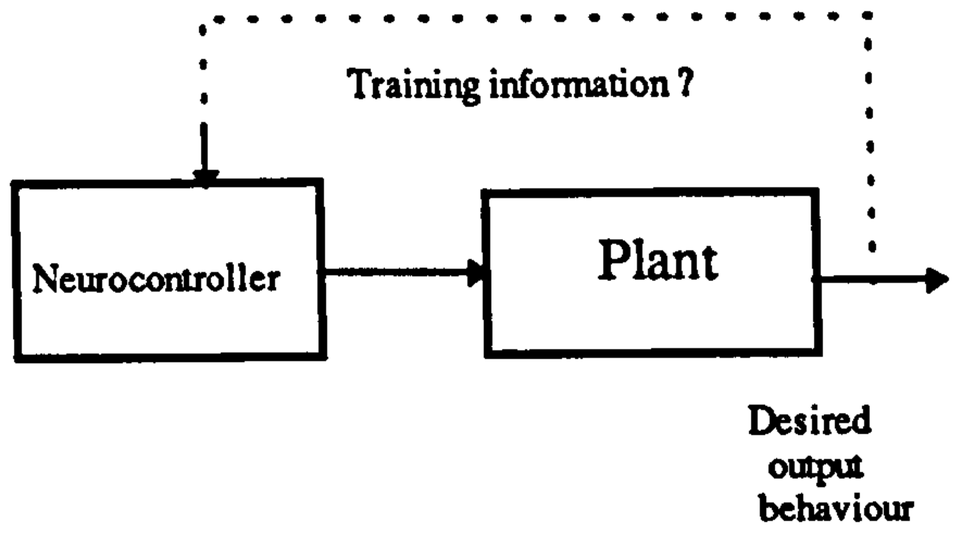


Figure 1.11 The neurocontroller training problem. What is fed back and how is it used?

As mentioned previously, where a data set of input-output examples exists, supervised learning can be used to train an associative memory network. Desired control output data is not always available and assumes that the control problem has been solved to some extent before using a neurocontroller. If this were not the case, how would desired control outputs be known for at least some of the situations encountered? Even if the control mapping is not represented as a mathematical model over a specified range of the input space—presumably the task of the neural network is to find such a mapping, implicitly or otherwise—a subset of the input-output set is available from some source. That source may be from a human expert or an existing controller (operator modelling) and represents a relationship between a system's inputs and outputs.

When specifying training data, care must be taken to ensure that the data set is of adequate size and sufficiently representative of the problem to allow determination of the model parameters—plant or neurocontroller model—to the required accuracy. The specification of performance criteria must be appropriate to the control problem being solved and the protocol must be sufficiently general so as to be applicable to other methods so that meaningful performance comparisons are possible. What about unknown systems with complex, non-linear dynamics which may include delays? How can information about plant delays be incorporated into any representation or model? If the order of the model is underestimated, it may lead to instability and an insufficiently effective control strategy.

How can neurocontrollers be made more autonomous and capable of extracting information for themselves during learning? How much, if any, *a priori* knowledge is to be included in a candidate neurocontroller? What is the source of this knowledge?

This brief discussion of some of the issues involved in neurocontrol has raised various considerations, some of which will be discussed further in later sections.

1.8 Adaptive Control

One area of control theory of relevance here is adaptive control. Adaptive control is a natural extension of feedback control (Åström, 1995). Control performance can be improved by increasing controller autonomy. Augmented error feedback is used to adjust controller parameters on-line (Figure 1.12). Many applications involve the automatic tuning of a simple controller (Åström, 1995).

Adaptive control can be divided into two general methods: *indirect* and *direct.*

- *Indirect* adaptive control involves on-line modelling of a plant and the synthesis of a control law from the model by inverting or otherwise using the model. The plant model is used to predict an output which is used to determine a control action. The success of the control strategy depends upon the accuracy of the plant model. One of the objectives is to estimate a set of plant parameters which specify the specific plant model. Closed-loop systems obtained using adaptive control are non-linear and complex owing to the parameter adjustment mechanism (Åström, 1995).

- *Direct* adaptive control builds an explicit model of a controller *on-line* without necessarily referring to an explicit plant model. Neurocontrollers using direct adaptive control have potential advantages over simple adaptive control

methods that use gain scheduling to change the parameters of a linear controller operating in a local region. The associative memory properties of some architectures allow the construction of a control map composed of multiple regions which may or may not be covered by a linear surface (hyperplane). If the learning algorithm is sufficiently stable—in the sense that there is little risk of catastrophic forgetting or overwriting (Sharkey and Sharkey, 1994)—then continual recomputation of locally operative controller models is not required. Recomputation of locally linear models is computationally expensive and unnecessary. Adaptability is desirable, but with the added constraint that a control strategy is not generated anew when re-entering a region of input or state-space for which a strategy was developed previously. In other words, a candidate neurocontroller must be adaptive but not memoryless. Two particular implementations of the direct adaptive control method are discussed at length in this thesis. Before discussing these implementations in detail, these two examples of direct adaptive control will be reviewed briefly to illustrate the concept.

Reinforcement learning methods can be viewed as "... a computationally simple, direct approach to the adaptive optimal control of nonlinear systems." (Sutton, Barto and Williams, 1992). Data is gathered on-line and used to compute a control output and performance evaluation at each time-step. Reinforcement learning will be covered in detail in Chapter 3 onwards where novel architectures are introduced.

Self-organising fuzzy control: this method attempts to build up a rule-base (commonly on-line) which represents a successful control strategy. The rule-base is modified according to the direct evaluation of control actions without reference to a plant model. Evaluation is usually in the form of an explicit look-up table (matrix) specifying corrections for each rule given the current error and change in error. A novel form of self-organising fuzzy control using an evaluation network to replace the look-up table is described in Chapter 5.

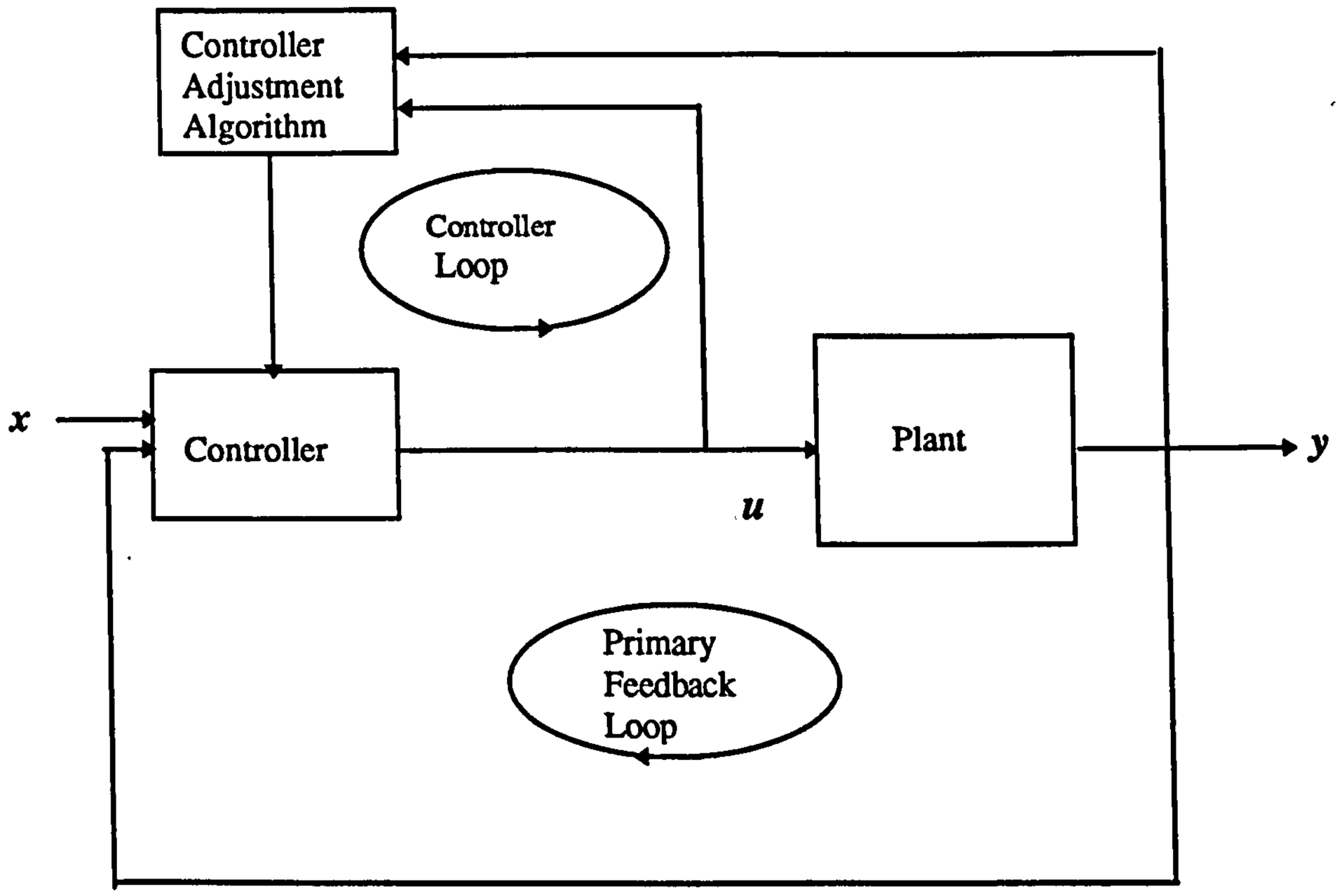


Figure 1.12 Block diagram of an adaptive system (after Åström, 1995). Two control loops are shown indicating the addition of a secondary loop which is involved in the adaptive tuning of the controller.

1.9 Delay Learning

This thesis will explore an application of the incremental paradigm to the dynamic partitioning of state space for control and related problems. As reviewed in section 1.7 it is very often difficult to establish more than crude qualitative information about state space trajectories on all but the simplest of analytical systems. Ascertaining an accurate model of system dynamics and contriving an objective or cost function signifying desired behaviour is usually the preferred route in optimal control problems. Most adaptive methods are indirect and use an estimated system model to recompute controls at each step (Sutton *et al*, 1992). Even if adequate knowledge is available, the *a priori* integration of this knowledge into the network structure can severely limit the autonomy and flexibility of the network. Autonomous learning systems need to be able to extract and organise information during experience in their particular data-rich environment, increasing their information capacity as necessary.

Consideration of autonomous, self-organising systems reveals another, related, aspect. The world exhibits obscure structure to any observer and convenient labels, indicating the spatiotemporal significance of, and relationships between, objects or events are simply not available *a priori* (Edelman, 1989). It is very clearly ordered but, often, the causal relationships between objects and events are not understood. For example, living organisms make sense of the world through experience and evaluation of behavioural consequences. They structure experiences autonomously and develop conceptual schemata with which to classify perceptual stimuli as a basis for future behavioural responses. Where desired responses are available for neural network training, the initial learning problem has been solved autonomously by a human operator who has organised and correlated relevant information to provide training data for the associative network. This is especially true in control applications where desirable control actions have to be specified and presented to a given neural network along with the conditions which necessitate such actions. To increase neural network autonomy, the processes of information extraction and organisation must be incorporated into the architecture to allow more intelligent use of "raw" data; the integration of pre-processing sub-systems into a neural network may reduce dependency on external pre-processing and may consequently increase network autonomy.

Additional motivation for the use of incremental self-organising systems for complex control tasks is the inadequacy of supervised learning in control problems. The problematic use of a fixed structure network is compounded by the lack of training information about the structure and dynamics of the environment or plant. The autonomy of an intelligent agent or neurocontroller lies in its ability to extract information from the environment.

One requirement of an adaptive learning system is that it be capable of dealing with delayed effects in the environment. Action and reaction are not instantaneous with the effects of control actions still having an effect beyond the instant of application. A system must be able to integrate the effects of delayed

control actions into the control assessment procedure. For the supervised learning method, the provision of input-output pattern pairs by an external teacher is an artificial process which relies upon several underlying presuppositions for its operation as a training method. One such being that of a temporal connection between input-output pairs; the nature of the assumed temporal connection forms the basis for a model of the state transition dynamics of the system being controlled. In other words, assumptions regarding the relative timing between stimuli and responses in a system determine the form of the system model; if those assumptions are wrong, or cannot be accommodated by the neural network architecture being used, then control is unlikely to be successful. An adequate representation of state transition dynamics is a prerequisite for successful control as these dynamics determine system responses to stimuli through state transitions that depend on both the present state and the current input. As well as a possible change in the current response, there might be a transition to a new state. These dynamics characterise a system and must be represented in some way by a candidate neurocontroller.

A sizeable proportion of neural network theory is based upon *associationism* which has its historical roots in psychology (James, 1892). Learning laws which associate pre-synaptic and post-synaptic outputs (Hebb, 1949) often assume little or no time delay between correlated signals. Networks based upon these learning laws, and variants of them, function as simple pattern associators which strengthen connections between frequently associated patterns, and which weaken others.

The effects of an input on state transitions are not usually limited to instantaneous changes unless memoryless systems are considered; a more accurate assessment of real world systems is that state transitions are influenced by inputs as a function of the time interval between a particular input and a given state transition. This temporal effect reduces the validity of simplistic stimulus-response pairing of input and output to some extent. Problems which involve delayed feedback to a learning system can be reduced to simple pattern association tasks but require a problem to be solved beforehand by a human teacher in order to specify the

optimal actions which should be taken by the learning system (Myers, 1992). One way to take delay into account is to present delayed inputs as part of the pattern pair. However, this requires assumptions about the system model; for example, what is the minimum delay time that can be assumed for a good model? Incorporating time delay information into the training data set increases the dimensionality of the input space.

Both the difficulty of obtaining relevant input-output pairs and the issue of the temporal connection between inputs and state transitions (and, therefore, outputs) are addressed in the paradigm of reinforcement learning (Barto *et al*, 1983; Sutton, 1988, 1992; Barto, 1992; Sutton *et al*, 1992; Daynan & Hinton, 1993); this paradigm will be considered in Chapter 3. A modified form of the backpropagation algorithm, *temporal backpropagation*,(e.g. Werbos, 1990) has been developed to overcome some of the problems associated with using feedforward neural networks for learning temporally correlated sequences of inputs. *Recurrent networks* with feedback connections, such as the *Elman network* (Elman, 1990) have also been developed; these and other approaches to the delay problem will be covered in section 4.2.10.

This chapter has provided a general background to the work presented in this thesis. Further introductory material will be given where appropriate to the discussion. Chapter two begins with an introduction to Adaptive Resonance Theory which underlies two of the novel architectures developed to overcome the limitations of some existing approaches to mapping and control problems.

Chapter 2 Adaptive Resonance Theory and PROBART

2.1 Generic ART

Cognitive psychology investigates the functions of perception and cognition and highlights many fundamental questions about the information processing capabilities of humans and animals (e.g. Best, 1992). For example,

- How do humans "make sense" of their environment? (Edelman, 1989)
- how do they order experience in time and space by discovering, learning and recognising invariant properties in the world? (Carpenter and Grossberg, 1987a).

Adaptation and survival of any organism within an environment requires that information is extracted, organised and acted upon in an efficient manner. Even general behavioural constraints observed from the natural world, although many are still without neural correlates (physiological counterparts), nevertheless impose conditions upon neural models both artificial and natural. Characteristics of intelligent information processing include the ability to:

- self-organise representational codes within the brain to order information,
- abstract invariant properties from the environment,
- generate and test hypotheses,
- maintain expectations to compare with "reality",
- form stable representations of formed categories but still be able to
- respond to significant inputs.

2.1.1 The Stability-Plasticity Dilemma

The last two points form the basis of the fundamental problem of competing constraints known as the *stability-plasticity dilemma* (Carpenter and Grossberg, 1987a). It is based upon the premise that it is desirable for any useful intelligent learning system to have two fundamental properties.

The first property is *plasticity*. Any system must be adaptive enough to respond to a series of environmental inputs. During learning, state transitions, resulting from inputs and the present state of the system, must lead to new steady states and attractors representing *critical feature patterns* (otherwise known as *prototypes* or *exemplars*) of the environment. Thus the system (or organism) must self-organise internal representations of invariant environmental features. It must be capable of recognising novelty and accommodating new inputs from the environment into its growing repository of experience—plasticity. This responsiveness to new information must not, however, be at the expense of previously established knowledge structures. That is, new learning should not disrupt old learning. This leads to the second property of *stability*. Learned representations must be stable regardless of new incoming information; invariant properties of objects/situations must be abstracted and isolated from detected irrelevancies. For instance, a person is able to recognise their best friend regardless of lighting conditions and changes of clothing etc. Thus, an invariant and stable internal representation of the best friend can be postulated.

The dilemma can be stated simply as a question: how can a system remain plastic enough to respond to novel stimuli and yet retain stable invariant representations against relearning and recoding? This question gives rise to the following issues involved in the development of artificial systems:

- self-organisation and representation of information
- the abstraction of invariant properties
- stability
- plasticity
- causal (possibly real-time) operation.

Adaptive Resonance Theory has been developed to deal with these complex issues and consideration of the stability-plasticity dilemma is reflected in the ART family of networks (Carpenter and Grossberg, 1986; 1987a,b; 1989; 1992; 1994) which has gone some way to resolving the conflict between stability and plasticity within neural networks.

Adaptive resonance theory comprises one of the major themes of this thesis and will be illustrated in both the contexts of off-line and on-line learning. Two novel architectures—one off-line and one on-line—are presented in sections 2.6 and 4.2 respectively. The novel off-line architecture, PROBART, is applied to the general mapping problem and is supervised during learning.

ART networks have the ability to allocate nodes dynamically, as required during processing, without the need for retraining to incorporate novel information; this property provides a natural basis for on-line adaptive learning. This thesis also presents a novel network, EUCART—based loosely upon some of the principles of ART— which is applied to a well-known control problem and acts as a self-organising state space decoder to provide an autonomously derived internal representation of state space. As a prelude to the description of the novel architectures, some members of the class of architectures which comprise ART will be considered in significant detail. Before this, however, it is desirable to get an overview of the ART philosophy.

2.1.2 The Instar and Outstar Elements

ART architectures are based upon an underlying structure consisting of two fundamental types of computing element, the *instar* (Grossberg, 1976a,b,; Harvey, 1994; Levine, 1991) and the *outstar* (Grossberg, 1968, 1980; Levine, 1991). Both elements form basic components in Kohonen's self-organising maps (Kohonen, 1989, 1995), the counterpropagation network of Hecht-Nielsen (1990) and the ART 1 network of Carpenter and Grossberg (1987a). A detailed

discussion of these artificial neural elements will not be given here; instead, this sub-section will provide a motivational overview so that further details of instar and outstar dynamics can be introduced during subsequent discussion of ART dynamics where appropriate.

The instar is an information processing element which learns to represent a cluster of inputs by developing *long-term memory* (LTM); it consists of a single *sink* node and a number of *source* nodes. A single instar element is shown in Figure 2.1. Note that, although there are a number of input nodes and an output node, the instar is treated as a single element.

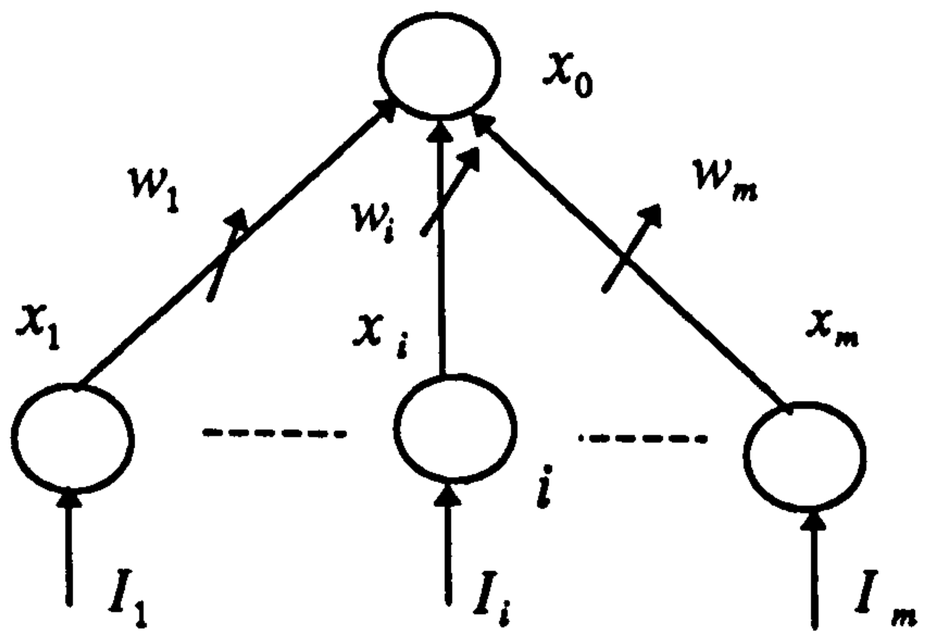


Figure 2.1. The instar element which consists of a set of input nodes and a single output node.

The instar operates upon an m dimensional input vector denoted by $\mathbf{I} = (I_1,\ldots,I_i,\ldots,I_m)^t$ which is fed directly to a layer of input neurons with activities denoted by $\mathbf{x} = (x_1,\ldots,x_i,\ldots,x_m)^t$ in vector form. The sink node activity is denoted by x_0. The LTM is stored as a set of weights denoted by $\mathbf{w} = (w_1,\ldots,w_i,\ldots,w_m)^t$ in vector form. The LTM activity is governed by the passive decay LTM equation (Grossberg, 1968). Weights decay if no input vector is present or the sink node is not active; a *gated instar* may be used to prevent this. The state of the instar is reflected in the activation functions which determine the *short term memory* (STM) trace or activity level for a given input vector (pattern). The instar output (sink) node dynamics are governed by an *additive STM equation* (without shunting) (Grossberg, 1968, 1980; Nigrin, 1993).

Whilst an input is present, the instar weight vector will tend towards the input vector. Instars learn to represent the stream of spatial patterns that they are exposed to; the representation will be an average of the pattern types depending upon exposure time.

If a layer of instars is formed, each instar will respond maximally to a given input, or cluster of inputs and the layer comprises a self-organising pattern classification system whose categories are represented by the output activation values; such a system is illustrated in Figure 2.2. which shows a competitive layer in which neurons compete for a share of the total activity available across the layer. If desired, an overall winner can be selected; this is known as *winner-takes-all* dynamics where the node with the largest activity wins the competition and signifies the resultant category (Rumelhart and Zipser, 1985).

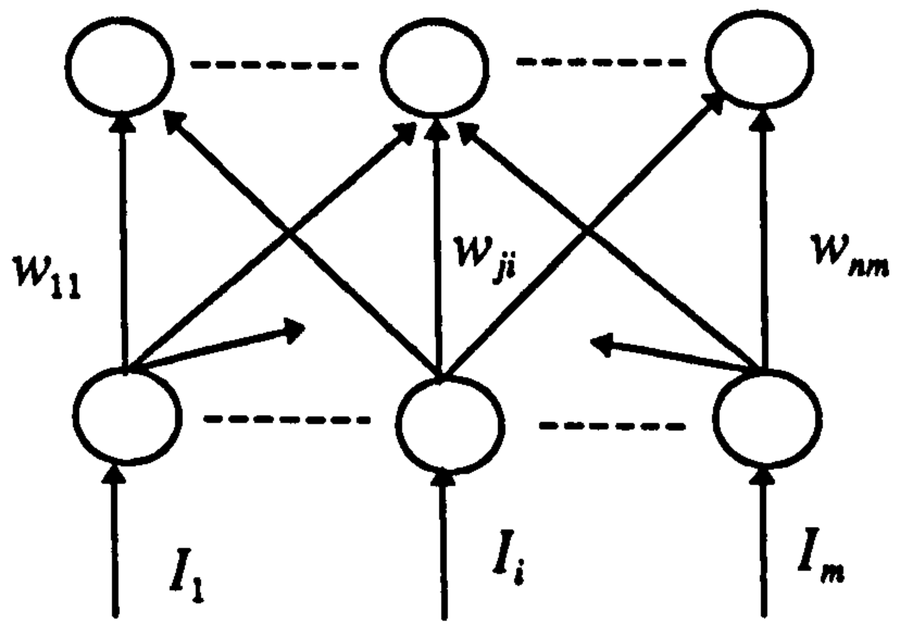


Figure 2.2. A neural network consisting of a layer of instars. Each node in the top layer is an instar sink and responds maximally to a cluster of inputs.

The underlying mechanism of the competitive network is the inhibitory interaction between sink nodes inspired by biological realism. These interactions are simplified to winner-takes-all dynamics by choosing the instar with the largest activation. By using a good choice of STM dynamical equation, it is permissible simply to take the node with the largest net input. The right choice of STM equation reflects the biologically plausible justification for winner-takes-all dynamics based upon the "closeness of match" between an input vector and the stored exemplars. If normalised input patterns are used, the following two points link competitive dynamics through inhibitory interactions and pattern clustering based upon Euclidean distance:

- competitive dynamics results in the choice of the node with the largest net input,

- normalisation of input and stored patterns implies that the node chosen with the largest net input reflects the smallest Euclidean distance between the stored pattern and the input pattern.

Biological plausibility does not mean that biological neural networks are concerned with normalisation and closeness of patterns in a Euclidean sense as if they were designed for some information processing purpose; networks exhibiting competitive pre-processing by *off-centre, on-surround networks* of the type found in biological organisms are capable of maintaining *relative reflectance patterns* (see section 2.1.4) regardless of absolute signal magnitude, and also bounding the inputs.

The *outstar*, is the minimal network capable of classical conditioning. It is able to learn arbitrary spatial patterns and recall them when stimulated. It consists of a single *source* node and several *sink* nodes (Figure 2.3) in contrast to the instar.

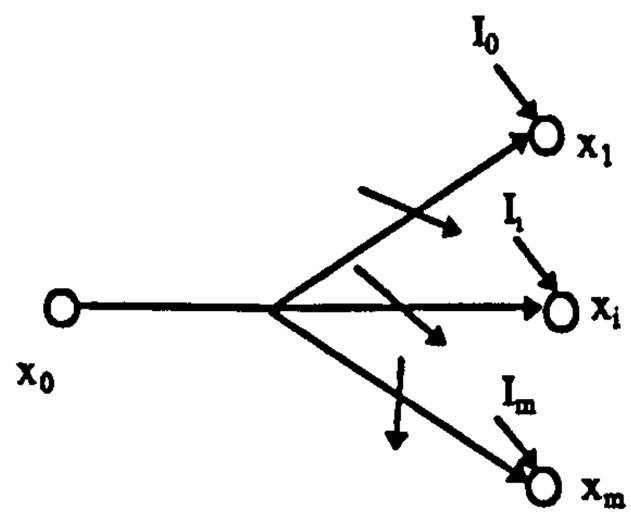


Figure 2.3. The outstar of Grossberg (Grossberg, 1968, 1980). It consists of a source node and several sink nodes.

The source node activation is denoted by x_0 and, when trained, is able to elicit an activity pattern $\mathbf{x} = (x_1, x_2, \ldots, x_i, \ldots, x_m)^t$ across sink nodes $1, \ldots, i, \ldots, m$. Consider an individual sink node as shown in Figure 2.4. As well as receiving a weighted connection from the source node with weight w_{i0}, the sink node receives a training input, I_i. This allows the outstar to learn an input pattern

$\mathbf{I} = (I_1, I_2, \ldots, I_i, \ldots, I_m)$ that is distributed across the sink nodes and to associate it with a source input, I_0.

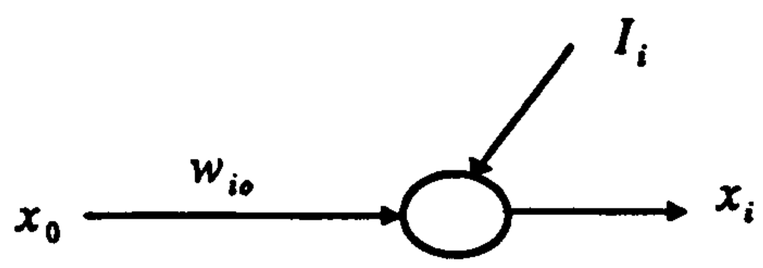


Figure 2.4. An isolated sink node showing the training input and the source node input.

The basic operation of the outstar is as follows:

- an input pattern, $\mathbf{I}$ is presented to the sink nodes of the outstar at the same time as the source node input, I_0.

- The LTM traces (weights) are modified to store the pattern across the source nodes while the training inputs are active.

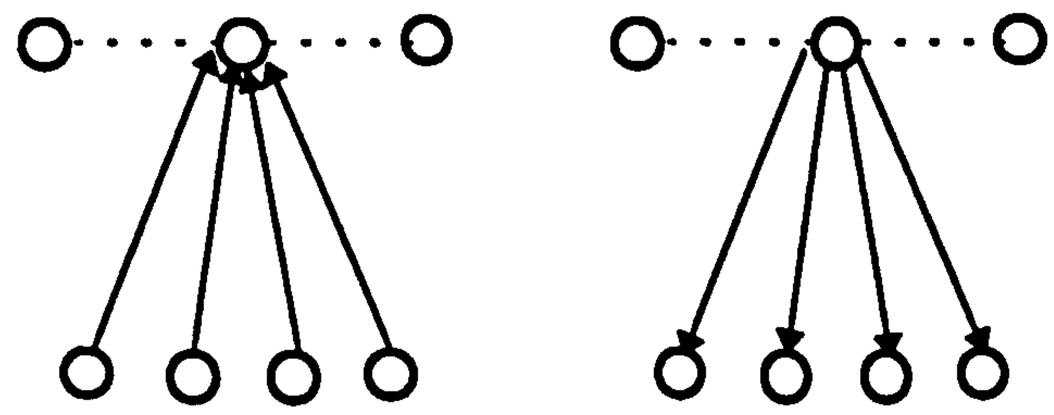
- When the training input is removed, the LTM traces retain the pattern; an outstar with this post-input retention property is known as a *gated outstar*

- when the source node input, is presented to the network following training, the associated pattern, $\mathbf{I}$, is recalled.

2.1.3 A Generic ART Module

If two layers of nodes are used in which the individual nodes of one layer act alternatively as an instar sink node and an outstar source node and the nodes of the other layer act accordingly as instar source and outstar sink nodes, then the two layers taken together comprise an *autoassociative memory network* capable of storing recognising and recalling patterns. Figure 2.5 shows the two phases of such a two layer system.

The simple, two phase system forms the basis for ART architectures; ART variants stem from the inclusion of various mechanisms into an underlying two-layer two-phase system. Before considering specific architectures, some of the refinements to the simple autoassociative network will be considered; the refined generic ART module will form the basis for subsequent discussion.



(a) Recognition phase (b) Recall phase

Figure 2.5. An autoassociative memory network consisting of two operational phases: (a) a recognition phase using the instar mode to select a winning node and (b) a recall phase using the outstar mode to recall the stored pattern.

A generic ART module comprises two layers or fields of nodes, the matching field and choice field which are labelled F1 and F2 respectively; this is shown in Figure 2.6 which illustrates a simplified model without additional features introduced later. There is a third layer, F0 which merely acts as a buffer for an input vector and is not counted. The F2 nodes act as sink nodes of inputs (instars) and source nodes of outputs (outstars) during a learning cycle involving a single input pattern.

The algorithm for a single cycle of a generic ART module is given by:

1. an input vector, **I**, is transferred from the buffer layer F0 to the matching field F1 and gives rise to an activity pattern, **x**
2. the activity pattern of F1 is then transmitted to the F2 layer nodes via a set of weighted connections; each F2 node acts as an instar which responds to a given input filtered by the bottom-up connections
3. the activation pattern across F2 , **y** is then *contrast enhanced* to find the maximally responsive instar for the current input; this is a winner-takes-all competition.
4. the winning instar sink node is then treated as an outstar source node which projects back down (top-down filter) to F1
5. the pattern elicited across F1 by the active F2 outstar is the prototype average stored by the winner called the *expectation* $\mathbf{x}'$
6. the current F1 layer activity pattern, **x** and the top-down expectation, $\mathbf{x}'$ are combined to give a resultant pattern $\mathbf{x}^*$ across F1.
7. if the match between **x** and $\mathbf{x}^*$ is sufficient, *resonance* is said to occur and the input pattern has been recognised and learning takes place
8. if not, the winning F2 node is inhibited and the whole cycle repeats from step 1
9. if all used nodes are exhausted the a F2 new node is recruited. A node recruited to represent a category is said to be *committed*, otherwise it is *uncommitted*.

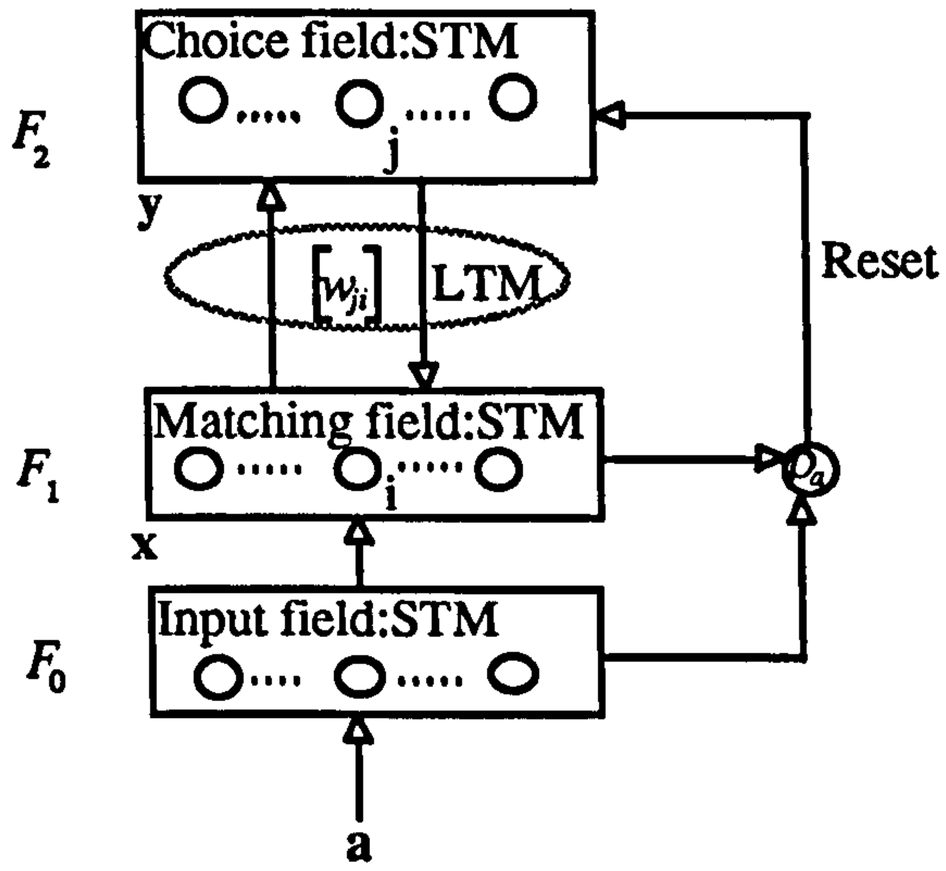
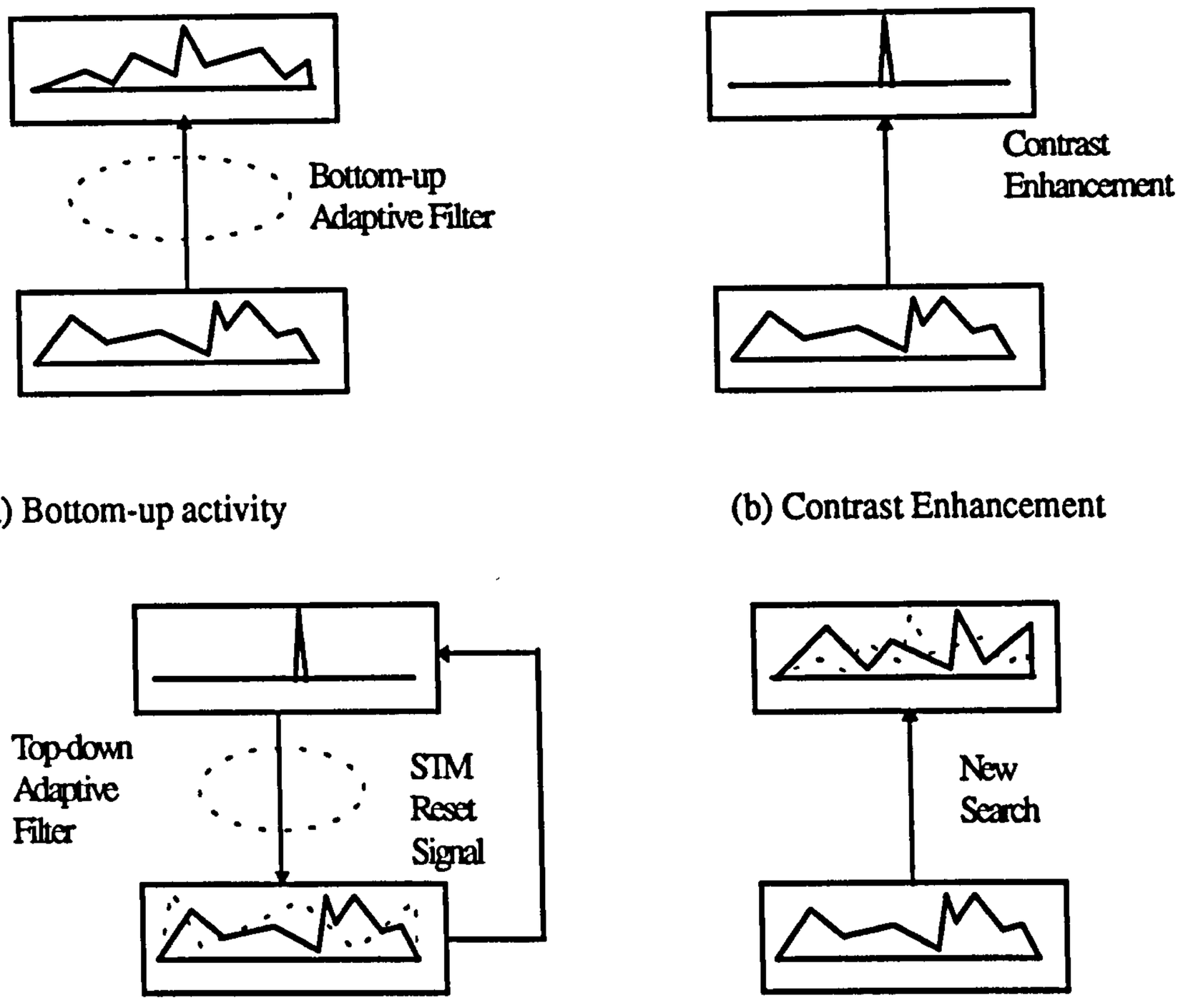


Figure 2.6 A generic ART module consisting of two layers. Note that layer F0 is an input buffer layer. This diagram is simplified, with a number of additional features not shown. These features are discussed in the text.

Figure 2.7 illustrates the pattern matching cycle schematically; in the case shown, a mismatch at F1 leads to a new search cycle. The ART algorithm is generic and, as such, is not specific to any one of the ART architectures. Indeed, a number of issues have to be dealt with prior to implementation. These include:

- specifying the method of combining activity patterns across F1
- establishing criteria for a "sufficient match"
- defining what constitutes an "activation pattern"
- specifying the method of "contrast enhancement"
- defining how pattern categories are chosen, and
- specifying the form of learning for the bottom-up and top-down weights which constitute the filters.

The examination of specific ART architectures will exemplify these issues. The first ART architecture to be examined in detail is ART 1 (Carpenter and Grossberg, 1987a). This will exemplify the ART paradigm and expand upon the issues mentioned and raise new issues. The consideration of generic ART within this subsection will provide the framework for exploring specific ART implementations and further modifications.



(a) Bottom-up activity (b) Contrast Enhancement

(c) Top-down activity: (matching) (d) Reset

Figure 2.7 A generic ART pattern matching cycle showing reset after pattern mismatch. (a) The F1 activity pattern is first transmitted to F2 where (b) the F2 activity is contrast enhanced (relative differences in activity levels are exaggerated) to give an overall winner. For stage (c) the top-down expectation of the F2 winner is transmitted to F1 to compare with the current input. (d) if the current input is not what is expected, the F2 winner is inhibited and competition is resumed.

2.1.4 Two Subsystems

A generic ART architecture is split into two subsystems for functional convenience: the *attentional subsystem* and the *orienting subsystem* (Carpenter and Grossberg, 1987a) (Figure 2.8).

The attentional subsystem deals with familiar events and the responses to those events through the internal representation of structure in the outside world. The

SHEFFIELD

attentional subsystem, by itself, cannot maintain stable category codes and create new categories for unfamiliar patterns. Alone it would cause categories to become rigid and produce no new categories or to exhibit ceaseless recoding, hence the requirement for the orienting subsystem.

The orienting subsystem deals with novel patterns and can distinguish whether a given pattern is sufficiently familiar or whether it is of a new type and requires a new category.

Attention plays an important role in the self-organisation of recognition codes. Three types of attentional mechanism, attentional priming, attentional vigilance, and attentional gain control will be discussed in the context of ART learning.

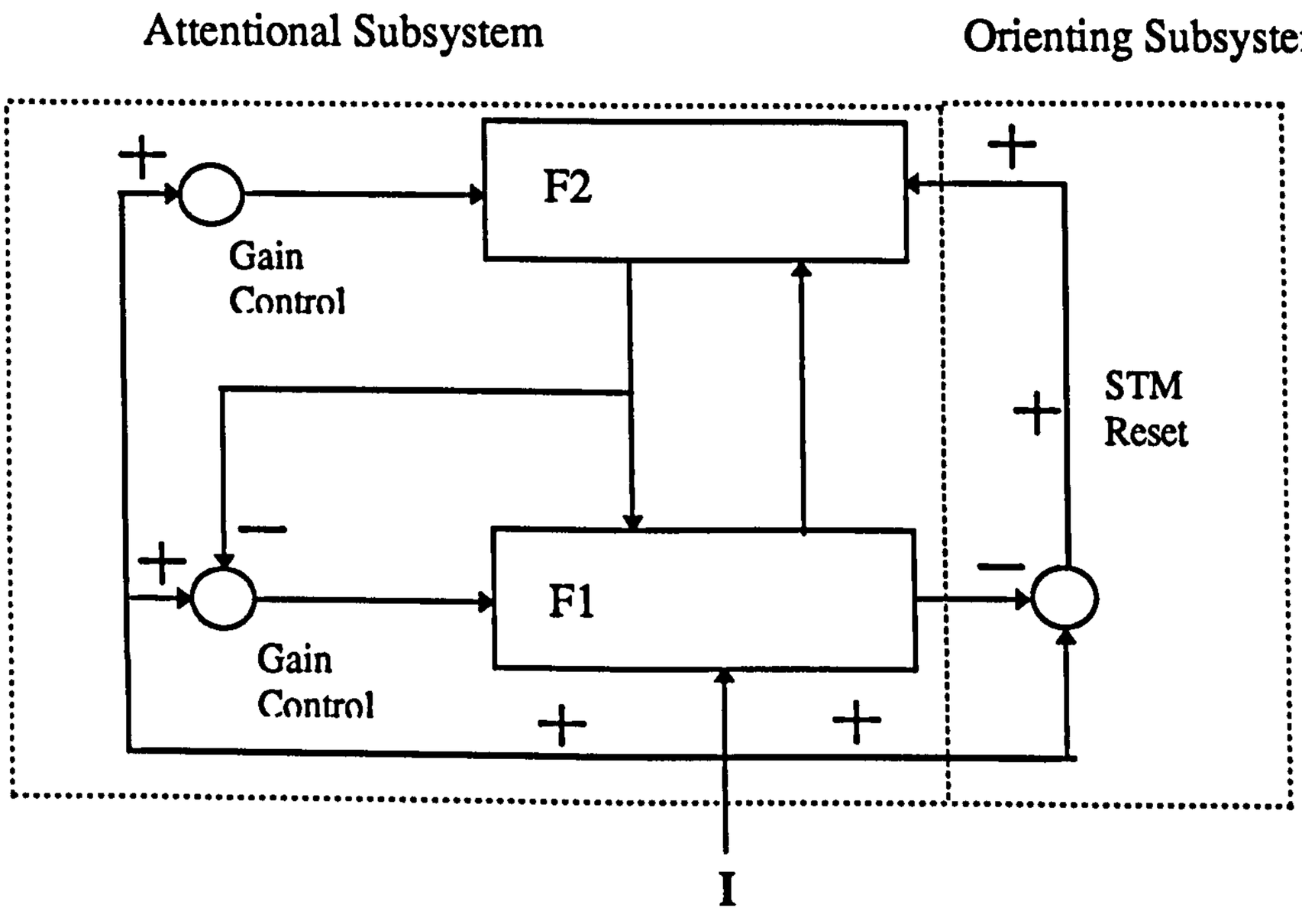


Figure 2.8 The attentional and orienting subsystems of a generic ART module.

ART places few restrictions on patterns such as orthogonality. There are no limitations on storage capacity; an ART architecture can store arbitrarily many patterns without degradation of sensitivity to novel information. ART units or nodes store *critical features* in the form of *spatial patterns*. Absolute magnitude can be misleading and data bits (binary valued ART) have a context dependent significance; self-scaling of patterns is carried out to enable invariant pattern

recognition. For an input $\mathbf{I} = (I_1(t),\ldots,I_i(t),\ldots,I_m(t))$, if $\mathbf{I}$ is a spatial pattern, then relative activities of the components maintain a constant relationship to one another regardless of the absolute activity (sum) of the pattern components. Thus,

$$\mathbf{I} = (I_1(t),\ldots,I_i(t),\ldots,I_m(t)) = (\theta_1 I(t),\ldots,\theta_i I(t),\ldots,\theta_m I(t)) = (\theta_1,\ldots,\theta_i,\ldots,\theta_m) I(t)$$

with the convention that $\sum_{i=1}^{m} \theta_i = 1$ to ensure that $I(t) = \sum_{i=1}^{m} I_i(t)$ (Grossberg, 1980).

Humans and animals recognise objects quickly. ART has no complex retrieval mechanism and access to recognition codes is direct (Carpenter and Grossberg, 1987a). Indeed, ART algorithms could be implemented in parallel on a suitable analogue or digital computer.

Although ART networks self organise recognition codes, they have the ability to alter attentional sensitivity in response to environmental influences. This possible 'teaching' mechanism—which allows negative reinforcement to increase sensitivity to incorrect category recognition—is known as *attentional vigilance* (Carpenter and Grossberg, 1987a). Changes to the ART network sensitivity induced by mismatch of actual and expected inputs is implemented in the supervised ARTMAP architecture (Carpenter, Grossberg and Reynolds, 1991); this attention increasing mechanism, known as match-tracking is discussed in section 2.3.

Another attentional mechanism is that of *attentional priming*. Top-down priming or expectation is fed down from F1 to F2 even when the input has been removed and before a new input is present. This top down input signal causes changes to the level of F1 activation thus priming F1. In this case, F1 activity does not necessarily elicit a bottom-up signal; F1 merely remains more sensitive to input from F0. However, an input pattern from F0 must be sufficient to cause activity across F1 without top-down processing, such that a bottom up signal is elicited. This poses a question. How are input signals from F2 and F0 distinguished by

F1? It is important that F0 derived signals are of sufficient strength to elicit bottom-up activity and consequent competition and recognition, yet misleading self-excitation be prevented.

The solution to this problem is provided by another attentional mechanism, *attentional gain control*, which allows the F1 layer to differentiate between signal sources. A gain controls F1 sensitivity depending upon the signal source; the operation of the attentional gain control mechanism is governed by the *two-thirds rule*.

2.1.5 The Two-Thirds Rule

The effective co-ordination of activity in a generic ART module requires that certain subtleties of operation be addressed. These subtleties are mainly concerned with maintaining sensible signal levels to avoid excessive auto-excitation or inhibition.

The attentional gain control allows modulation of signal levels as required and provides a third signal source to augment those of F1 and F2. ART operation is succinctly represented by the *two-thirds rule* of Carpenter and Grossberg (1987a) which states that two out of the three signal sources are required to be present to activate the F1 nodes. Table 2.1 states the conditions under which the gain is active.

Status	F1	F2	Gain
TD+BU	1	1	0
BU only	1	0	1
TD only	0	1	0
no activity	0	0	0

Table 2.1 The gain signal is switched on when there is activity across F1 owing to an input signal from F0. This is to ensure that the F1 activity is sufficiently strong to elicit activity across F2. TD signifies that a top-down signal (F2 to F1) is present and BU a bottom-up signal.

2.2 ART 1

The ART 1 architecture (Carpenter and Grossberg, 1987a) is a specific implementation of the generic ART algorithm given in section 2.1.3. It is a self-organising binary vector clustering system and forms part of the supervised ARTMAP architecture of section 2.3. A simplified account of the ART 1 dynamics will be given with a view to simulation. The details of the development of ART 1 dynamics from their biological origins, although interesting and informative, are of no direct relevance to this discussion. A more comprehensive account may be found elsewhere (e.g. Grossberg, 1987, 1988; Freeman and Skapura, 1992)

2.2.1 The F1 Layer: Input Phase

Let the input to the F0 layer of an ART 1 module at time, t, be denoted by $\mathbf{I}(t) = \left(I_1(t), I_2(t), \ldots, I_i(t), \ldots, I_M(t)\right)$ and the activity across the F1 layer by $\mathbf{x}(t) = \left(x_1(t), x_2(t), \ldots, x_i(t), \ldots, x_M(t)\right)$ where M is the number of F1 nodes. The activity across the F2 layer is given by $\mathbf{y}(t) = \left(y_1(t), y_2(t), \ldots, y_j(t), \ldots, y_N(t)\right)$ where N is the number of F2 nodes.

The total excitatory input to an F1 layer node is given by

$$x_i(t) = I_i(t) + g + V_i \qquad (2.1)$$

Where g is the *gain* signal specified by

$$g = \begin{cases} 1 & \text{if F0 is active and F2 is inactive} \\ 0 & \text{otherwise} \end{cases}$$

and the net top-down input from layer F2 to layer F1 for the i th F1 node is given by

$$V_i = \sum_{j=1}^{N} y_j w_{ij}^{(TD)} \qquad (2.2)$$

where $w_{ij}^{(TD)}$ is the connection strength (or weight) from the j th F2 node to the i th node of layer F1 (Figure 2.9).

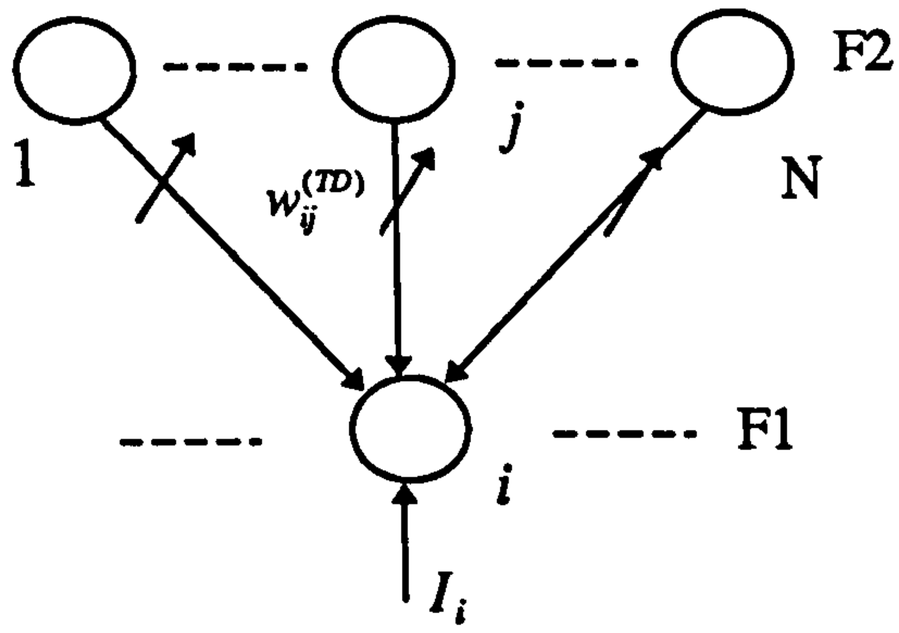


Figure 2.9 The input sources to an F1 layer node.

The F1 layer weights, representing F1 layer LTM, will be discussed in section 2.2.4.

If an input vector *is* present but F2 is *not* active (no net top-down input possible) then the condition $I_i > 0$ causes the i th F1 node to fire owing to the non-specific gain which increases input sensitivity. Without it the input signal may not be of sufficient strength to produce a feedforward signal. When both an input *is* present and F2 *is* active, then both the input and the top-down activity is sufficient to cause F1 layer nodes to fire.

During the input phase, the input, I(t), is propagated to F1 giving $\mathbf{x}(t) = \mathbf{I}(t)$.
Because no top-down signal is present, g=1 and $x_i(t) = I_i(t) + 1$.

2.2.2 The F2 Layer: the Bottom-Up Phase

The expression for the bottom-up net input to a F2 layer unit from layer F1 is analogous to equation (2.2) viz.:

$$T_j(t) = net_j(t) = \sum_{i=1}^{M} x_i(t) w_{ji}^{(BU)}(t) \qquad (2.3)$$

where $w_{ji}^{(BU)}$ is the connection strength (or weight) from the i th F1 node to the j th node of layer F2 (Figure 2.10).

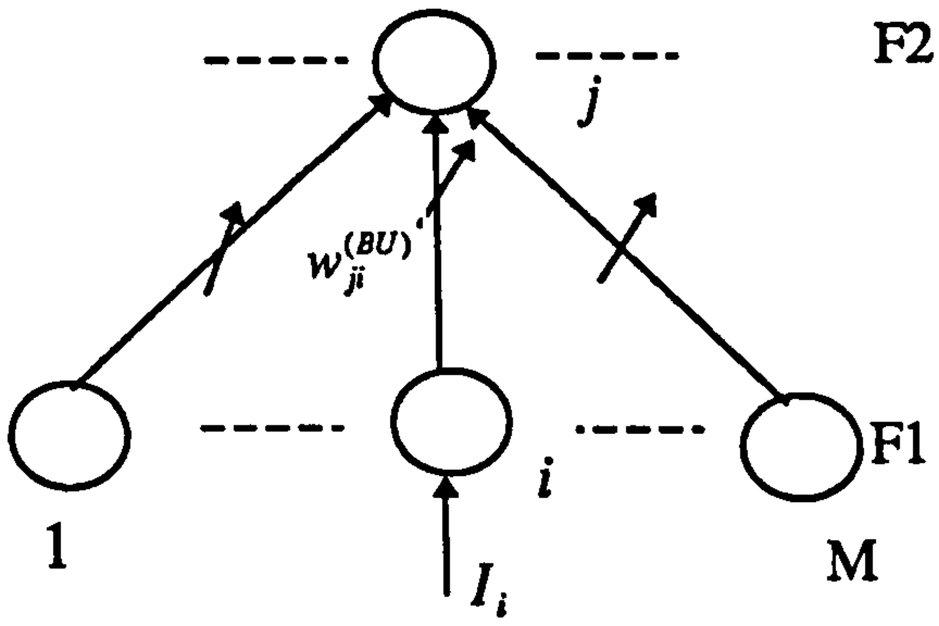


Figure 2.10. The F2 layer nodes of an ART1 module operating in instar mode.

Thus, there is a distributed activation pattern across layer F2 depending upon the level of net input to the individual F2 nodes (Figure 2.11)

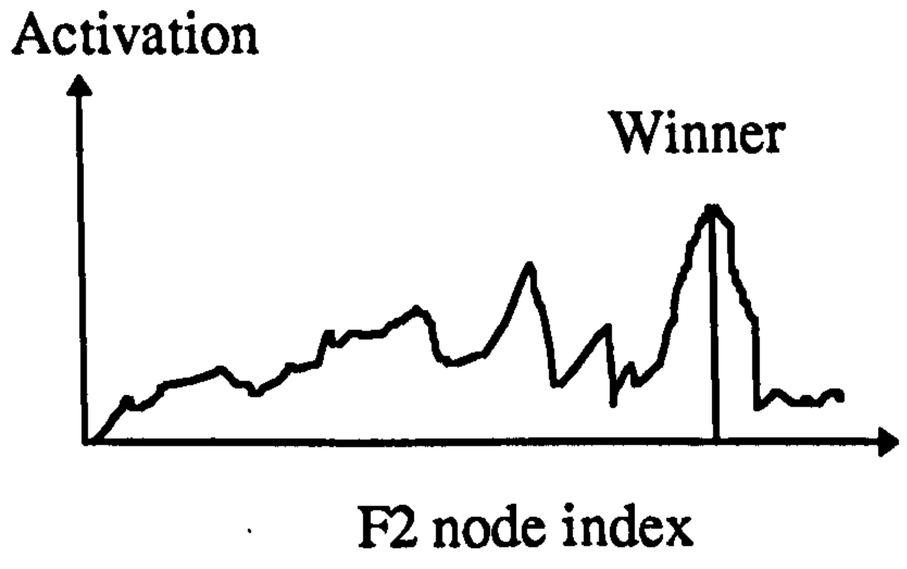


Figure 2.11. An illustration of a typical distributed activation pattern across F1. The pattern will be contrast enhanced to find the winning node.

The F2 layer dynamics can be simplified considerably by assuming a *winner-takes-all* function of the form

$$y_j = \begin{cases} 1 & if \quad T_J = \max_k \{T_k\}, \forall k \\ 0 & otherwise \end{cases} \qquad (2.4)$$

where J is the index of the winning F2 node.

2.2.3 The F1 Layer: the Top-Down Phase

Top-down activity alone should be insufficient to cause F1 layer activity. However, top-down activity should have some effect on the F1 layer; this effect is to *prime* the layer ready for incoming signals so that the probability of F1 layer activity is increased.

To analyse the contribution, V_i made by the F2 layer, certain assumptions (Freeman and Skapura, 1992) are made about the F2 layer. These assumptions are:

- only a single F2 node has a nonzero output at any given time
- the maximum output of an F2 node is 1, and
- the maximum weight on a top-down connection is also 1.

The first assumption is reasonable in that after inter-nodal competition across F2, a single node becomes dominant and represents the category chosen by the ART module. The second and third assumptions are based upon design considerations.

Returning to the net input value to the i th F1 layer node given by equation (2.2)

$$V_i(t) = \sum_{j=1}^{N} y_j(t) w_{ij}^{(TD)}(t) = y_J(t) w_{iJ}^{(TD)}(t) = w_{iJ}^{(TD)}(t) \qquad (2.5)$$

for some winning F2 node J. All the other F2 layer nodes have zero output.

When top-down activity is present, applying the gain criterion to equation (2.1) gives

$x_i(t) = I_i(t) + w_{ij}^{((TD))}(t)$, or, in vector form, $\mathbf{x}(t) = \mathbf{I}(t) + \mathbf{w}_J^{(TD)}(t)$ where the weight vector of top-down LTM traces is represented as $\mathbf{w}_J^{(TD)} = \left\{ w_{1j}^{(TD)}, w_{2j}^{(TD)}, \ldots, w_{Mj}^{(TD)} \right\}$. The F2 generated top-down activity is combined with the F0 input activity to give a resultant activity across F1. For

some $0 < \bar{z} < 1$, the condition *if* $x_i(t) \geq 1 + \bar{z}$ *then* $x_i(t) = 1$ is applied to each component of the F1 activity vector which is equivalent to

$$\mathbf{x}(t) = \mathbf{I}(t) \cap \mathbf{w}_j^{(TD)}(t) \qquad (2.6)$$

where $\cap$ is the *intersection* or the *logical AND* operation defined by

$(p_i \cap q_i) = \begin{cases} 1 & \textit{iff} \quad p_i = q_i = 1 \\ 0 & \textit{otherwise} \end{cases}$ for the binary vectors $\mathbf{p}$ and $\mathbf{q}$.

2.2.4 The F1 Layer: Top-Down LTM Traces

Short-term activities (STM) are alone insufficient to provide a basis for adaptive learning systems; longer term information must be stored for future activity and adaptation to an information rich environment. This is where long term memory—in the form of weights on connections—becomes important. It can be stated that $\frac{d}{dt}\mathbf{w}_j^{(TD)} = \left(-\mathbf{w}_j^{(TD)} + \mathbf{x}\right) y_j$ which is the learning law for a *gated outstar*. For the winning F2 node J, $\frac{d}{dt}\mathbf{w}_J^{(TD)} = \left(\mathbf{x} - \mathbf{w}_J^{(TD)}\right)$ which implies that the weights of the winning F2 node, J, move in the direction of $\mathbf{x} - \mathbf{w}_J^{(TD)}$. This describes the difference vector between the current set of weights and the desired asymptotic value $\mathbf{x}$ representing F1 activity. Changes in $\mathbf{w}_J^{(TD)}$ are made proportional to this difference vector $\mathbf{x} - \mathbf{w}_J^{(TD)}$ (Figure 2.12); this ensures that $\mathbf{w}_J^{(TD)} \rightarrow \mathbf{x}$ as required. For *fast learning*, disregarding transients, $\mathbf{w}_J^{(TD)} = \mathbf{x}$.

The weights tend to represent the signal present across the F1 layer when the particular F2 node was excited. The top-down weight update equation is thus given by

$$\mathbf{w}_j^{(TD)}(t+1) = \begin{cases} \mathbf{x}(t) & \textit{if} \quad j = J \\ \mathbf{w}_j^{(TD)}(t) & \textit{otherwise} \end{cases}$$

This is the fast learning model which assumes that the input patterns remain active on F1 for a time exceeding that required for equilibrium to be achieved following transient activity across F1 (Freeman and Skapura, 1992).

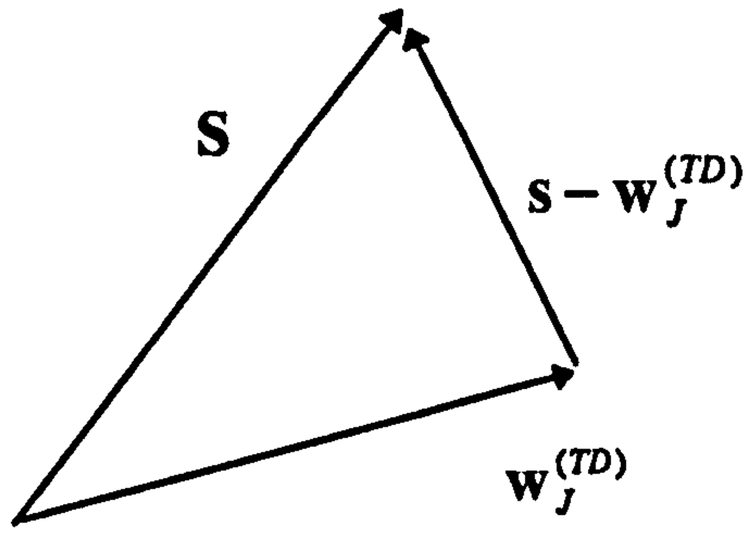


Figure 2.12. The movement of the top-down weight vector in the direction of the input vector. For fast learning, it is set equal to the activity across F1.

Although the weights are on connections feeding into the F1 layer, they are treated as belonging to the F2 layer nodes for convenience, i.e. a top-down weight vector is associated with each F2 layer node. The F2 layer is treated as a layer of gated outstars feeding into layer F1; only one outstar is active at any one time. The F2 nodes represent input categories arising from the self-organising activities of the ART 1 module. For the active F2 layer node, the associated weight vector tends towards the F1 layer output vector; this represents the top-down expectation of the F2 layer for subsequent processing cycles.

2.2.5 Matching Across F1.

Referring to back to the discussion of generic ART 1 in section 2.1, for a correct match to occur (across layer F1), it is required that the top-down signal fed to the F1 layer by the winning F2 node approaches the F1 layer output pattern. From the discussion of section 2.2.4, when learning a new input, this is indeed the case.

Denoting, the number of 1's across the F1 layer by $|\mathbf{x}|$ where $|\mathbf{x}| = \sum_{i=1}^{M} x_i$ the F1 layer matching condition for ART 1, which indicates that the correct F2 class or cluster has been triggered, is given by

$$\frac{|\mathbf{x}(t)|}{|\mathbf{I}(t)|} = \frac{|\mathbf{I}(t) \cap \mathbf{w}_J^{(TD)}(t)|}{|\mathbf{I}(t)|} \geq \rho \qquad (2.7)$$

where ρ is a threshold constant called the vigilance parameter. If there is no match between the current input and the top-down expectation, then the winning F2 node is inhibited and competition resumes.

2.2.6 The F2 Layer: Bottom-Up LTM Traces

The weight update equation for the bottom-up LTM traces (weights) is given by

$$w_{Ji}^{(BU)}(t+1) = \frac{x_i(t)}{\alpha + |\mathbf{x}(t)|} = \frac{w_{iJ}^{(TD)}(t+1)}{\alpha + |\mathbf{x}(t)|} \qquad (2.8)$$

(because $w_{iJ}^{(TD)}(t+1) = x_i(t)$) where, α is a positive constant. For a discussion of why this form is used, refer to Freeman and Skapura, (1992) suffice to say that it allows the comparison between the resultant across F1 and a previously stored resultant. This comparison facilitates the choice of a winning F2 node.

For the instars of Figure 2.12, recalling equation (2.3) the net input

$$T_j(t) = net_j(t) = \sum_{i=1}^{M} x_i(t) w_{ji}^{(BU)}(t)$$

It is clear that the node with the largest net input is the eventual F2 layer winner but what factors determine this? That is, can a function of the input vector, **I** be

derived which explicitly indicates the dependency of the F2 layer node choice upon the input? An explicit *choice function* $T_j(\mathbf{I})$ would allow comparison between F2 node activities for a given input. To derive such a function, the dependency of x_i and $w_{ji}^{(BU)}$ upon the input vector, $\mathbf{I}$ must be considered.

When F2 is inactive, the signal across F1 is given by $\mathbf{x}(t) = \mathbf{I}(t)$, that is, some unaltered pattern is transmitted to F2 (via the weights) for competition to occur in order to select a winning node or category. If F2 is active then the winning F2 node, j=J feeds down an exemplar or template representing the top-down expectation. Recall that this *top-down expectation* vector is denoted by $\mathbf{w}_J^{(TD)} = \left\{w_{1j}^{(TD)}, w_{2j}^{(TD)}, \ldots, w_{Mj}^{(TD)}\right\}$. This expectation or template vector is fed down and combined with the input vector present across F1 giving $\mathbf{x}(t) = \mathbf{I}(t) \cap \mathbf{w}_j^{(TD)}(t)$ by (2.6). In short,

$$\mathbf{x}(t) = \begin{cases} \mathbf{I}(t) & F2 \quad inactive \\ \mathbf{I}(t) \cap \mathbf{w}_j^{(TD)}(t) & F2 \quad active \end{cases} \qquad (2.9).$$

Substituting Equation (2.8) into Equation (2.3) gives

$$T_j(t) = \sum_{i=1}^{M} x_i(t) w_{ji}^{(BU)}(t) = \sum_{i=1}^{M} x_i(t) \frac{w_{ij}^{(TD)}(t)}{\alpha + \left|\mathbf{x}(t-\tau)\right|} = \sum_{i=1}^{M} I_i(t) \frac{w_{ij}^{(TD)}(t)}{\alpha + \left|\mathbf{x}(t-\tau)\right|} = \frac{\left|\mathbf{I}(t) \cap \mathbf{w}_j^{(TD)}(t)\right|}{\alpha + \left|\mathbf{w}_j^{(TD)}(t)\right|}$$

the time delay, τ is used to signify that F1 activity was stored in the top-down weights some time previously. The F2 choice function is now given by

$$T_j(\mathbf{I}(t)) = \frac{\left|\mathbf{I}(t) \cap \mathbf{w}_j^{(TD)}(t)\right|}{\alpha + \left|\mathbf{w}_j^{(TD)}(t)\right|} \qquad (2.10)$$

This form of function for the F2 choice function has important properties as will become apparent in the following example.

2.2.7 Bottom-Up Dynamics: an Example

Consider two input patterns $\mathbf{I}_1$ and $\mathbf{I}_2$ where $\mathbf{I}_1 \subset \mathbf{I}_2$, that is, pattern 1 is a strict subset of pattern 2. For correct recall, pattern 1 must trigger its own associated node and not that of its superset, pattern 2. Assume that F2 nodes 1 and 2 are associated with patterns $\mathbf{I}_1$ and $\mathbf{I}_2$ respectively. Thus,

$$w_{1i}^{(BU)} = \begin{cases} \dfrac{1}{\alpha + |\mathbf{I}_1|} & \forall i \quad s.t. \quad I_{1i} = 1 \\ 0 & \forall i \quad s.t. \quad I_{1i} = 0 \end{cases} \qquad (2.11)$$

and,

$$w_{2i}^{(BU)} = \begin{cases} \dfrac{1}{\alpha + |\mathbf{I}_2|} & \forall i \quad s.t. \quad I_{2i} = 1 \\ 0 & \forall i \quad s.t. \quad I_{2i} = 0 \end{cases} \qquad (2.12)$$

where I_{1i} and I_{2i} are the i th component of patterns $\mathbf{I}_1$ and $\mathbf{I}_2$ respectively

Now, for $\mathbf{I}_1$, from Equations (2.3) (2.8) and (2.11)

$$T_1 = \sum_i x_i \frac{w_{i1}^{(TD)}(t)}{\alpha + |\mathbf{I}_1|} = \frac{|\mathbf{I}_1 \cap \mathbf{w}_1^{(TD)}|}{\alpha + |\mathbf{I}_1|} = \frac{|\mathbf{I}_1|}{\alpha + |\mathbf{I}_1|}$$

as $\mathbf{w}_1^{(TD)} = \mathbf{I}_1$ which was learned previously.

For F2 node 2, $T_2 = \sum_i x_i \frac{w_{i2}^{(TD)}(t)}{\alpha + |\mathbf{I}_2|} = \frac{|\mathbf{I}_1 \cap \mathbf{w}_2^{(TD)}|}{\alpha + |\mathbf{I}_2|} = \frac{|\mathbf{I}_1 \cap \mathbf{I}_2|}{\alpha + |\mathbf{I}_2|} = \frac{|\mathbf{I}_1 \cap \mathbf{I}_2|}{\alpha + |\mathbf{I}_2|} = \frac{|\mathbf{I}_1|}{\alpha + |\mathbf{I}_2|}$

because $\mathbf{w}_2^{(TD)} = \mathbf{I}_2$ and $\mathbf{I}_1 \subset \mathbf{I}_2$. So, $T_1 > T_2$ giving the winning F2 node J=1 because $|\mathbf{I}_1| < |\mathbf{I}_2|$ thus $\frac{1}{\alpha + |\mathbf{I}_1|} > \frac{1}{\alpha + |\mathbf{I}_2|}$.

The presentation of pattern $\mathbf{I}_2$ gives $T_1 = \frac{|\mathbf{I}_2 \cap \mathbf{I}_1|}{\alpha + |\mathbf{I}_1|} = \frac{|\mathbf{I}_1|}{\alpha + |\mathbf{I}_1|}$ and

$T_2 = \frac{|\mathbf{I}_2 \cap \mathbf{I}_2|}{\alpha + |\mathbf{I}_2|} = \frac{|\mathbf{I}_2|}{\alpha + |\mathbf{I}_2|}$.

Thus, by the monotonic increasing property of the choice function (see Appendix A) $|\mathbf{I}_2| > |\mathbf{I}_1| \Rightarrow \frac{L|\mathbf{I}_2|}{\alpha + |\mathbf{I}_2|} > \frac{L|\mathbf{I}_1|}{\alpha + |\mathbf{I}_1|}$ and the fact that $|\mathbf{I}_2| > |\mathbf{I}_1| \quad (\mathbf{I}_1 \subset \mathbf{I}_2)$ it is the case that $T_2 > T_1$ as required.

2.2.8 Simulating ART 1

What must the initial bottom-up weight values be? From equation (2.10) above, if the initial weight values are too large another uncommitted node could be incorrectly triggered resulting in errors being propagated throughout the system. To prevent this, consider the worst possible case where all bits in the input layer F0 are set to 1. If this pattern is learned $w_{ji}^{(BU)}(t+1) = \frac{1}{\alpha + |M|} \quad \forall i \quad i = 1..M$ so all weights must fulfil the initial condition $w_{ji}^{(BU)}(0) < \frac{1}{\alpha + |M|}$ in order to prevent an uncommitted node from winning incorrectly. So, let $w_{ji}^{(BU)}(0) = \frac{1}{\alpha + |M|} - \delta$ where δ is a small constant. Thus, for $|\mathbf{x}| \leq M$

$$\frac{1}{\alpha + |\mathbf{x}|} > \frac{1}{\alpha + |M|} - \delta \text{ or } \frac{1}{\alpha + |\mathbf{w}_j^{(BU)}(t)|} > \frac{1}{\alpha + |M|} - \delta \text{ for all } t.$$

For the initial top-down weights, $w_{ij}^{(TD)}(0) = 1.0$ to ensure that $\mathbf{w}_j^{(TD)}(t) \cap \mathbf{x}(t) = \mathbf{x}(t)$ in the beginning for an uncommitted node.

An ART 1 algorithm:

1. Initialise weights
2. Input, I(t),
3. Propagate to F1: $\mathbf{x}(t) = \mathbf{I}(t)$
4. Propagate to F2 using

$$net_j(t) = \sum_{i=1}^{M} w_{ji}^{(BU)}(t) x_i(t)$$

5. Find winning F2 node J such that
6. $net_J = \max_j \{net_j\}$

   or choose first index if there is a tie.
7. For the category chosen by winner-takes-all set

$$y_j = \begin{cases} 1 & if \quad j = J \\ 0 & otherwise \end{cases}$$

8. Propagate back to F1 (top-down processing): where

$$\mathbf{x}'(t) = \mathbf{I}(t) \cap \mathbf{w}_J^{(TD)}(t)$$

9. Match: For resonance the match condition $\frac{|\mathbf{x}'|}{|\mathbf{I}|} \geq \rho$ must be met.
10. If a match occurs, the weights are updated according to:

$w_{Ji}^{(BU)}(t+1) = \frac{1}{\alpha + |\mathbf{x}(t)|}$, if F1 node i is active and $w_{iJ}^{(TD)}(t+1) = x_i$ (Equivalent to

$\mathbf{w}_J^{(TD)}(t+1) = \mathbf{I}(t) \cap \mathbf{w}_J^{(TD)}(t)$), then return to step 2 else

11. Inhibit the F2 winner and repeat from step 3 until a winner is found or recruit a new node from the remaining uncommitted nodes. If there are no more nodes left, read in a new input vector at step 2.

For the incremental version, new F2 nodes may be added as required.

2.3 ARTMAP

ART 1 is a self-organising binary pattern clustering system which uses unsupervised learning. For supervised learning, two ART1 modules, ARTa and ARTb, are linked via a map field to form ARTMAP. Refer to Figure 2.15 which shows a continuous valued input variant, Fuzzy ARTMAP, which has a similar form but uses identical bottom-up and top-down weights; this variant will be covered in section 2.5.

ARTMAP (Carpenter, Grossberg and Reynolds, 1991) allows the association of binary patterns through supervised learning. The input and output spaces are self-organised by the two ART 1 modules, ARTa and ARTb respectively. These modules are linked by a *map field* which implements the mapping from input to output by associating ARTa and ARTb categories via compressed F2 codes. Dynamic control over the matching threshold in the ARTa module is provided by *match-tracking* (attentional vigilance). The ARTMAP algorithm will be discussed as a prelude to the discussion of fuzzy ARTMAP and a variant which forms part of the subject of this thesis.

2.3.1 The ARTMAP Algorithm

A brief discussion of notation is required to avoid possible confusion. Input vectors to the ART a and ART b modules are denoted by $\mathbf{I}^a$ and $\mathbf{I}^b$ respectively. The ARTa and ARTb F1 layers are denoted by F1a and F1b respectively. The number of nodes in layers F1a and F1b are denoted by M_a and M_b. Similarly N_a and N_b denote the number of nodes in F2a and F2b respectively.

The top-down weights in ARTa and ARTb are denoted by $w_{ij}^{(TDa)}(t)$ and $w_{ij}^{(TDb)}(t)$ respectively. Similarly, the bottom-up weights are denoted by $w_{ji}^{(BUa)}(t)$ and $w_{ji}^{(BUb)}(t)$ respectively.

Inputs $\mathbf{I}^a$ and $\mathbf{I}^b$ are presented to ARTa and ARTb respectively. The two ART1 modules are allowed to self-organise and produce F2 recognition codes for their respective inputs as per the ART 1 algorithm. The resultant ARTa and ARTb categories then have either to become associated or any current association verified. ARTMAP can learn associations between input and output space and can recall a response given an input stimulus. Figure 2.13 shows the situation schematically.

The dynamics of the map field are similar to those of F1 in that map field activation is given by $x_k^{ab} = y_k^b + G + y_J^a w_{kJ}^{(ab)}$ where G is a gain. When F2a and F2b are both active there is no gain i.e. $G=0$. The logical AND is carried out between the actual and expected F2 b patterns using the condition $x_k^{ab} > 1 + \overline{w}$ for $x_k^{ab} = 1$ else $x_k^{ab} = 0$ where $0 < \overline{w} < 1$.

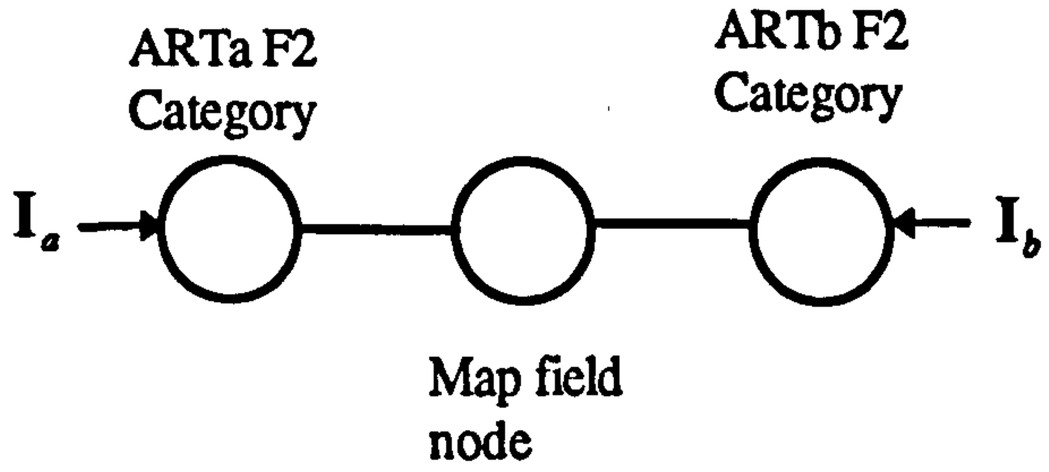


Figure 2.13 The linking of input and output categories via a map field in ARTMAP

The ARTMAP algorithm:

1. Initialise weights
2. Present inputs $\mathbf{I}^a$ and $\mathbf{I}^b$ to ARTa and ARTb respectively
3. Find winning ARTa and ARTa category nodes
4. Calculate map field activation using $\mathbf{x}^{ab} = \mathbf{y}^b \cap \mathbf{w}_J^{ab}$
5. match: $\frac{|\mathbf{x}^{ab}|}{|\mathbf{y}^b|} \geq \rho_{ab}$; if there is mismatch, increase the ARTa vigilance and trigger a new ART a search (step 2) else
6. Update: $w_{kJ}^{(ab)}(t+1) = x_k^{ab}$

2.3.2 ART 1 and ARTMAP: an Example

The following sample calculations are included to illustrate the ART 1 and ARTMAP architectures and to motivate the fuzzy ART / Fuzzy ARTMAP discussions of section 2.4 onwards. Pattern association by ARTMAP is covered because the operation of the ART 1 algorithm forms an integral part of ARTMAP and any separate discussion would be redundant. Only key points pertinent to the discussion will be given here; full details will be found in Appendix B.

The patterns to be associated in this simple example are:

$$\mathbf{I}_1^a, \quad 111110 \rightarrow 1010, \quad \mathbf{I}_1^b$$

$$\mathbf{I}_2^a, \quad 111100 \rightarrow 0101, \quad \mathbf{I}_2^b$$

$$\mathbf{I}_3^a, \quad 111000 \rightarrow 1010, \quad \mathbf{I}_1^b$$

The parameters used in this example are: $\alpha = 2.0$, $\delta = 0.01$, $M_a = N_a = 6$, $M_b = N_b = 4$. The baseline vigilance for ARTa, specified by $\overline{\rho}_a = 0.4$, illustrates match-tracking in ARTMAP. ARTb vigilance, $\rho_b = 0.9$

Present inputs:, $\mathbf{I}^a(1) = \mathbf{I}_1^a = [111110]^t$, $\mathbf{I}^b(1) = \mathbf{I}_1^b = [1010]^t$

After processing, the first ART a input vector is stored as the top-down expectation of F2 node 1 as indicated by the first column of the ART a top-down weight matrix:

$$W^{(TDa)} = \begin{bmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{bmatrix}$$

Similarly:

$$W^{(TDb)}(1) = \begin{bmatrix} 1.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & 1.0 \end{bmatrix}$$

where the first ART b vector is stored as the top-down expectation of ART b F2 node 1.

The map field weights are given by $w_{kJ}^{(ab)}(t+1) = x_k^{ab}(t)$ and, thus,

$$W^{ab}(1) = \begin{bmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{bmatrix}$$

indicating that ART a node 1 is linked with ART b node 1.

For the next cycle, $\mathbf{I}^a(2) = \mathbf{I}_2^a = [111100]^t$ and $\mathbf{I}^b(2) = \mathbf{I}_2^b = [0101]^t$. Note that $\mathbf{I}_2^a \subset \mathbf{I}_1^a$ and $\mathbf{I}_2^b = \left(\mathbf{I}_1^b\right)^c$

ARTb node 2 wins the competition this time and stores the second input vector at node 2.

For ARTa, owing to the subset property, the second input vector triggers ART a F2 node 1 giving the resultant (after top-down expectation):

$$\mathbf{x}^a(2) = \mathbf{I}_2^a \cap \mathbf{w}_1^{(TDa)}(1)$$
$$= \mathbf{I}_2^a \cap \mathbf{I}_1^a$$
$$= \mathbf{I}_2^a$$

Matching at ARTa gives, for $\mathbf{x}^b(2) = \mathbf{I}_2^a$, $\frac{|\mathbf{x}^a(2)|}{|\mathbf{I}^a(2)|} = \frac{|\mathbf{I}_2^a|}{|\mathbf{I}_2^a|} 1.0 \geq \rho_b = 0.9$ which is an accepted match. The F2 activity vectors for ARTa and ARTb are now given by $\mathbf{y}^a(2) = [100000]^t$ and $\mathbf{y}^b(2) = [0100]^t$ respectively.

ARTa node (category) 1 is linked with ARTb node (category) 1 and so a mismatch occurs. Match-tracking will not be of any uses because the ratio $\frac{|\mathbf{x}^a(2)|}{|\mathbf{I}^a(2)|} = \frac{|\mathbf{I}_2^a|}{|\mathbf{I}_2^a|} = 1$. Increasing the ART a vigilance parameter beyond unity is meaningless as no new ARTa node may be recruited to make the new required association between the second ART a input vector and the second ARTb input vector.

For the third input $\mathbf{I}_3^a \subset \mathbf{I}_1^a$ and $\mathbf{I}_1^a$ belongs to ARTa category 1 which is linked to ARTb category 1 as required.

2.3.3 Match-Tracking Revisited

For the case where the current ART a input is not a subset of a previous input and there is a match with an incorrect prediction, match tracking will allow the ARTa vigilance to be raised to ensure that a new search is triggered; this allows the current input to be distinguished from previous inputs so that the correct ARTb pattern can be associated with it.

For example, given the following inputs to be associated:

$\mathbf{I}_1^a$, $100111 \rightarrow 1010$, $\mathbf{I}_1^b$

$\mathbf{I}_2^a$, $110110 \rightarrow 0101$, $\mathbf{I}_2^b$

Say pattern 1 is stored in an ARTa node and that it is pattern 2 which triggers that particular node. So, $\frac{|\mathbf{x}^a|}{|\mathbf{I}_2^a|} = \frac{|\mathbf{I}_2^a \cap \mathbf{w}_J^{(TDa)}|}{|\mathbf{I}_2^a|} = \frac{|\mathbf{I}_2^a \cap \mathbf{I}_1^a|}{|\mathbf{I}_2^a|} = \frac{\|[100110]\|}{\|[110110]\|} = \frac{3}{4} = 0.75$.

If the ARTa vigilance, ρ_a was, for example, 0.6 and, thus, gave the wrong prediction, it could be increased to 0.75+δ to avoid a subsequent incorrect prediction. If a new ARTa node was created during match tracking, then the new ARTa node would give $\mathbf{x}^a = \mathbf{I}^a \cap \mathbf{W}^{(TDa)} = \mathbf{I}^a \cap \mathbf{I}^a = \mathbf{I}^a$ ensuring that the ARTa match criterion is fulfilled because the ratio $\frac{|\mathbf{I}^a|}{|\mathbf{I}^a|} = 1.0$. The new ARTa node is the linked to the correct ARTb node via the map field. Next time the second pattern is presented, the newly created node wins the competition and ensures the correct prediction.

The case where an input pattern is a subset of a pattern encountered previously causes problems as illustrated in the computations. A solution to this problem is discussed next.

2.3.4 Complement Coding

The crux of the ARTMAP subset problem is that for some input **I** there is a stored weight **w** such that $\mathbf{I} \subset \mathbf{w}$. This situation must be prevented to allow the use of another category node or the recruitment of an uncommitted node.

To prevent dissimilar inputs from being subsets of one another, define a new input $\mathbf{I}' = [\mathbf{I}\ \mathbf{I}^c]$ where $\mathbf{I}^c$ has all "one" entries where "zero" entries were previously and vice versa. This technique known as *complement coding* (Carpenter, Grossberg and Reynolds, 1991) circumvents the subset problem as shown in the following theorem which ensures that the subset problem will not occur:

Theorem: The *ARTMAP Match-Tracking Theorem:* Any ARTa input which is not equal to any previously stored ARTa input will always trigger match tracking activity in ARTMAP if complement coding is used.

A proof is given in Appendix C. The property detailed by the ARTMAP match-tracking theorem ensures that match-tracking always allows associations between ARTa and ARTb nodes providing that the ARTa vectors are not equal even in the subset case described previously. To illustrate match-tracking when complement coding is used, the following numerical example gives the final top-down and map field weight matrices for the case where the ARTa vectors of the previous numerical example are complement coded to circumvent the subset problem.

2.3.5 An Example of Match-Tracking

Consider the following pattern pairs to be associated:

$\mathbf{I}_1'^a$, $11111000001 \rightarrow 1010$, $\mathbf{I}_1^b$

$\mathbf{I}_2'^a$, $11110000011 \rightarrow 0101$, $\mathbf{I}_2^b$

$\mathbf{I}_3'^a$, $11100000111 \rightarrow 1010$, $\mathbf{I}_1^b$

After all three pattern pairs have been presented the two top-down weight matrices and the map field weight matrix are given by:

$$\mathbf{W}^{(TDa)}(3) = \begin{bmatrix} 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 \\ 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 \\ 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 \\ 1.000 & 1.000 & 0.000 & 1.000 & 1.000 & 1.000 \\ 1.000 & 0.000 & 0.000 & 1.000 & 1.000 & 1.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 1.000 & 1.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 1.000 & 1.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 1.000 & 1.000 \\ 0.000 & 0.000 & 1.000 & 1.000 & 1.000 & 1.000 \\ 0.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 \\ 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 \end{bmatrix}$$

$$\mathbf{W}^{(TDb)}(3) = \begin{bmatrix} 1.000 & 0.000 & 1.000 & 1.000 \\ 0.000 & 1.000 & 1.000 & 1.000 \\ 1.000 & 0.000 & 1.000 & 1.000 \\ 0.000 & 1.000 & 1.000 & 1.000 \end{bmatrix}$$

$$\mathbf{W}^{(ab)}(3) = \begin{bmatrix} 1.000 & 0.000 & 1.000 & 1.000 & 1.000 & 1.000 \\ 0.000 & 1.000 & 0.000 & 1.000 & 1.000 & 1.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 1.000 & 1.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & 1.000 & 1.000 \end{bmatrix}$$

The first three columns of the ARTa top-down matrix show that the three ARTa input patterns have been stored separately. The ARTb top-down matrix is as before with two patterns stored. The final map field weight matrix indicates that all three associations have been made. Column 2 shows that ARTa node 2 is associated with ARTb node 2 as required.

So far, the case where $\mathbf{I}_j^a \neq \mathbf{I}_i^a$ has been dealt with but what about when $\mathbf{I}_j^a = \mathbf{I}_i^a$ and both ARTa inputs are to be associated with *different* ARTb inputs?

Here, $\mathbf{I}_j^a \rightarrow \mathbf{I}_j^b$ and $\mathbf{I}_i^a \rightarrow \mathbf{I}_i^b$ with $\mathbf{I}_j^b \neq \mathbf{I}_i^b$ Complement coding cannot help because $\mathbf{I}_j^a = \mathbf{I}_i^a$ implies that $\mathbf{I}_j'^a = \mathbf{I}_i'^a$. Match tracking will also be of no use because, following the association $\mathbf{I}_i^a \rightarrow \mathbf{I}_i^b$, input $\mathbf{I}_j^a = \mathbf{I}_i^a$ gives an ARTa match of 1.0. When the ARTb node is predicted incorrectly and the ARTa module is reset, the ARTa vigilance will be raised to a value greater than unity which is not allowed. Thus the association $\mathbf{I}_j^a \rightarrow \mathbf{I}_j^b$ will be ignored. What this means is that *no one-to-many mappings are possible with ARTMAP.*

2.4 Fuzzy ART

Both ART 1 and ARTMAP operate on binary valued data. A real valued version of ARTMAP, *Fuzzy ARTMAP* (Carpenter *et al*, 1992) can be constructed from real-valued analogues of ART 1 modules known as *fuzzy ART* modules (Carpenter, Grossberg and Rosen, 1991). Operation of fuzzy ARTMAP is analogous to that of ARTMAP.

2.4.1 Structure

Each fuzzy ART module consists of three fields, or layers, of nodes: an input field, a matching field and a choice field. A schematic outline of a fuzzy ART module is shown in Figure 2.14. The input field, F_0 stores the current input vector and transmits it to the matching field, F_1 which also receives top-down input from the choice field F_2; this latter field representing the active category assignment of the input data.

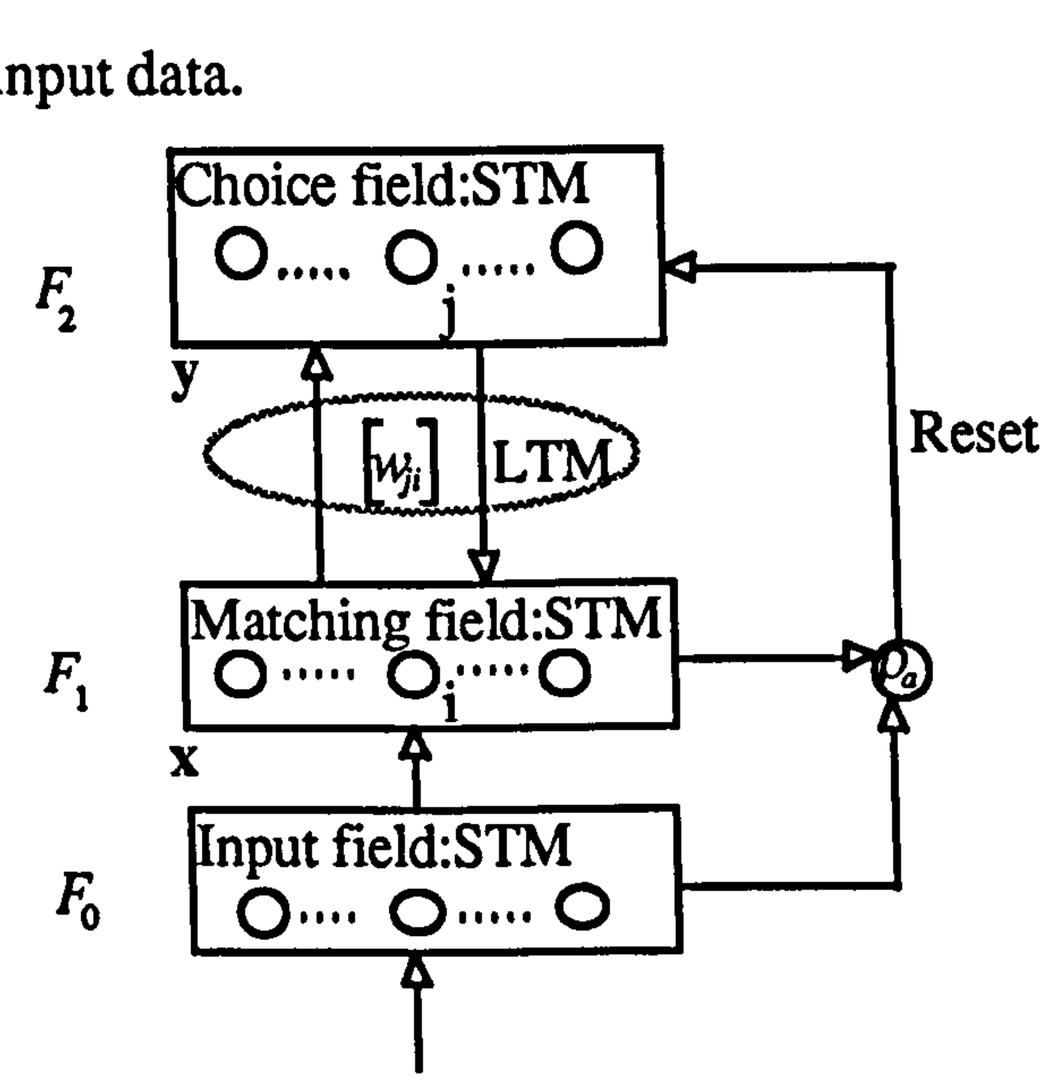


Figure 2.14. The fuzzy ART module. This figure illustrates the relationship between long term memory and short term memory and is identical to Figure 2.6 because fuzzy ART is a specific implementation of the functional (generic) description. Specific implementational details are given in the text. The Figure is reproduced here for convenience.

The F_0 activity vector is denoted by $\mathbf{I} = (I_1, \ldots, I_M), \quad I_i \in [0,1] \in \Re, \forall i = 1, \ldots, M$. The F_1 and F_2 activity vectors

are denoted by $\mathbf{x}=(x_1,\ldots x_M)$ and, $\mathbf{y}=(y,\ldots,y_N)$ respectively. Each F_2 node represents a class or category of inputs grouped together around an exemplar or prototype generated during the self-organising activity of the fuzzy ART module. Furthermore, each F_2 category node, j has its own set of adaptive weights stored in the form of a vector $\mathbf{w}_j=(w_{j1},w_{j2},\ldots,w_{jM}),\quad \forall j=1,\ldots,N.$

These weights represent the long term memory traces which evolve during network operation. The initial weight vector values are given by:

$$w_{ji}(0)=1,\quad \forall j=1,\ldots,N,\quad \forall i=1,\ldots,M.$$

With no categories being allocated to F_2 nodes at this stage, the nodes are said to be *uncommitted* (Carpenter *et al*, 1992). Once a category node is chosen to represent a category it then becomes *committed*. Unlike ARTMAP sub-systems (ART1 modules), the fuzzy ARTMAP components (fuzzy ART modules) differ in that the weight matrix $[w_{ji}]$ includes both top-down and bottom-up weight information.

2.4.2 Choice Field Activity

The choice field (F_2) nodes operate with winner-takes-all dynamics modelled by the F_2 output function (choice function)

$$T_j(\mathbf{I})=\frac{|\mathbf{I}\wedge\mathbf{w}_j|}{\alpha+|\mathbf{w}_j|},\quad \forall\mathbf{I}\in[0,1]^M, \tag{2.13}$$

where $\mathbf{I}$ is the given input vector, $\mathbf{w}_j$ is the j^{th} F_2 node weight vector, α is the *choice parameter* where $\alpha\cong 0$ in the case of fuzzy ART, $\wedge$, is the fuzzy AND operator so that $(\mathbf{p}\wedge\mathbf{q})_i\equiv\min_i(p_i,q_i)$, and the L^1 norm $|.|$ is defined by $|\mathbf{p}|=\sum_{i=1}^{M}|p_i|$.

This form of choice function given by equation (2.13) is the continuous-valued analogue of equation (2.10)

The overall F_2 winner, node J, is selected by $T_J = \max_j \{T_j : j = 1,\ldots,N\}$ to represent a category choice for a given input vector $\mathbf{I}$. $T_j(\mathbf{I})$ reflects the degree of match between the current input, $\mathbf{I}$ and the LTM of the j^{th} node, $\mathbf{w}_j$. The ratio, $0 \le \frac{|\mathbf{p} \wedge \mathbf{q}|}{|\mathbf{q}|} \le 1$, gives a measure of the fuzzy subsethood of $\mathbf{q}$ with respect to $\mathbf{p}$. The limit, $\frac{|\mathbf{p} \wedge \mathbf{q}|}{|\mathbf{q}|} = 1$ indicates that $\mathbf{q}$ is a fuzzy subset of $\mathbf{p}$.

Specifically, if $\frac{|\mathbf{I} \wedge \mathbf{w}_j|}{|\mathbf{w}_j|} = 1$, which occurs when $|\mathbf{I} \wedge \mathbf{w}_j| = |\mathbf{w}_j|$, then $\mathbf{w}_j$ is a fuzzy subset of $\mathbf{I}$. The greatest degree of match between input and weight vectors, for competing nodes, ensures selection as $\frac{|\mathbf{I} \wedge \mathbf{w}_j|}{|\mathbf{w}_j|} > \frac{|\mathbf{I} \wedge \mathbf{w}_k|}{|\mathbf{w}_k|}$ gives

$\frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} > \frac{|\mathbf{I} \wedge \mathbf{w}_k|}{\alpha + |\mathbf{w}_k|}$ and, thus, $T_j(\mathbf{I}) > T_k(\mathbf{I})$ as desired.

The choice parameter, α breaks the deadlock between competing nodes when $\mathbf{w}_j$ and $\mathbf{w}_k$ are both fuzzy subsets of $\mathbf{I}$, by selecting the node j such that $|\mathbf{w}_j| > |\mathbf{w}_k|$. This is because $T(\mathbf{I})$ is monotonically increasing so that, $|\mathbf{I} \wedge \mathbf{w}_j| = |\mathbf{w}_j|$ giving

$T_j(\mathbf{I}) = \frac{|\mathbf{w}_j|}{\alpha + |\mathbf{w}_j|}$. Thus for $|\mathbf{w}_j| > |\mathbf{w}_k|$, $T_j(\mathbf{I}) > T_k(\mathbf{I})$.

In the case that $T_j = T_k$ for some $j,k \le N$, such that $T_j, T_k > T_l \;\; \forall l \neq j,k$ the node with the lowest index is chosen.

Thus, the small value of the choice parameter is motivated by the mutual fulfilment of two constraints involving fuzzy subsethood and deadlock breaking.

2.4.3 Matching field Activity

The F_1 layer activity of fuzzy ART is analogous to that of ART 1 with equation (2.9) being replaced by

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

The match condition of equation (2.7) is replaced by

$$\frac{|\mathbf{I} \wedge \mathbf{w}|}{|\mathbf{I}|} \geq \rho, \qquad (2.14).$$

to ensure that the input vector belongs to the chosen category (Carpenter, Grossberg and Rosen, 1991).

This approach, with individual nodes representing categories, allows for dynamic adjustment of network size without disrupting previously acquired information as happens with, for example, feedforward networks. Extra nodes are simply assigned as and when required to represent new categories or pattern clusters. Both the fuzzy ARTMAP and the PROBART implementations discussed in this thesis use dynamic node allocation. However a fixed number of nodes can be allocated at the outset if desired.

2.4.4 Learning

Following a successful search, LTM changes are made according to

$$\mathbf{w}_J^{(new)} = \beta(\mathbf{I} \wedge \mathbf{w}_J^{(old)}) + (1-\beta)\mathbf{w}_J^{(old)} \qquad (2.15)$$

for the winning F_2 node, J. These changes correspond to the notion of learning. The learning rate parameter, β, with $0 \leq \beta \leq 1$ ensures that the new weight vector $\mathbf{w}_J$ is a convex combination of the resultant vector across F_1 and the F_2 layer expectation template. For $\beta = 1$, known as *Fast-Commit-Fast-Recode (FCFR)*, F_1 resultant vectors directly replace the present category exemplars.

An option, *Fast-Commit-Slow-Recode(FCSR)* , allows for initial fast learning prior to the convex combination learning rule of equation (2.15) by setting $\beta = 1$ for uncommitted nodes only (Carpenter, Grossberg and Rosen, 1991).. Thus, $\mathbf{w}_J^{(new)} = \mathbf{I}$ initially.

2.4.5 Complement Coding

According to Carpenter *et al,* (Carpenter, Grossberg and Reynolds 1991, Carpenter Grossberg and Rosen 1991, Carpenter *et al* 1992) normalisation of the input vectors is required to prevent category proliferation. In Carpenter Grossberg and Rosen (1991) it is proved geometrically that, without complement coding, the monotonically decreasing weight components would eventually result in many categories clustering near to the origin with others being created to replace them. For example, on the real line, when all categories to the left of an input value are inhibited, the first category to the right will be selected as any categories further to the right will result in a smaller activation value for the function $T(I)$. Furthermore, the condition of equation (2.14) is always fulfilled as $I < w_J$ gives $\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} = \frac{|\mathbf{I}|}{|\mathbf{I}|} = 1 \geq \rho$ An algebraic proof of category proliferation on the real line without complement coding (Marriott and Harrison, 1994) is given in Appendix D.

Normalisation is represented by $|\mathbf{I}| \equiv \gamma, \quad \forall \mathbf{I} \in [0,1]^M$ for some $\gamma > 0$. To achieve this for arbitrary $\mathbf{I} \in [0,1]^M$ take $\mathbf{I} = (\mathbf{a}, \mathbf{a}^c) \in [0,1]^{2M}$ where $\mathbf{a} \in [0,1]^M$ is the original input and $\mathbf{a}^c = \mathbf{1} - \mathbf{a}$ where $\mathbf{1} = (1,1,\ldots,1)$, and, $|\mathbf{1}| = M$.

Thus, the new F_0 layer input vector , $\mathbf{I}$ is complement coded and of dimension 2M with $|\mathbf{I}| = \gamma = M, \quad \forall \mathbf{I} \in [0,1]^{2M}$.

2.5 Fuzzy ARTMAP

For heteroassociative tasks, two connected fuzzy ART modules are required with each module receiving either the input (stimulus) or output (response) component of each pattern pair to be associated. Thus, the input and output spaces are organised into distinct categorised sets during processing.

2.5.1 Structure

The heteroassociative network discussed here is fuzzy ARTMAP (Carpenter *et al*, 1992) which uses a layer of nodes, called the map field, to link the two fuzzy ART modules. This configuration is illustrated in Figure 2.15. The main function of the map field is to associate compressed representations of the original pattern pair components (Carpenter, Grossberg and Reynolds, 1991; Carpenter *et al*, 1992)

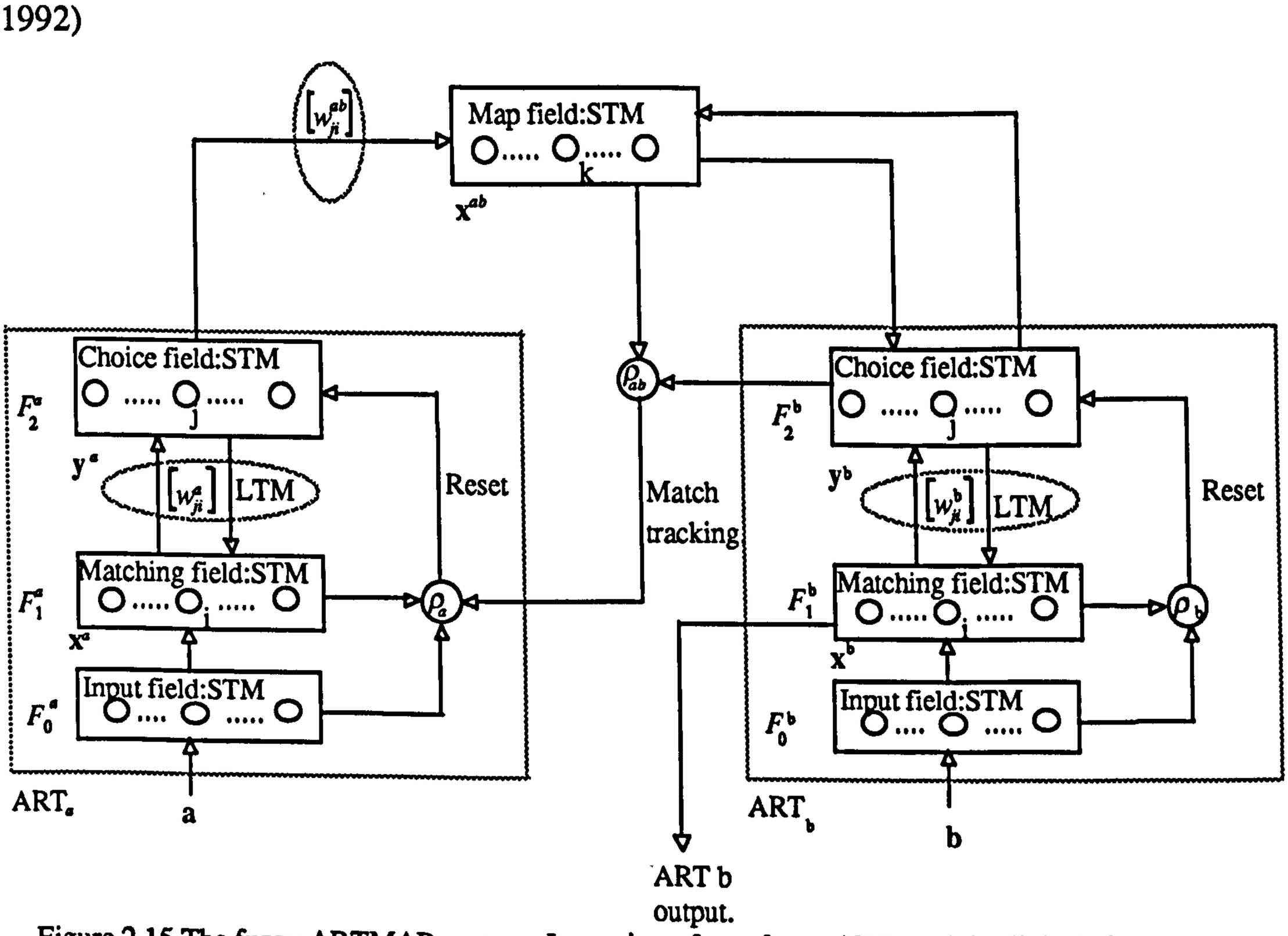


Figure 2.15 The fuzzy ARTMAP system. It consists of two fuzzy ART modules linked via a field of nodes called the map field (Carpenter *et al*, 1992).

The two fuzzy ART modules, ARTa and ARTb, accept inputs in complement coded form denoted by $\mathbf{I}_a = (\mathbf{a}, \mathbf{a}^c)$ and $\mathbf{I}_b = (\mathbf{b}, \mathbf{b}^c)$ respectively (Carpenter *et al*, 1992).

Following the convention of Carpenter *et al* (1992), the ARTa F_1 and F_2 layers are denoted by F_1^a and F_2^a respectively, with output vectors $\mathbf{x}^a = (x_1^a, \ldots, x_{2M_a}^a)$ and $\mathbf{y}^a = (y_1^a, \ldots, y_{N_a}^a)$ respectively. Let, $\mathbf{w}_j^a = (w_{j1}^a, w_{j2}^a, \ldots, w_{j,2M_a}^a)$ denote the j^{th} ARTa weight vector. Similarly, the F_1^b and F_2^b output vectors are denoted by $\mathbf{x}^b = (x_1^b, \ldots, x_{2M_b}^b)$ and $\mathbf{y}^b = (y_1^b, \ldots, y_{N_b}^b)$ respectively, and $\mathbf{w}_k^b = (w_{k1}^b, w_{k2}^b, \ldots, w_{k,2M_b}^b)$ denotes the k^{th} ARTb weight vector. The map field is denoted by F^{ab} with output vector $\mathbf{x}^{ab} = (x_1^{ab}, \ldots, x_{N_b}^{ab})$ and weight vector $\mathbf{w}_j^{ab} = (w_{j1}^{ab}, w_{j2}^{ab}, \ldots, w_{j,N_b}^{ab})$ for the j^{th} F_2^a node to F^{ab}.

Activity vectors are reset to zero between data presentations.

2.5.2 Map Field Activation

Map field activation is governed by both F_2^a and F_2^b activity in the following way:

$$\mathbf{x}^{ab} = \begin{cases} \mathbf{y}^b \wedge \mathbf{w}_J^{ab} & \text{if the } J^{th} \; F_2^a \text{ node is active and } F_2^b \text{ is active,} \\ \mathbf{w}_J^{ab} & \text{if the } J^{th} \; F_2^a \text{ node is active and } F_2^b \text{ is inactive,} \\ \mathbf{y}^b & \text{if } F_2^a \text{ is inactive and } F_2^b \text{ is active,} \\ 0 & \text{if } F_2^a \text{ is inactive and } F_2^b \text{ is inactive.} \end{cases} \tag{2.16}$$

which is analogous to the ARTMAP map field.

The four cases will be considered in order below.

i) F_2^a active and F_2^b active:

This corresponds to a pattern pair $(\mathbf{I}_a, \mathbf{I}_b)$ being present. $\mathbf{I}_a$ elicits an ARTa category selection with, say, the J^{th} F_2^a node winning the competition. This index, J will correspond to a weight vector, $\mathbf{w}_J^{ab}$ in the map field which links the F_2^a node with a predicted F_2^b layer activation. This predicted F_2^b node represents the ARTb category associated with the presently active ARTa category.

Simultaneously, the ARTb input, $\mathbf{I}_b$ has excited a category represented by the F_2^b output $\mathbf{y}^b = (\ldots,0,1,0,\ldots)$ with a 1 in the k^{th} position indicating node k is active. The fuzzy AND operation, $\mathbf{y}^b \wedge \mathbf{w}_J^{ab}$ ensures that the map field activity is non-zero only if the predicted and actual ARTb categories coincide (the kth category being predicted by ARTa) or if node J is uncommitted; all components of $\mathbf{w}_J^{ab}$ being equal to unity in the latter case.

ii) F_2^a active and F_2^b inactive.

This corresponds to prediction with $\mathbf{w}_J^{ab}$ representing the ARTb category associated with the currently active ARTa category. Heteroassociative mapping is achieved by working backwards within the ARTb module; the fuzzy ARTb weight vector associated with the predicted F_2^b node represents the expectation template fed down from F_2^b to F_1^b; this corresponds to the current exemplar for that ARTb category and, thus, the predicted output.

iii) F_2^a inactive and F_2^b active.

In this case only an ARTb input is present; thus, the map field activation represents the active ARTb category via the one-to-one relationship between the map field and ARTb.

iv) The final case represents the network in a quiescent state with no inputs impinging upon it.

2.5.3 Match-Tracking

The concept of vigilance is extended in fuzzy ARTMAP, analogous to ARTMAP, by allowing the ARTa vigilance parameter, ρ_a to vary whilst the ARTb vigilance parameter is fixed for a given training cycle. When an input is first presented, ρ_a is set to its baseline value, $\overline{\rho}_a$. Matching between ARTa and ARTb categories, again, depends upon the condition $\frac{|\mathbf{x}|}{|\mathbf{y}|} \geq \rho_{ab}$. If this is not fulfilled, i.e. the ARTa

category results in an incorrect prediction, match tracking activity is initiated where ρ_a is increased such that $\rho_a > \frac{|\mathbf{I}_a \wedge \mathbf{w}_J^a|}{|\mathbf{I}_a|}$ to prevent reselection of the J^{th} F_1^a node. Then the ARTa search cycle is carried out once more to select a new ARTa category which correctly predicts the current ARTb category. One of three conditions must occur to end the match tracking cycle: a matching ARTa category is selected from those already learned by ARTa, a new category is created (during training) or the condition $\rho_a > 1$ occurs which leads to shutdown of F_2^a until a new ARTa input becomes active.

2.5.4 Pattern Pair Association

Pattern pairs are associated via their compressed representations or category nodes. LTM information regarding inter-module F_2 node linkages is stored in the map field weight matrix which assigns a vector to each ARTa node reflecting the associated ARTb node.

A clearer idea of heteroassociative learning and prediction under FCSR is gained by considering the operation of fuzzy ARTMAP when presented with a previously unseen pattern pair which does not belong to any of the current categories. The pattern pair $(\mathbf{I}_a, \mathbf{I}_b)$ causes new categories J and K to be created in ARTa and ARTb respectively.

Initially, $w_{jk}^{ab}(0) = 1, \quad \forall j = 1,\ldots,N_a, \quad \forall k = 1,\ldots,N_b$. When resonance occurs, in which the J^{th} ARTa category becomes active, w_J^{ab} is set equal to $\mathbf{x}^{ab}$. The map field activation is given by $\mathbf{x}^{ab} = \mathbf{y}^b \wedge \mathbf{w}_J^{ab(old)} = \mathbf{y}^b$ (K^{th} vector entry = 1 only) as the J^{th} ARTa node is uncommitted (all entries =1). Map field learning requires $\mathbf{w}_J^{ab(new)} = \mathbf{x}^{ab}$ which gives $\mathbf{w}_J^{ab} = \mathbf{y}^b$

If $\mathbf{I}_a$ is presented alone, the J^{th} ARTa node is selected which predicts the K^{th} ARTb category through the J^{th} map field weight vector.

2.6 PROBART

As discussed in section 2.1 onwards, ART architectures have some interesting and useful properties. Some of these properties will be exploited in the novel architectures introduced in this thesis. However, the original formulation of ART has some drawbacks. The following list identifies the main ones:

- Both ARTMAP and fuzzy ARTMAP cannot deal with *one-to-many* mappings, that is, there can be no more that one output (output node) associated with a given input vector (input node). A way of allowing one-to-many mappings is developed within this section.

- Fuzzy ARTMAP suffers from *over-learning* and cannot easily distinguish between rapidly varying curves and noise because match tracking leads fuzzy ARTMAP to treat an incorrect prediction as a novel prediction requiring a new node instead of as a noisy input. Thus, ARTMAP tracks the noise and attempts to reproduce a noisy mapping exactly. The over-learning problem is illustrated in the simulations of section 2.7.

- Fuzzy ARTMAP does not *generalise* well when applied to mapping problems and fails to give a predicted output if an unknown input pattern is presented at recall. The novel architecture also suffers from this problem but possible modifications are suggested in the discussion. The lack of generalisation may appear to be a major drawback but is a trade-off for the added flexibility of the mapping. The highly localised construction of the mapping allows the addition and removal of nodes without disrupting the overall mapping. For some applications, a 'rough and ready' mapping is an acceptable trade-off. For classification problems, however, fuzzy ARTMAP generalisation is much better as illustrated in section 2.10

These points are intended to give an overview only. These and other issues will be discussed at greater length at appropriate points in the discussion of simulation results.

2.6.1 PROBART Structure

PROBART (Marriott and Harrison, 1995a) is the result of modifications to the fuzzy ARTMAP system motivated by empirical findings on the operational characteristics of fuzzy ARTMAP under certain conditions. A comparative analysis of fuzzy ARTMAP and PROBART operation is presented below. First, the fuzzy ARTMAP modifications incorporated into PROBART are described, and a description of PROBART operation is presented.

As with fuzzy ARTMAP, PROBART uses a pair of fuzzy ART modules linked by a map field; this is where the similarity ends owing to different map field dynamics. The inputs are again accepted in complement coded form. The notation introduced above in the sections describing fuzzy ARTMAP is retained in the description of PROBART. Exceptions are described where appropriate.

2.6.2 Map Field Activation

In PROBART equation (2.16) is replaced by

$$\mathbf{x}^{ab} = \begin{cases} \mathbf{y}^b + \mathbf{w}_J^{ab} & \text{if the } J^{th}\ F_2^a \text{ node is active and } F_2^b \text{ is active,} \\ \mathbf{w}_J^{ab} & \text{if the } J^{th}\ F_2^a \text{ node is active and } F_2^b \text{ is inactive,} \\ \mathbf{y}^b & \text{if } F_2^a \text{ is inactive and } F_b^2 \text{ is active,} \\ 0 & \text{if } F_2^a \text{ is inactive and } F_2^b \text{ is inactive.} \end{cases} \tag{2.17}$$

in which the fuzzy AND operation ($\wedge$) is replaced by vector addition (+). As will become apparent, this allows the nodal association frequency counts maintained in LTM to be incremented.

Before interpreting equation (2.17) it is important to realise that the map field weight matrix now contains information about the frequency with which pairs of ARTa and ARTb categories are associated e.g. $w_{jk}^{ab} = f, \quad f \in \mathrm{N}$, where N is the set of natural numbers. This indicates that the j^{th} ARTa node has been associated with the k^{th} ARTb node f times during the training phase.

Initial map field weight values are given by

$$w_{jk}^{ab}(0) = 0 \quad \forall j = 1,\ldots,N_a, \quad \forall k = 1,\ldots,N_b.$$

The four cases of equation (2.17) are analogous to those given in equation (2.16)

.

i) F_2^a active and F_2^b active:

As with fuzzy ARTMAP, the pattern pair $(\mathbf{I}_a, \mathbf{I}_b)$ results in selection of the J^{th} ARTa category and the K^{th} ARTb category. The vector $\mathbf{y}^b$ is, again, a unit vector with the K^{th} entry equal to one. The vector $\mathbf{x}^{ab}$ now represents the updated frequency distribution of node associations between the J^{th} ARTa node and nodes in the ARTb F_2^b layer; the map field weight matrix entry w_{JK}^{ab} being incremented by one, reflecting the new association.

ii) F_2^a active and F_2^b inactive.

Analogous to fuzzy ARTMAP, this corresponds to prediction but care has to be taken to determine in which sense the prediction is made. The implementation of PROBART discussed in this paper uses a weighted average given by

$$\mu_{Jm} = \frac{1}{\left|\mathbf{w}_J^{ab}\right|} \sum_{n=1}^{N_b} \varepsilon_{nm} w_{Jn}^{ab}, \quad m = 1,\ldots,2M_b \tag{2.18}$$

where μ_{Jm} is the expected value (mean) of the m^{th} component of the predicted output pattern associated with the J^{th} ARTa node, $\left|\mathbf{w}_J^{ab}\right|$ is the total number of associations of ARTb nodes with the J^{th} ARTa node, ε_{nm} is the m^{th} component of the n^{th} ARTb category exemplar and w_{Jn}^{ab} is the frequency of association between the n^{th} ARTb node and the J^{th} ARTa node. Other possible prediction measures can be used. These include: choosing the exemplar with the highest frequency, giving relative association frequency information, and using alternative higher order moments. The predicted ARTb output vector is denoted by $\mu_J = (\mu_{J1},\ldots,\mu_{J,M_b})$. Note that only the first M_b components which are not complement coded are meaningful and correspond to the original pattern pair data, with $\hat{\mathbf{b}} = \mu_J$ being an estimate of the true output $\mathbf{b}$ associated with input pattern $\mathbf{a}$.

Equation (2.18) can also be written as $\mu_{Jm} = \sum_{n=1}^{Nb} \varepsilon_{nm} p_{Jn}$ where p_{Jn} is the empirically estimated probability of association between the J^{th} ARTa node and the n^{th} ARTb node given by $p_{Jn} = \frac{w_{Jn}^{ab}}{|\mathbf{w}_J^{ab}|}$

Conditions iii) and iv) are identical to those in fuzzy ARTMAP.

2.6.3 Learning

As with fuzzy ARTMAP $\mathbf{w}_J^{ab(new)} = \mathbf{x}^{ab}$ but note that there is now no match tracking. The ARTa vigilance parameter, ρ_a, is held constant to maintain fixed category sizes. This is to prevent corruption of frequency information which would occur if category sizes were variable. How would association frequencies pertaining to a single category be apportioned to its eventual sub-categories when associations have previously been recorded in relation to the original, coarser category? This is similar to the problem of retrieving finer scale information from a coarser grey scale image in which information has been discarded. Further details will be found in the general discussion section below. Without match tracking, supervised associations are not judged to be correct or incorrect but recorded as they occur; training values distorted by noise are not associated with higher vigilance sub-categories within ARTa. More frequent associations are more heavily weighted in prediction mode. Note that the map field vigilance parameter, ρ_{ab} is not required for PROBART. This allows for one-to-many mapping between ARTa and ARTb categories which may be important in situations where more than one action results from a single stimulus or input. The relative importance of the ARTb categories associated with a single ARTa category are reflected in the map field frequencies.

Carpenter and Tan (1993) modify the map field of fuzzy ARTMAP to give estimates of the probability of whether or not an input belongs to a particular category. This is achieved by using a map field learning parameter to govern the rate of change of map field weights. In the slow-learning mode, the current map field weight vector and current map field activation vector are combined to give the new map field weight vector.

Although not investigated in this thesis, as with fuzzy ARTMAP, it is possible for PROBART to be operated in an on-line mode and in a non-stationary information environment. In the latter case, node association frequencies would change concomitantly with changes in underlying trends.

2.7 Simulations

2.7.1 The Mapping Task

A continuous non-linear signal was used for comparison of fuzzy ARTMAP and PROBART performance (Marriott and Harrison, 1995a):

$f:[0,1] \subset \Re \rightarrow [0,1] \subset \Re$ with,

$f(x) = (\sin(10x) + \sin(20x) + \sin(30x) + \sin(40x) + \sin(50x) + \sin(60x) + \sin(70x) + 10)/20,$
and x in radians. See Figure 2.16.

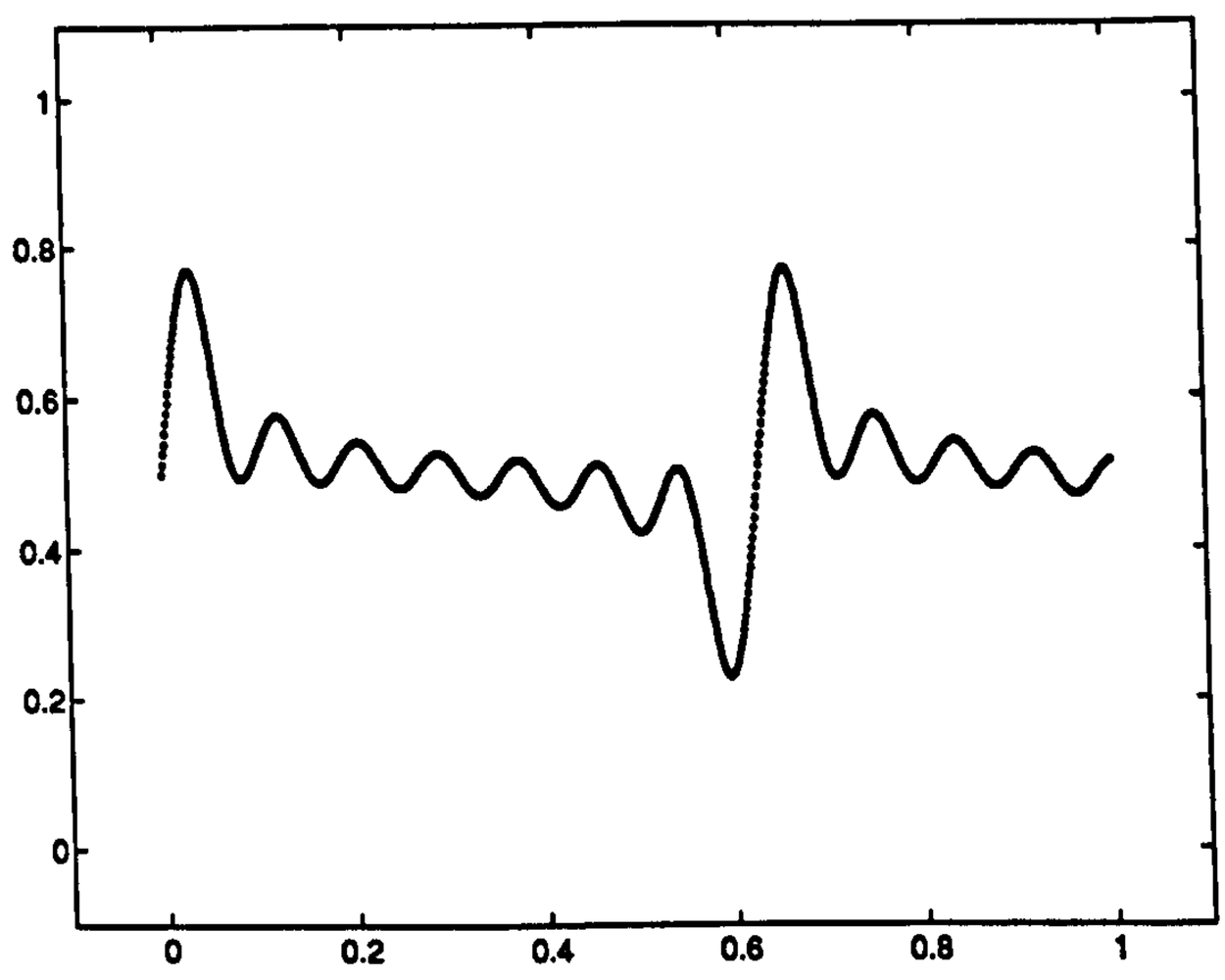


Figure 2.16. Noise-free test signal used in the evaluation of fuzzy ARTMAP and PROBART performance.

The range of the test function $f:\Re \rightarrow \Re$ is [0.2295, 0.7705] for the input domain [0,1]. Gaussian noise, derived from a zero mean source with unit variance, is added to the signal with a scale factor of 0.02. Thus, the corrupted output signal for pattern pair p is given by $y_p = y(p) = f(x_p) + 0.02\varepsilon_p$, where $\varepsilon_p \sim N(0,1)$ is the Gaussian noise, for the p th pattern pair and x_p is the x-coordinate of this pair. The x-coordinates were randomly chosen from a uniformly distributed source. The training and testing files were generated with different sets of x-coordinates unless otherwise stated. The testing sets being noise-free coordinates, or pattern

pairs, (x_p, y_p) chosen at random from the test curve, $f(x)$. For all experiments the choice parameter, $\alpha = 0.001$ and the learning mode chosen was FCFR unless otherwise stated.

2.7.2 Performance Measures

Performance is judged by both the root mean square error (RMSE) and maximum absolute error (MAXAE) measures. The RMSE value is computed by

$$RMSE = \sqrt{\frac{1}{N}\sum_{p=1}^{N}\left\|\mathbf{d}_p - \mathbf{y}_p\right\|^2},$$

where $\mathbf{d}_p$ is the desired output for pattern p, $\mathbf{y}_p$ is the actual output and N is the number of patterns used for training or testing.

In the following tables, TR denotes the noisy training set, TE(NF) denotes the noise-free test set using the same x-coordinates as the noisy training set, and TE denotes the noise-free test set selected using a different x-coordinate sample. The purpose of TE is to test the generalisation of the mapping.

As a further illustration of network performance, the error profile is plotted below the actual network output signal. RMSE and MAXAE error measures alone are very coarse indicators of network performance, especially when applied over the whole curve. Error profiles provide more detailed information in a visual form. For the simulations described below, example results are given in the text, and mean results, together with their respective error bounds, will be found in appendix E.

It will be shown that PROBART requires fewer nodes than fuzzy ARTMAP to achieve comparable performance on a noisy mapping task. Also included is a comparison of fuzzy ARTMAP and PROBART performance when both architectures are applied to a classification problem; this illustrates the purpose of match tracking in fuzzy ARTMAP.

2.7.3 Simulation 1

Fuzzy ARTMAP was trained on both noise-free and noisy data. Its parameters were set as follows: $\alpha = 0.001, \rho_a = 0.99, \rho_b = 0.99$ and $\rho_{ab} = 0.9$. Both the training and test sets consisted of 1,000 data pairs. Typical results for the training signal without noise are shown in Table 2.2.

No of categories.		Error measures	
ARTa	ARTb	RMSE	MAXAE
312	53	0.0074	0.01

Table 2.2. An example of fuzzy ARTMAP performance with noise-free training and test data. Training is off-line using 1,000 pattern pairs.

For the typical results, only a single training epoch was required for fuzzy ARTMAP to acquire an internal representation of the test mapping signal with the RMSE ranging between approximately 1% of the input signal at its maximum point to approximately 3% at its minimum point. This is shown in Figure 2.17.

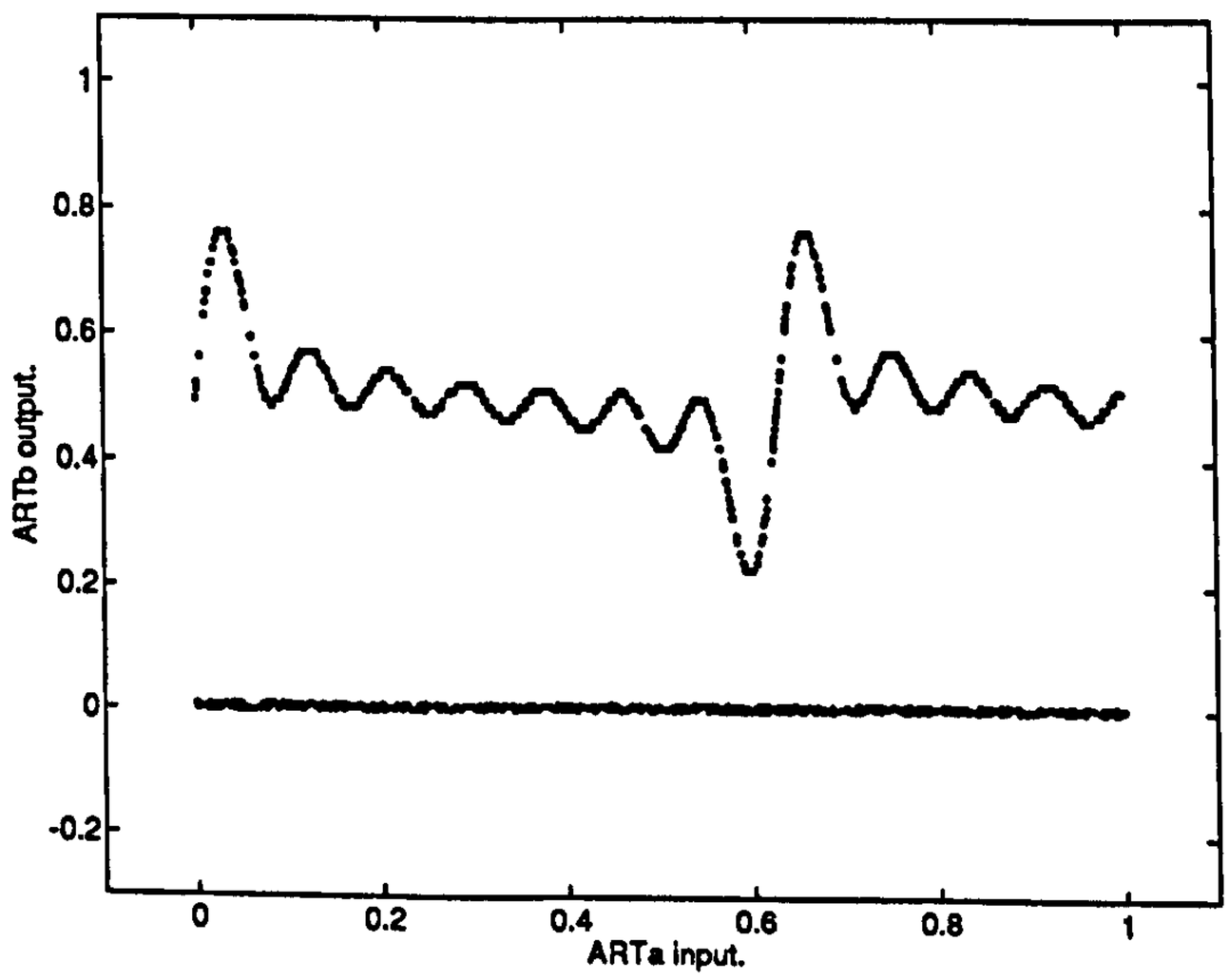


Figure 2.17. Fuzzy ARTMAP performance with noise-free data. The network has been both trained and tested using the same noise-free data file. The lower section of the graph shows the error profile.

Note the uniformity of the error profile which attains an absolute maximum range of only 4.4% to 1.3% of the test signal at its maximum and minimum points respectively.

The effect of match tracking on the fuzzy ARTa module is immediately apparent from the distribution of category numbers between the two modules in Table 2.2. Taking the ratio of the total input signal range (1.0) to the total output signal range (0.541) predicts a category ratio of approximately 2:1 for the ARTa and ARTb modules respectively. This ratio assumes that both modules have the same vigilance parameters and, hence, the same input resolution or category sizes. At the beginning of each training pattern pair presentation the condition $\rho_a = \rho_b$ is fulfilled. For the typical results of Table 2.2., match tracking has increased the ratio to about 6:1 by reducing category sizes through increased vigilance in order to resolve sub-categories. Data compression of approximately 3.3 data points per category node is achieved.

Typical results for the training signal with noise are shown in Table 2.3.

		Error measures					
No. of categories		RMSE			MAXAE		
ARTa	ARTb	TR	TE(NF)	TE	TR	TE(NF)	TE
806	61	0.0137	0.0302	0.0302	0.0878	0.0678	0.0679

Table 2.3. Typical results for fuzzy ARTMAP trained using a noisy data file of 1,000 items and tested using a noise-free data file also consisting of 1,000 data items.

When fuzzy ARTMAP is trained with pattern pairs derived from the input signal of Figure 2.17, but this time distorted by noise, two training epochs are required to obtain the lowest training RMSE value. Both training epochs consist of presenting the pattern pairs and adjusting the network weights after each individual presentation on the basis of erroneous predictions. A single training epoch requires that the whole training file be processed in this way. Following training, the training file is used purely as a test file (with the learning mode disabled) to assess the current learning progress. The disabling action prevents further learning from taking place during testing. The typical results of Table 2.3 are illustrated in figure 2.18.

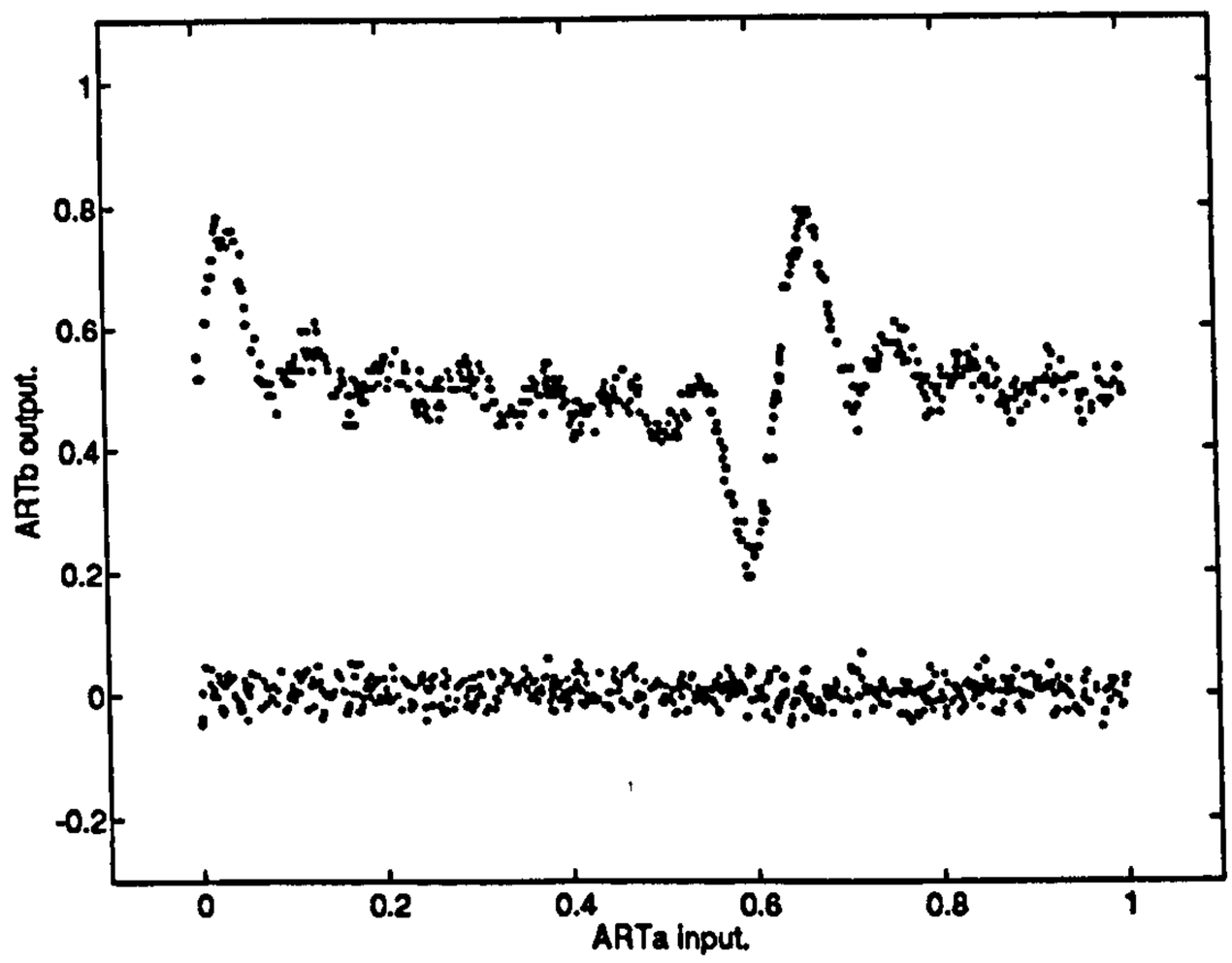


Figure 2.18. Fuzzy ARTMAP performance with noisy data. The network output and corresponding error profile illustrate the results shown in Table 2.3.

The error profile, coupled with the number of ARTa categories, indicates that each disturbance is being faithfully recorded on an almost individual basis. Its characteristics do not vary across the input domain. Thus, it appears that the source of error has not been effectively filtered or altered. FCFR results ($\beta = 1$) are quoted because both the RMSE and MAXAE measures did not vary greatly for values of β in the range 0.1 to 1. Variation of β, using FCSR, did not appear to effect noise suppression through equation (2.15) with the maximum measured difference between training RMSE values for this data set being approximately 4% of the lowest value. This apparent insensitivity to β was consistently observed and was the result of the high vigilance values confining categories within narrow ranges. This situation is depicted graphically in Figure 2.19.

Using FCSR and reducing the vigilance values of the ARTa and ARTb modules to increase the effect of β in equation (2.15) was found to be counter-productive. For example, reducing ρ_a and ρ_b to 0.9 increased the testing RMSE (TE) by a factor of approximately 2.5 for β in the range 0.1 to 1.0. The numbers of ARTa and ARTb categories were reduced concomitantly with their increased coverage of regions of input and output space but without any corresponding increase in generalisation.

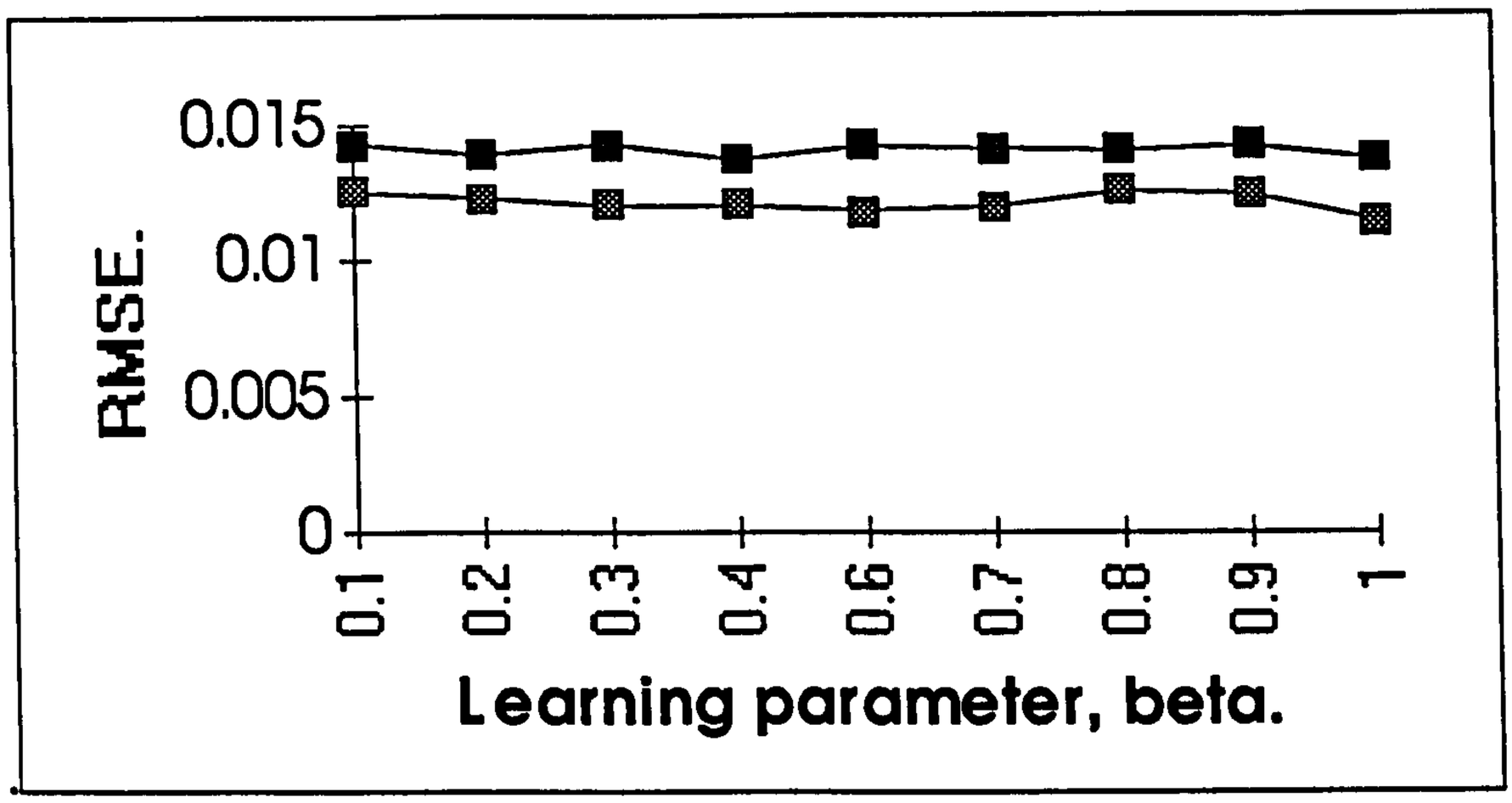


Figure 2.19. Plot of RMSE against β for two different runs with $\rho_a = \rho_b = 0.99$ illustrating the lack of effectiveness of β in reducing noise.

Note the significant increase, when comparing Tables 2.2. and 2.3., in the number of ARTa categories required to represent the noisy mapping while the number of ARTb categories did not increase unduly The latter increase reflects an extended ARTb input range as a consequence of noise. The large number of ARTa categories did not reduce when β was varied using FCSR. The mean ratio of approximately 1.25 data points per ARTa category (Appendix E. Table E1.4) indicates that fuzzy ARTMAP appears to be learning the noisy signal in contrast to the underlying mapping. This observation is further confirmed by the RMSE results for the training data set, with the mean noise-free testing RMSE value (TE(NF)) being greater than twice that of the mean noisy training RMSE (TR) (Appendix E. Table E1.4) after training fuzzy ARTMAP on noisy data. However, this example must not be taken to indicate poor performance by the network in general. The data here is highly disorganised, having no clusters, while fuzzy ARTMAP performs best with clustered data. Match tracking allows sub-clusters to be resolved in classification problems by varying the ARTa vigilance parameter during learning, but this enhanced performance mechanism becomes a disadvantage in highly disorganised data sets such as those used here. To understand the operation of match tracking under these circumstances, refer to Figure 2.20 (a) where, for clustered data, the category delimited by ρ_{a1} maps to an ARTa node and via the map field to, say, ARTb category 1 (class 1). If data is

found within the ARTa node category which does not map to category 1, match tracking increases ARTa vigilance to $\rho_{a2} > \rho_{a1}$ which leads to the activation or formation of a sub-category capable of being associated with ARTb category 2. This mechanism is suited to classification problems. Thus, sub-categories are formed which allow learning of infrequent but perhaps significant features which may be ignored or averaged out by other architectures including PROBART.

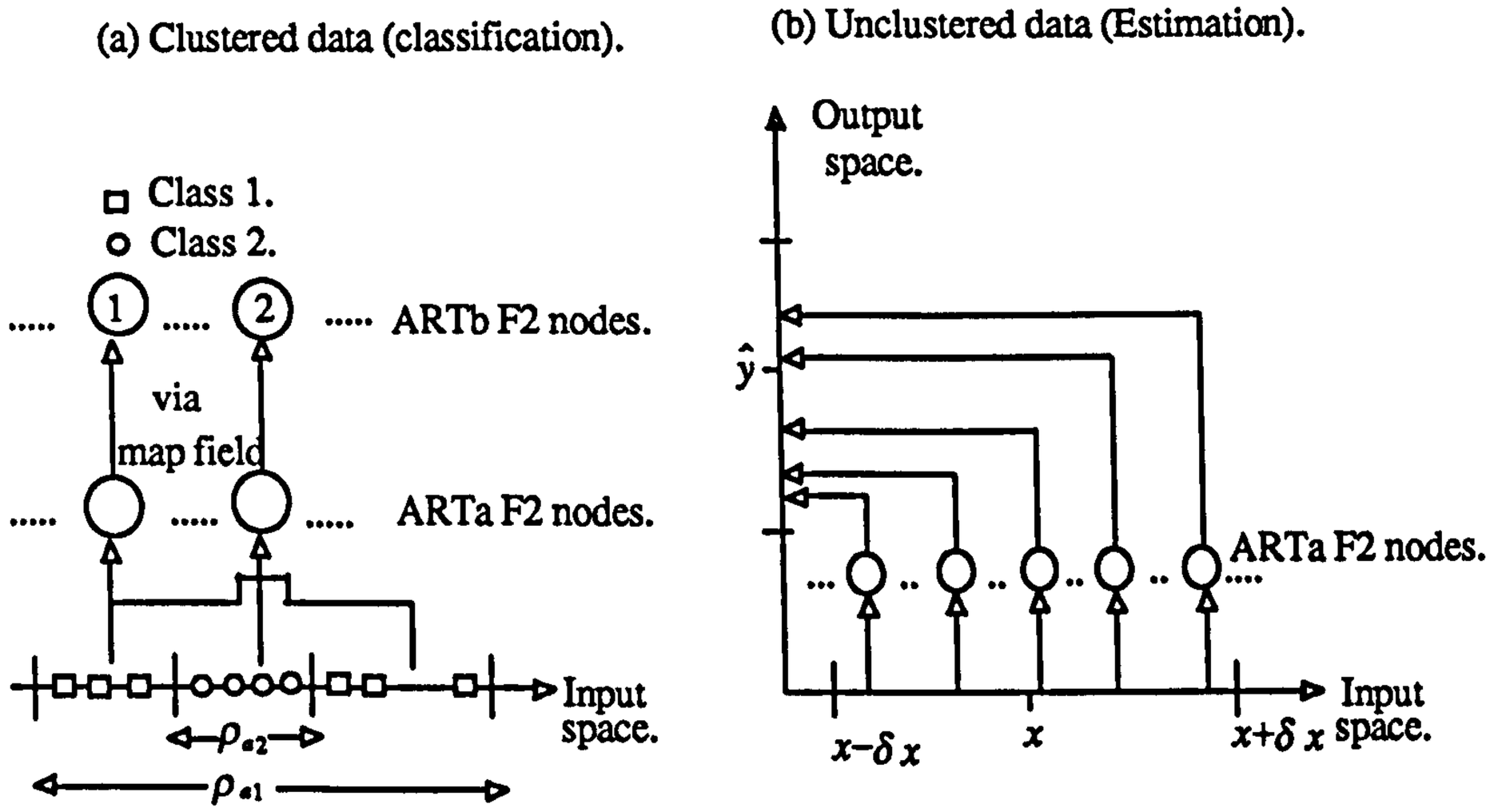


Figure 2.20. Comparison of classification and estimation modes. (a) highly organised data leads to the establishment of distinct categories. (b) disorganised data, i.e. not belonging to discrete categories, found within a $\delta-$neighbourhood centred around an input value, leads to an output estimate corresponding to that input value.

With unclustered data deviations in ARTb values are treated as novel features and new ARTa sub-categories are created individually to encompass many of the data points (see Figure 2.20(b)). Thus, a small subset of the input space may be mapped to a larger range of output space determined by the noise which is treated as a multitude of predicted output classes. Ideally, the range of output space would be transformed to provide an estimated output which the given input range $x \pm \delta x$ would map to, but this does not happen. In other words, fuzzy ARTMAP does not map an input belonging to the $\delta-$neighbourhood of x to an estimate $\hat{y}_1$. It creates a sub-category for such inputs and individually maps them to the noisy outputs with which they are associated during training.

2.7.4 Simulation 2

PROBART was trained on the same sets of noisy and noise-free data used in simulation 1. The parameters: $\alpha = 0.001, \rho_a = 0.99, \rho_b = 0.99$ are set identically to those in the previous experiment wherever possible. The map field vigilance does not exist in PROBART as match tracking has been removed. Typical results for the training signal without noise are shown in Table 2.4.

No. of categories.		Error measures	
ARTa	ARTb	RMSE	MAXAE
110	53	0.0169	0.0755

Table 2.4. PROBART performance with noise-free training and test data. Training is off-line using 1,000 pattern pairs identical to those used in producing the results shown in Table 2.2.

Figure 2.21 illustrates the performance of PROBART with noise-free data after a single epoch (typical results).

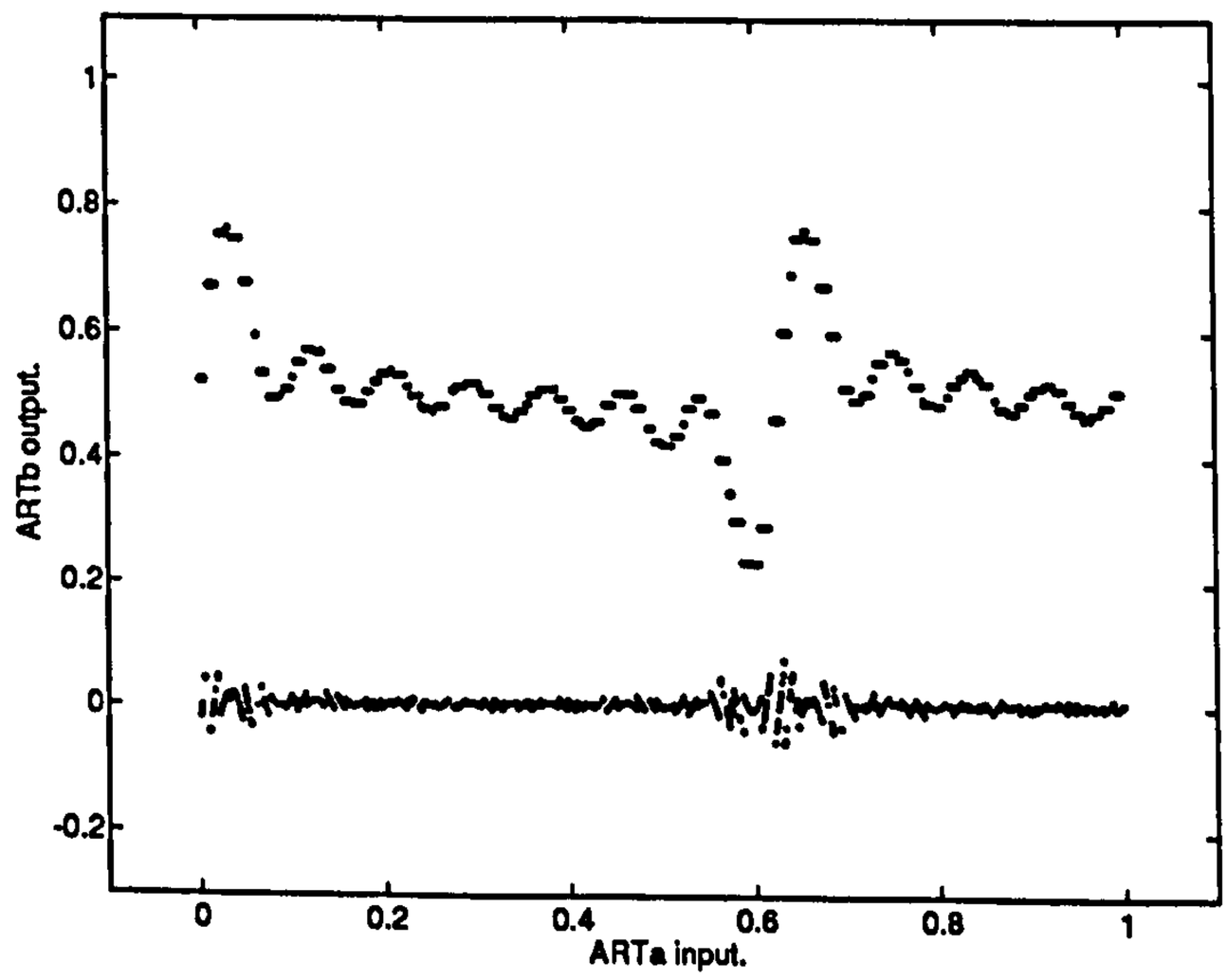


Figure 2.21. PROBART performance with noise-free data. As with fuzzy ARTMAP, the network has been both trained and tested using the same noise-free data file.

Note the different error profile when Figure 2.21 is compared with Figure 2.17. The former is not uniform, exhibits structural properties and is considerably larger in magnitude at some points, notably where large increases in signal slope occur.

As will become apparent, this is a consequence of the trade-off between plasticity and stability. When match tracking is removed, sensitivity to rapidly fluctuating noise signals is greatly reduced as ARTa sub-categories are not created to represent the noisy associations. However, this fixed quantization of the input domain leads to inaccuracies in signal representation. The relative importance of these inaccuracies, compared to overall noise reduction with noisy signals and increased generalisation, depends upon the application. Typical results for the training signal with noise are shown in Table 2.5.

		Error measures					
No. of categories		RMSE			MAXAE		
ARTa	ARTb	TR	TE(NF)	TE	TR	TE(NF)	TE
112	61	0.0322	0.0189	0.0202	0.1057	0.0769	0.0905

Table 2.5. Typical results for PROBART trained and tested using the data files of simulation 1 which generated the results of Table 2.5.

Figure 2.22 shows the typical results of this simulation after a single training epoch.

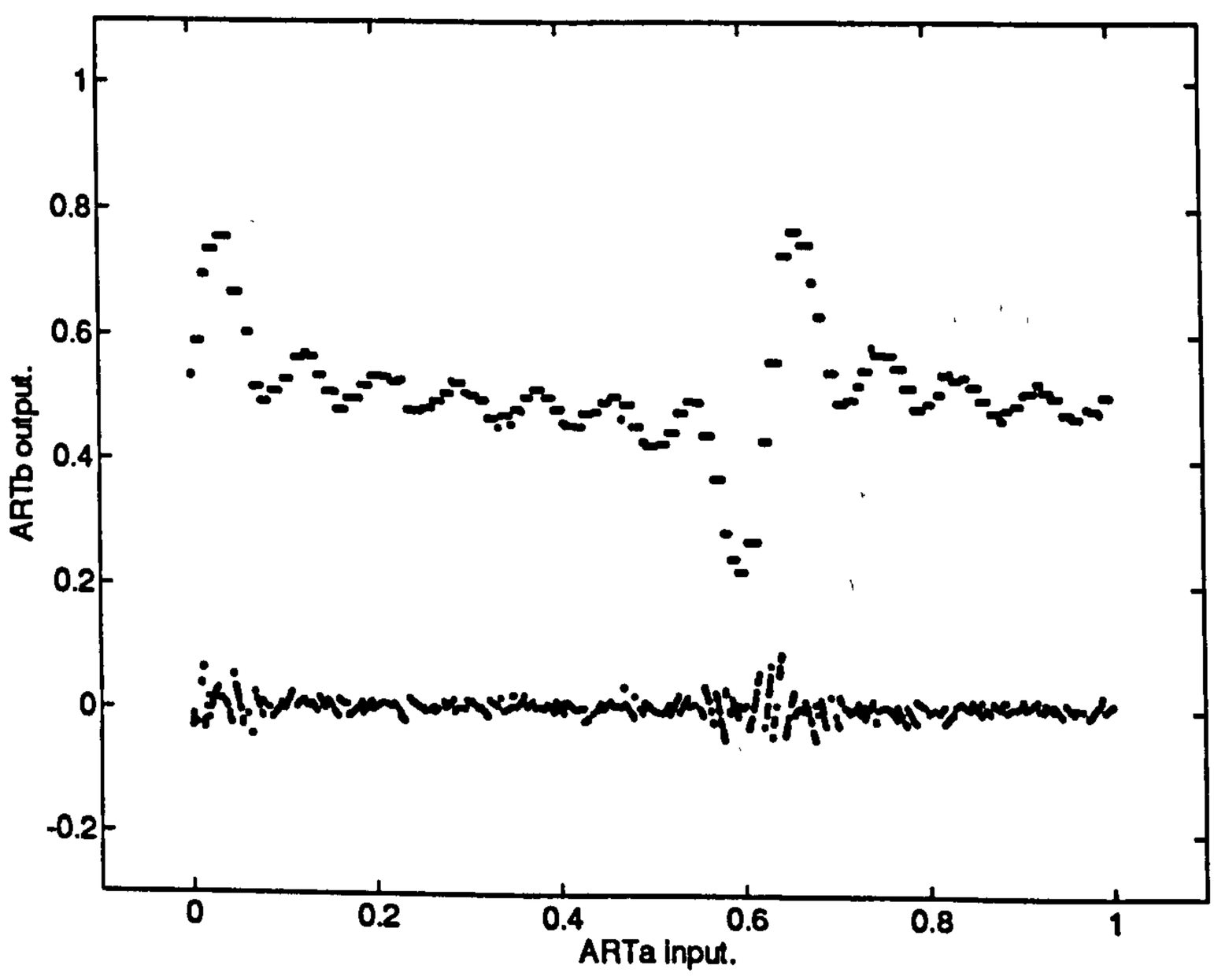


Figure 2.22. PROBART performance with noisy data. The network output and corresponding error profile illustrate the results shown in Table 2.2.2.4. Note that PROBART carries out a single training epoch only, when learning a mapping.

FCFR results are, again, quoted with only a 5% maximum variation from the lowest training RMSE value for $0.1 \leq \beta \leq 1$ using FCSR.

The predicted category ratio of 2:1 for the number of ARTa nodes compared to the number of ARTb nodes is reflected in both Table 2.4 and Table 2.5. Again, the increase in ARTb nodes is a consequence of output range extension by the additive Gaussian noise.

The mean ratio of 9.0 data points per ARTa category indicates that PROBART uses a coarser partitioning of the input space than that generated by fuzzy ARTMAP to represent the function/mapping domain. This reduction in categories results from the use of a fixed ARTa vigilance which, unlike fuzzy ARTMAP, does not allow subdivision of existing categories. In mapping problems this data compression is desirable to prevent the network from degenerating into a 'look-up' table and, thus, being incapable of generalisation. Observe in Table E2.4. of Appendix E that the mean noise-free test RMSE value (TE(NF)) is lower than the mean noisy training RMSE value (TR) (both sets of data used as test data following training with noisy data). As expected, this indicates that the opposite effect to that observed in fuzzy ARTMAP simulations is taking place. PROBART tends to learn the underlying signal which is, of course, the objective here.

The larger mean RMSE of PROBART (Table E2.1. in Appendix E) for the noise-free training/testing data set compared to that exhibited by fuzzy ARTMAP (Table E1.1. in Appendix E) results from the fixed vigilance which limits the input domain partitioning. The reduction in resolution in rapidly changing signal regions (increasing gradient) is apparent from Figures 2.21 and 2.22, both in the actual output signals and in the error profiles. Thus, prediction errors are increased in those subsets of the input domain where small ARTa inter-category distances give rise to larger ARTb inter-category distances in the function range. These errors, unrelated to noise, account for a sizeable proportion of the RMSE value in PROBART simulations trained with a noisy data set.

Comparison of Tables E2.4 and E1.4 in Appendix E reveals that PROBART reduces the mean RMSE value for the noise-free test set to 67% of the value for

fuzzy ARTMAP. This gain in performance is considerably enhanced when comparing the number of ARTa categories generated by both systems. PROBART has achieved generalisation, using approximately one seventh of the number of ARTa category nodes required by fuzzy ARTMAP.

To investigate the gradient/error relationship further, an experiment was performed using a straight line as the training function, where the gradient was varied in the range 1.0-10.0 for a fixed vigilance of 0.99 at fixed intercepts The results of a single experiment consisting of 5 runs of the same noise-free training file using different gradients is shown in Figure 2.23. The test file used was identical to the training file to eliminate the introduction of errors related to the use of different x-coordinate values.

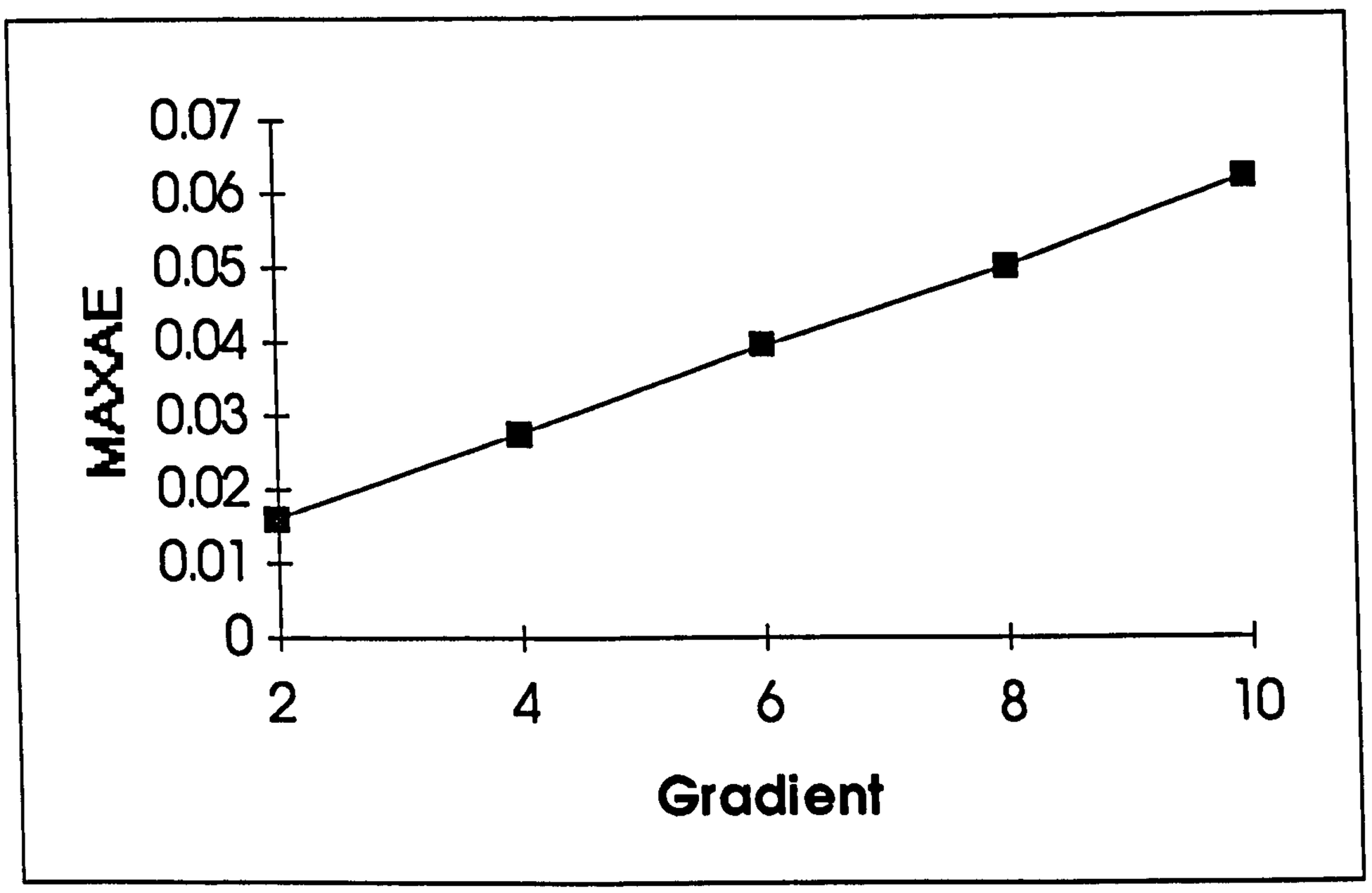


Figure 2.23. Plot of maximum absolute error vs. gradient for PROBART using a noise-free straight line training function. The relationship shows that intrinsic (non-noise related) errors increase with increasing function gradient. See text for discussion.

Note the linear relationship between the maximum absolute error and the gradient confirming that, as expected, rapidly changing signal regions decrease predictive accuracy. This linearity was consistently observed. Thus, signal quantization, resulting from the use of fixed vigilance parameters, introduces inaccuracies which can only be removed by increasing system vigilance to provide finer coverage of the input (stimulus) space and output (response) space. Reduction of the

quantization interval size is used to compensate for the removal of match tracking. The effect of increasing both the ARTa and ARTb vigilance parameters to increase signal resolution was investigated in the following simulations.

2.7.5 Simulation 3

PROBART was trained using the same noise-free data and value of α of simulation 2 but with increased vigilance parameters: $\rho_a = 0.999, \rho_b = 0.999$. Again, the test file was identical to the noise-free training file and consisted of 1,000 coordinate pairs. An example of typical results are shown in Table 2.6.

No. of categories.		Error measures	
ARTa	ARTb	RMSE	MAXAE
499	243	0.0016	0.0084

Table 2.6. Typical results obtained from PROBART using increased vigilance ($\rho_a = 0.999, \rho_b = 0.999$). Both training and testing were carried out using a noise-free data file consisting of 1,000 pattern pairs.

These results are illustrated in Figure 2.24.

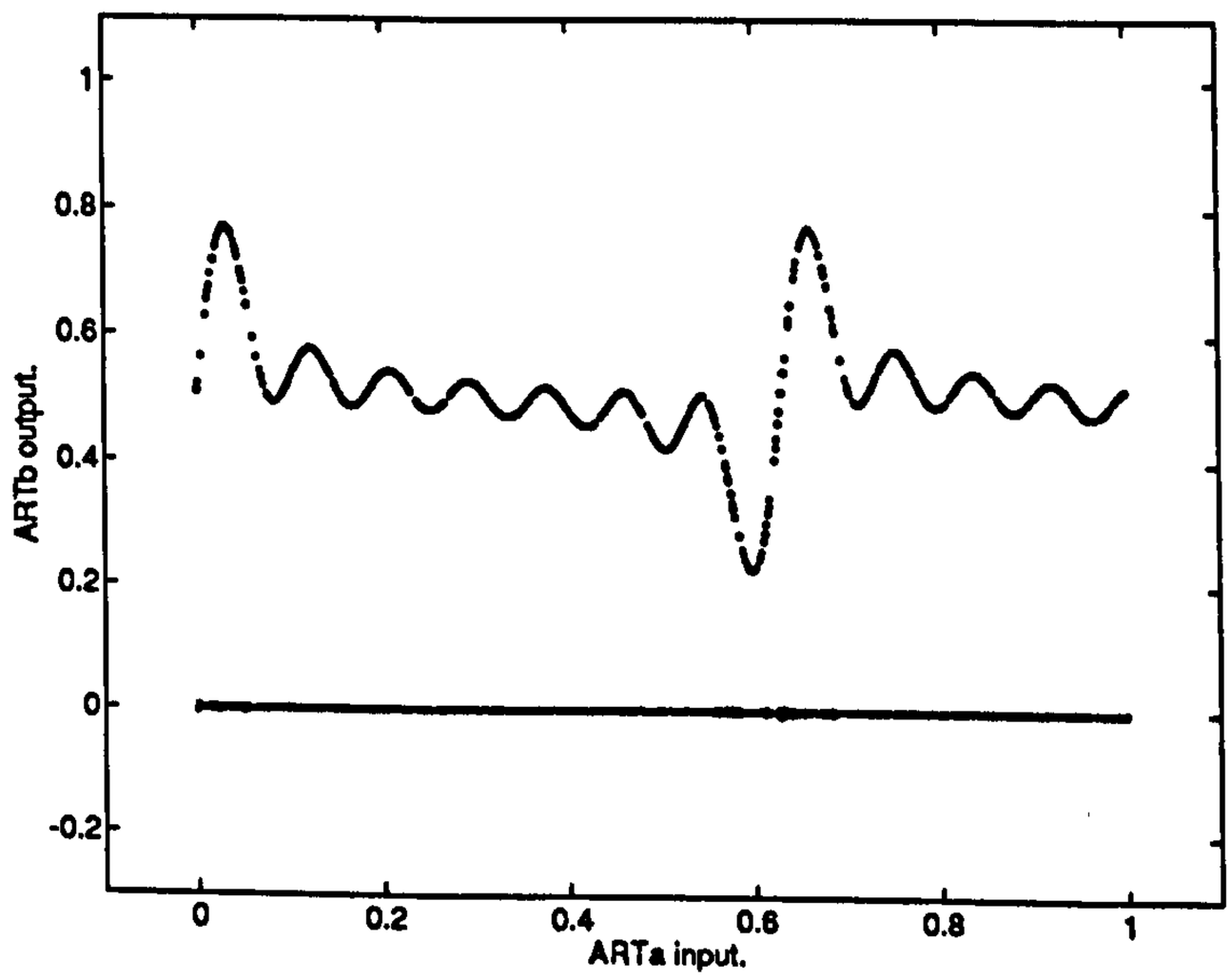


Figure 2.24. PROBART performance with noise-free data illustrating the effect of increased vigilance on the error profile.

Note the improvement in the error profile over that of Figure 2.21. Disturbances in the profile in areas of rapid signal change have been greatly reduced. Compared with the mean noise-free results of simulation 2 (Appendix E, Table E2.1), both the ARTa and the ARTb modules have shown an approximately five-fold increase in the mean number of category nodes (Appendix E, Table E3.1). These increases are reflected in the reduction of both mean error measures to about 10% of the previous values. Thus, the signal has been represented more accurately but at the expense of an increase in overall network size. Again, varying β (using FCSR) made very little difference, producing less than 10% maximum variation in the range of RMSE values for the typical results quoted. However, the benefits of simply increasing input/output space resolution are not realised when noisy training data is used as the following simulation illustrates.

2.7.6 Simulation 4

PROBART was trained with the noisy data set used previously in simulations 1 and 2 with parameters set as for simulation 3. Table 2.7 summarises the results of an example run.

		Error measures					
No. of categories		RMSE			MAXAE		
ARTa	ARTb	TR	TE(NF)	TE	TR	TE(NF)	TE
504	277	0.0208	0.0196	0.0192	0.0527	0.0544	0.0545

Table 2.7. Typical results for PROBART trained using a noisy data file of 1,000 items and tested using a noise-free data file also consisting of 1,000 data items. The increased vigilance values of $\rho_a = 0.999, \rho_b = 0.999$ leads to an expected increase in the number of category nodes in both the ARTa and ARTb modules.

The results of the typical run summarised in Table 2.7 are illustrated in Figure 2.25.

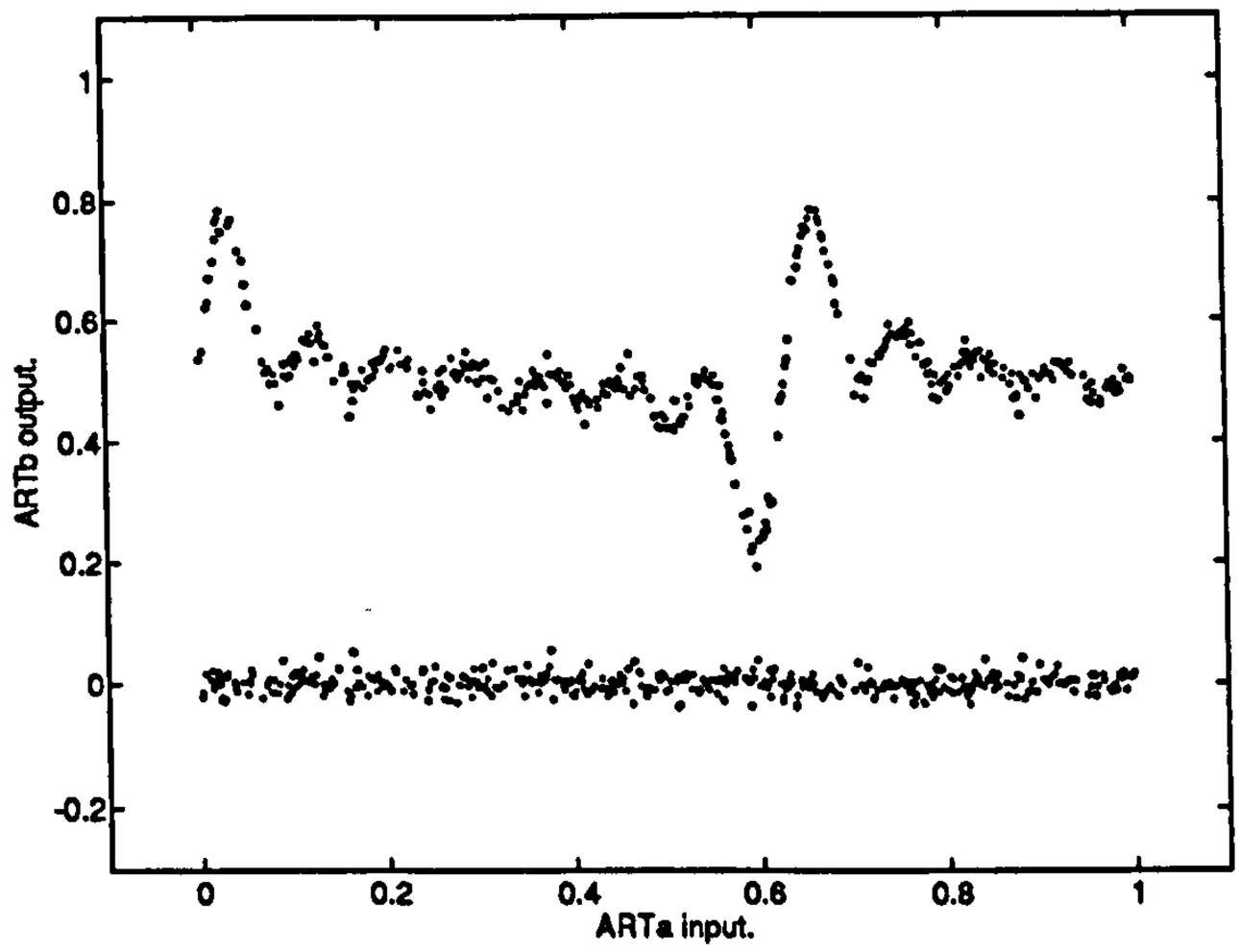


Figure 2.25. PROBART performance with noisy data and increased vigilance ($\rho_a = 0.999, \rho_b = 0.999$). This figure illustrates the effect of decreased category sizes brought about by the combination of increased vigilance and noisy input data.

The error profile bears some similarity to that of Figure 2.18 and reflects the increased vigilance leading to reduced category sizes and poorer generalisation. Comparing the mean results (Appendix E, Table E4.1) with the second mean set of simulation 2 (Appendix E, Table E2.4), it is apparent that a five-fold increase in the number of ARTa nodes has resulted in a 40% decrease in training RMSE (TR) and negligible change in both testing RMSE values. The mean MAXAE has been reduced in all three cases with a 50% reduction in mean training error (TR). Thus, although the testing RMSE values, TE(NF) and TE, are comparable, comparison of Figures 2.22 and 2.25 gives a clearer indication of what is happening.

This altered performance is explained by considering the ratio of approximately 2 data points per ARTa node which gives small samples for averaging to give an estimated output value. Thus, estimates are based on smaller sample sizes and are correspondingly less accurate.

2.7.7 Simulation 5

Increasing the number of training data points to 10,000 and using similar parameters gives the results of Table 2.8.

		Error measures					
No. of categories		RMSE			MAXAE		
ARTa	ARTb	TR	TE(NF)	TE	TR	TE(NF)	TE
1145	618	0.0265	0.0096	0.0114	0.0785	0.0255	0.0472

Table 2.8. Typical results for PROBART obtained using the parameters of simulation 4 with the noisy training file increased to 10,000 items.

Note that the mean RMSE value for the test set (TE) (Appendix E, Table E5.1) after training on a noisy data file of 10,000 points has been reduced to about 56% of its previous value for 1,000 data points (Appendix E, Table E4.1). There is also an additional two-fold increase in ARTa category nodes. This latter increase is explained by the increased number of uniformly distributed x-coordinates causing the packing density of ARTa nodes to rise, restricted only by the vigilance parameter.

The following graph, Figure 2.26, illustrates the variation in test RMSE for ARTa and ARTb vigilance in the range 0.99-0.999. The typical data set used throughout this run consists of a noisy training file of 10,000 pattern pairs and a noise-free test file of 1,000 pattern pairs. The general trend appears to indicate a reduction in RMSE for increased vigilance as expected. The upturn for a vigilance value of 0.999 further confirms the hypothesis that high vigilance values lead to smaller sample sizes and, thus, less accurate estimates of output values. There is a fundamental conflict between providing an adequate partitioning of the ARTa input space and adequate sample sizes for calculating the expected output value.

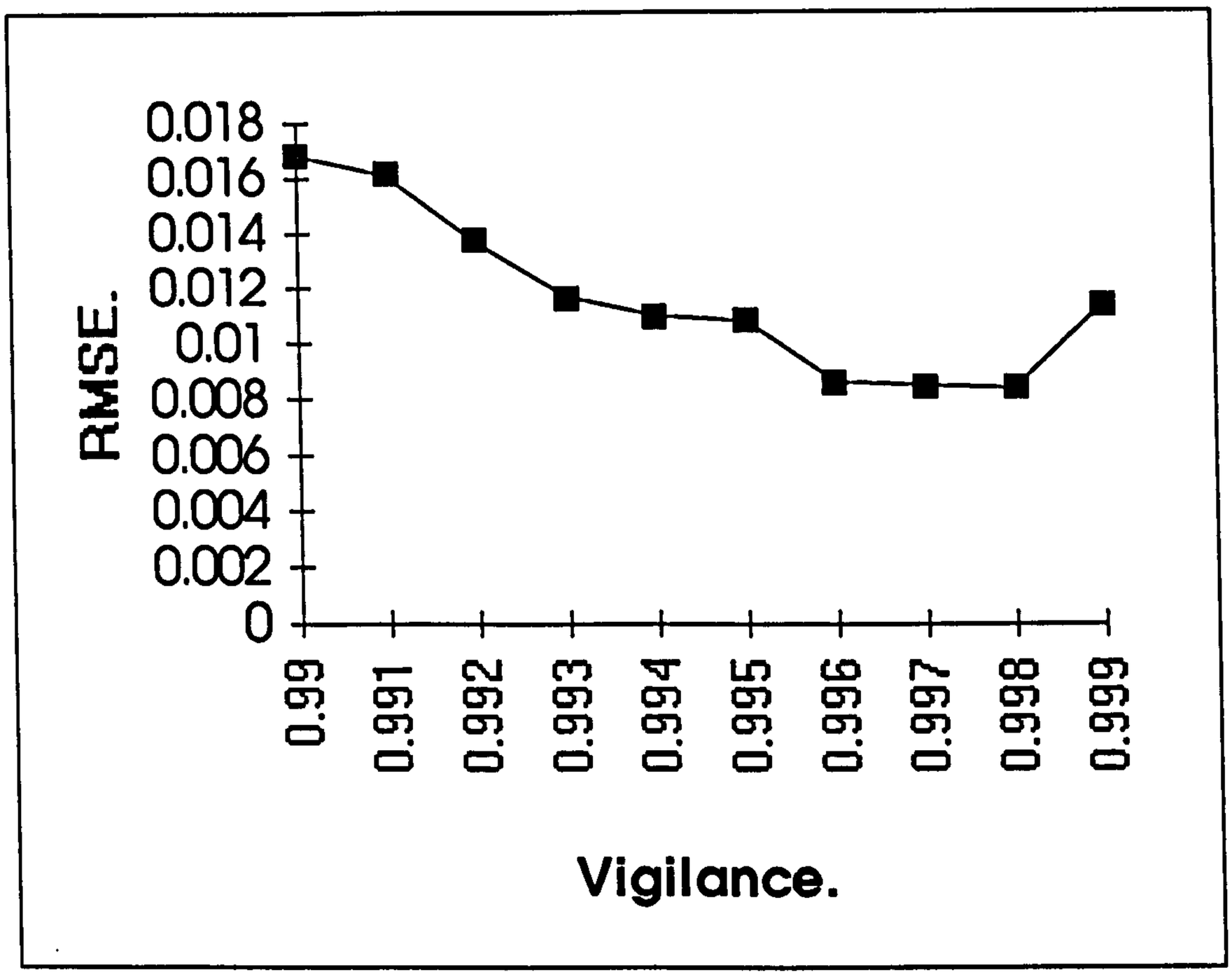


Figure 2.26. Plot of test (TE) RMSE vs. vigilance for typical noisy training data file of 10,000 pattern pairs. It illustrates the problem of increasing PROBART vigilance parameters to increase accuracy. Tighter categories, whilst increasing signal resolution, lead to smaller sample sizes and, thus, less accurate estimates of the underlying function.

Figure 2.27 illustrates the effect of increasing the size of the noisy training file for the typical data.

For vigilance values of $\rho_a = \rho_b = 0.998$, the results of Table 2.9 were obtained.

		Error measures					
Categories		RMSE			MAXAE		
ARTa	ARTb	TR	TE(NF)	TE	TR	TE(NF)	TE
608	341	0.0276	0.0079	0.0084	0.0779	0.0219	0.0269

Table 2.9. Typical results obtained by training PROBART on the 10,000 point noisy data file used in the previous simulations with vigilance levels set at $\rho_a = \rho_b = 0.998$. The test file used consisted of 1,000 noise-free pattern pairs.

Table 2.9 gives a further reduction of the test set RMSE (TE) over and above the typical value obtained in simulation 5 to 44% of that obtained with a 1,000 point training file (Table 2.7).

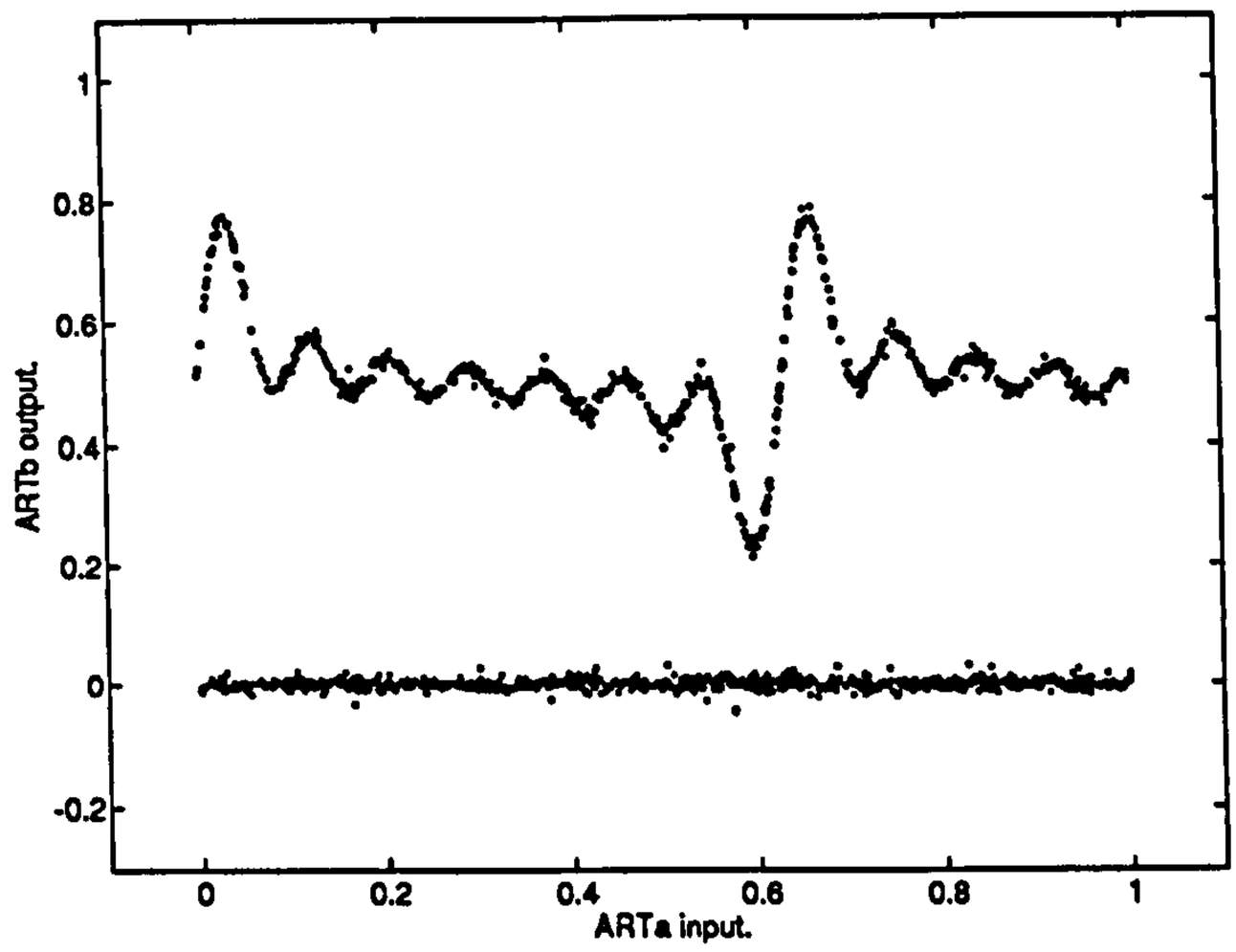


Figure 2.27. PROBART performance when trained on 10,000 point noisy data file with increased vigilance values of $\rho_a = 0.999, \rho_b = 0.999$. Comparing the error profile with that of Figure 12 indicates a reduced error as expected.

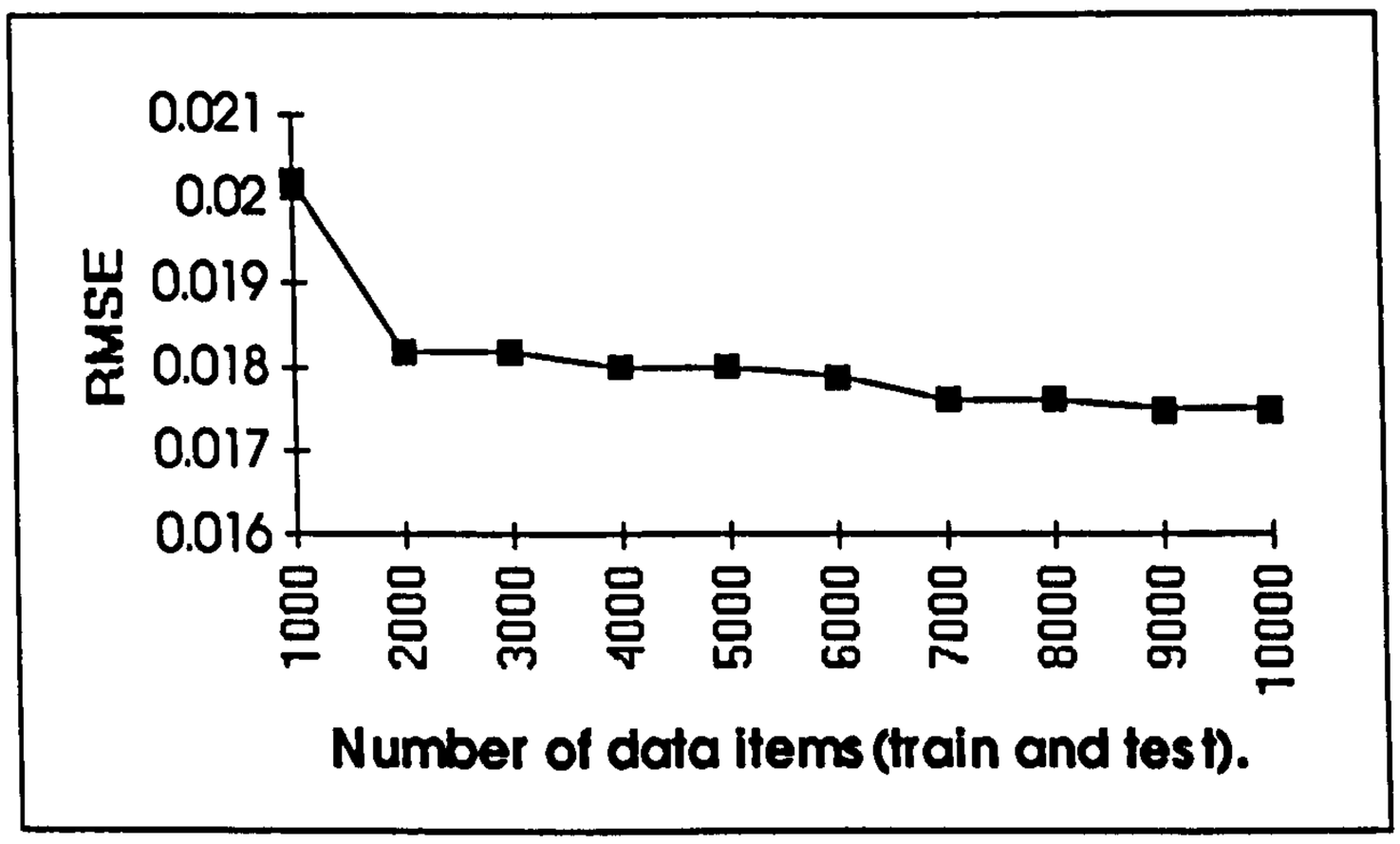


Figure 2.28. Plot of RMSE value against increasing data set size (both training and testing file size) for PROBART with fixed vigilance of $\rho_a = \rho_b = 0.99$. This plot shows the stability of RMSE with respect to changes in the data sample size. See text for explanation.

Figure 2.28 illustrates the stability of RMSE values for increasing training data sample size. The slight improvement for the larger amounts of data is explained by the increased cover density of the input and output spaces by exemplars and their category zones. Changes in RMSE values are directly affected by changes in the vigilance parameters. Increasing the amount of data only serves to pack the existing categories and create new categories limited by the vigilance values.

2.8 General Discussion

Both fuzzy ARTMAP and PROBART perform effectively with noise-free data, requiring only one pass through the training file (one epoch) for optimum learning in the RMSE sense (lowest error energy). In contrast with fuzzy ARTMAP, PROBART carries out a single epoch for all training and testing as match tracking has been removed. This prevents distortion of the computed probabilities (frequency count/ total pattern pairs). For example, for a fixed vigilance, an output, y_1 has the conditional probability given the interval I_{x_1} of $p(y_1|I_{x_1})$ for an interval I_{x_1} based around an exemplar x_1. Were the interval partitioned into two sub-intervals $I_{x_{11}}$ and $I_{x_{12}}$, by increasing vigilance (formation of a sub-category), there is no method of allocating the current frequency count based upon interval I_{x_1} to intervals $I_{x_{11}}$ and $I_{x_{12}}$ individually. Thus, $p(y_1|I_{x_{11}})$ and $p(y_1|I_{x_{12}})$ cannot be derived from $p(y_1|I_{x_1})$. Also, feedback via match tracking alters the frequency of inter-ART node associations by assessing current inputs on the basis of previous data and not by recording raw frequencies. This situation cannot reflect a true empirical frequency distribution upon which the estimated outputs or pattern association probabilities are based.

Fuzzy ARTMAP is extremely good at classification problems but match tracking tends to cause the allocation of many nodes for noisy mappings with the noisy disturbances seen as novel features. The dynamics expressed in equation (2.15) do not act as an effective filter at high vigilance levels (≥ 0.9) using FCSR. This is a consequence of LTM exemplar weights being very near to the noisy input values which fall into their categories. The convex combination of equation (2.15) gives LTM weight values close to the original exemplar values.

It is difficult to classify neural networks as good or bad on the basis of raw results alone. Overall performance also depends upon the problem to which the network or algorithm is applied. Another factor is the degree of specialisation of the network. Enhanced performance is often obtained at the expense of decreasing

generality, i.e. the architecture moves away from being general purpose and becomes oriented towards a particular problem or problem schema. This specialisation frequently requires the incorporation of *a priori* information or structure into the neural network and its dynamics and, thus, restricts its range of applicability.

To a certain extent, PROBART is a trade-off between performance and generality in that better performance could no doubt be obtained using a more specialised network architecture but it does not require *a priori* information about the mapping to be learned. Given that PROBART deviates significantly from fuzzy ARTMAP, it begs the question why use fuzzy ARTMAP at all? The answer lies in the known attractive properties of ART, in particular, their stability. Other clustering algorithms based, say, on Euclidean distance are known to have stability problems under some circumstances. Moore (1989) cites the *Cluster Euclidean* algorithm which chooses the node coding for the nearest exemplar to the input vector in the Euclidean distance sense. Incorporating equation (2.15) to give the *Cluster Unidirectional* algorithm (Moore, 1989) removes the endless cycling of weight vectors but suffers from the category proliferation problem countered by the use of complement coding in fuzzy ART.

2.9 Multidimensional Mappings

As stated, fuzzy ARTMAP is capable of mapping subsets of $\Re^m$ to $\Re^n$. PROBART is also capable of such mappings. A visual illustration of this capability is included here in the form of a continuous non-linear mapping from $\Re^2$ to $\Re$ which is shown in Figure 2.29.

Again, Gaussian noise, derived from a zero mean source with unit variance, is added to the signal with a scale factor of 0.02. Conditions and performance measures are similar to those used in the previous single variable mapping but are generalised for the present multivariable mapping.

2.9.1 Simulation 6

Fuzzy ARTMAP was trained on noisy data. Its parameters were set as follows: $\alpha = 0.001, \rho_a = 0.99, \rho_b = 0.99$ and $\rho_{ab} = 0.9$. Both the training and test sets consisted of 1,000 data pairs.

An example of fuzzy ARTMAP performance for the training signal with noise is shown in Table 2.10.

		Error measures					
No. of categories		RMSE			MAXAE		
ARTa	ARTb	TR	TE(NF)	TE	TR	TE(NF)	TE
955	63	0.0075	-	0.0235	0.01	-	0.077

Table 2.10. Typical results for fuzzy ARTMAP trained using a noisy version of the signal illustrated in Figure 2.29 Both the noisy training file and the non-noisy test file consisted of 1,000 pattern pairs to be associated. The network parameters used were $\alpha = 0.001, \rho_a = 0.99, \rho_b = 0.99$ and $\rho_{ab} = 0.9$.

The network output and error profile are shown in Figures 2.30(a) and 2.30(b) respectively.

Fuzzy ARTMAP requires almost one node per data item. Thus, under high vigilance conditions, it acts as a look-up table by storing and retrieving individual pattern pairs. The error profile reproduces the original errors almost faithfully as nearly all individual errors are recorded. It is also apparent from Table 2.10, as with the single variable examples, fuzzy ARTMAP has learnt the noisy signal.

Changing the learning parameter, β (using FCSR) made very little difference. Using values of 0.5 and 0.9 gave testing RMSE values of 0.0236 and 0.0235 respectively. The numbers of ARTa categories were 955 and 950 respectively. The high vigilance parameters for ARTa and ARTb prevented the occurrence of large changes in RMSE values during training.

2.9.2 Simulation 7

Increasing the number of training data points to 5,000 and using similar parameters (FCFR) gives the results shown in Table 2.11.

		Error measures					
No. of categories		RMSE			MAXAE		
ARTa	ARTb	TR	TE(NF)	TE	TR	TE(NF)	TE
4528	101	0.0076	-	0.0307	0.01	-	0.0743

Table 2.11. Typical Results for Fuzzy ARTMAP when retaining the parameters used to obtain the results of Table 2.10. The noisy training data file was increased from 1,000 to 5,000 pattern pairs. The noise-free testing file remained at 1,000 pattern pairs.

These results are illustrated in Figures 2.31(a) and 2.31(b). Note the number of ARTa categories which indicate that, as expected, little generalisation has occurred.

2.9.3 Simulation 8

PROBART was trained on the same sets of noisy and noise-free data used in simulation 6. The parameters are set identically to those in that simulation except for the map field vigilance which is not required.

Typical results for the training signal with noise are shown in Table 2.12.

		Error measures					
No. of categories		RMSE			MAXAE		
ARTa	ARTb	TR	TE(NF)	TE	TR	TE(NF)	TE
739	63	0.0163	-	0.0196	0.0497	-	0.0775

Table 2.12. Typical results for PROBART trained using a noisy version of the signal illustrated in Figure 16. Both the noisy training file and the non-noisy test file consisted of 1,000 pattern pairs to be associated. The network parameters used were

$\alpha = 0.001, \rho_a = 0.99, \rho_b = 0.99$.

These results are illustrated in Figures 2.32(a) and 2.32(b).

Note that, compared to simulation 6, approximately 23% fewer ARTa nodes are required to represent the mapping for a comparable value of testing RMSE. The following simulation (simulation 9) illustrates further reductions in the number of ARTa nodes for PROBART relative to fuzzy ARTMAP.

2.9.4 Simulation 9

Categories		RMSE			MAXAE		
ARTa	ARTb	TR	TE(NF)	TE	TR	TE(NF)	TE
2283	101	0.0232	-	0.0216	0.065	-	0.067

Table 2.13. Typical Results for PROBART when retaining the parameters used to obtain the results of Table 11. The noisy training data file was increased from 1,000 to 5,000 pattern pairs. The noise-free testing file remained at 1,000 pattern pairs.

The results of Table 2.13 are illustrated in Figures 2.33(a) and 2.33(b).

Comparing Table 2.13 with Table 2.11 shows a reduction of approximately 50% in the number of ARTa nodes required to represent the mapping. This reduction is not at the expense of testing RMSE (TE) which has been reduced by 30%. This indicates the improved performance offered by PROBART when dealing with larger data sets.

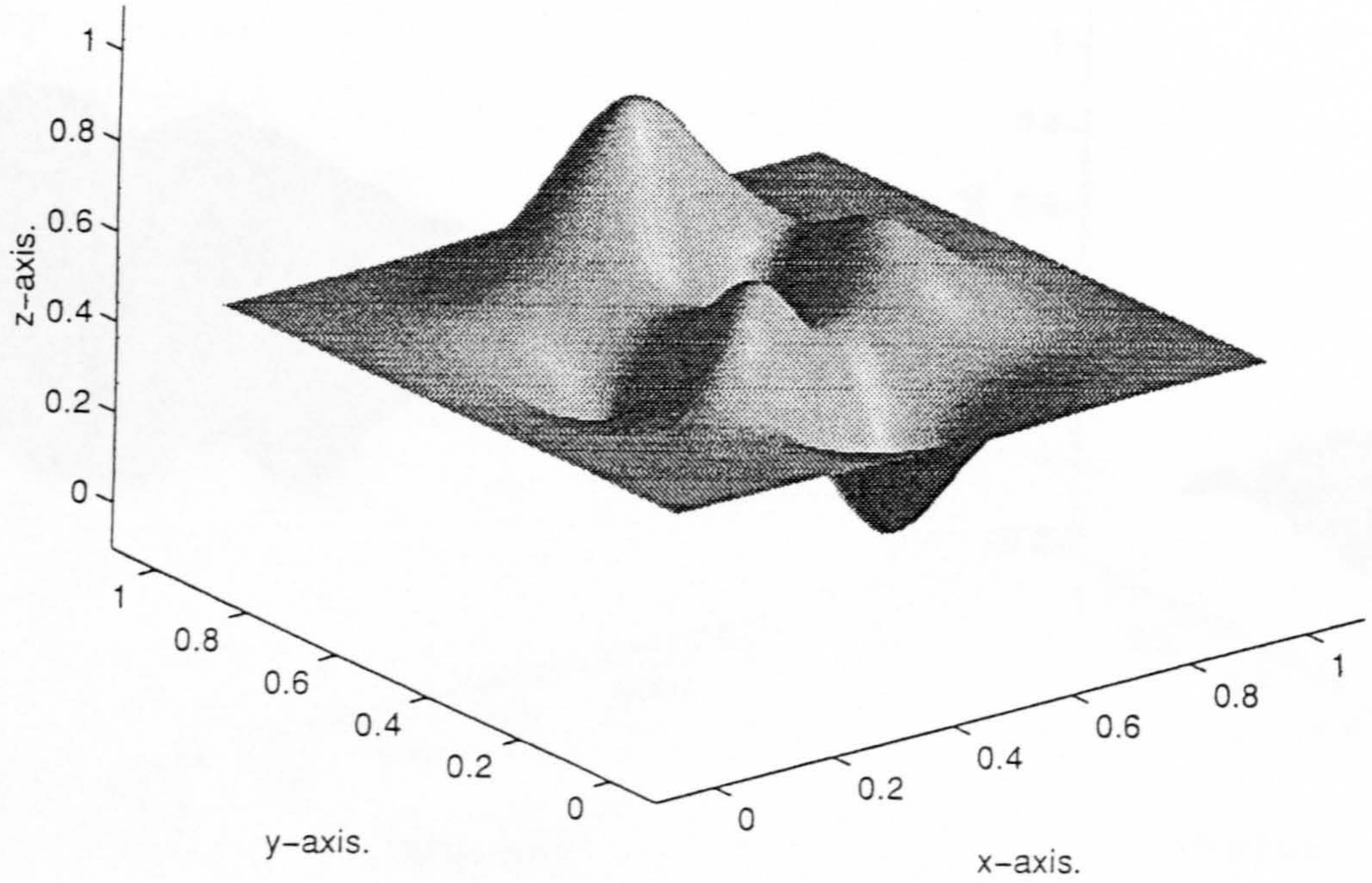


Figure 2.29. Non-linear test signal $[0,1]^2 \subset \Re^2 \rightarrow [0,1] \subset \Re$ used in the evaluation of fuzzy ARTMAP and PROBART performance.

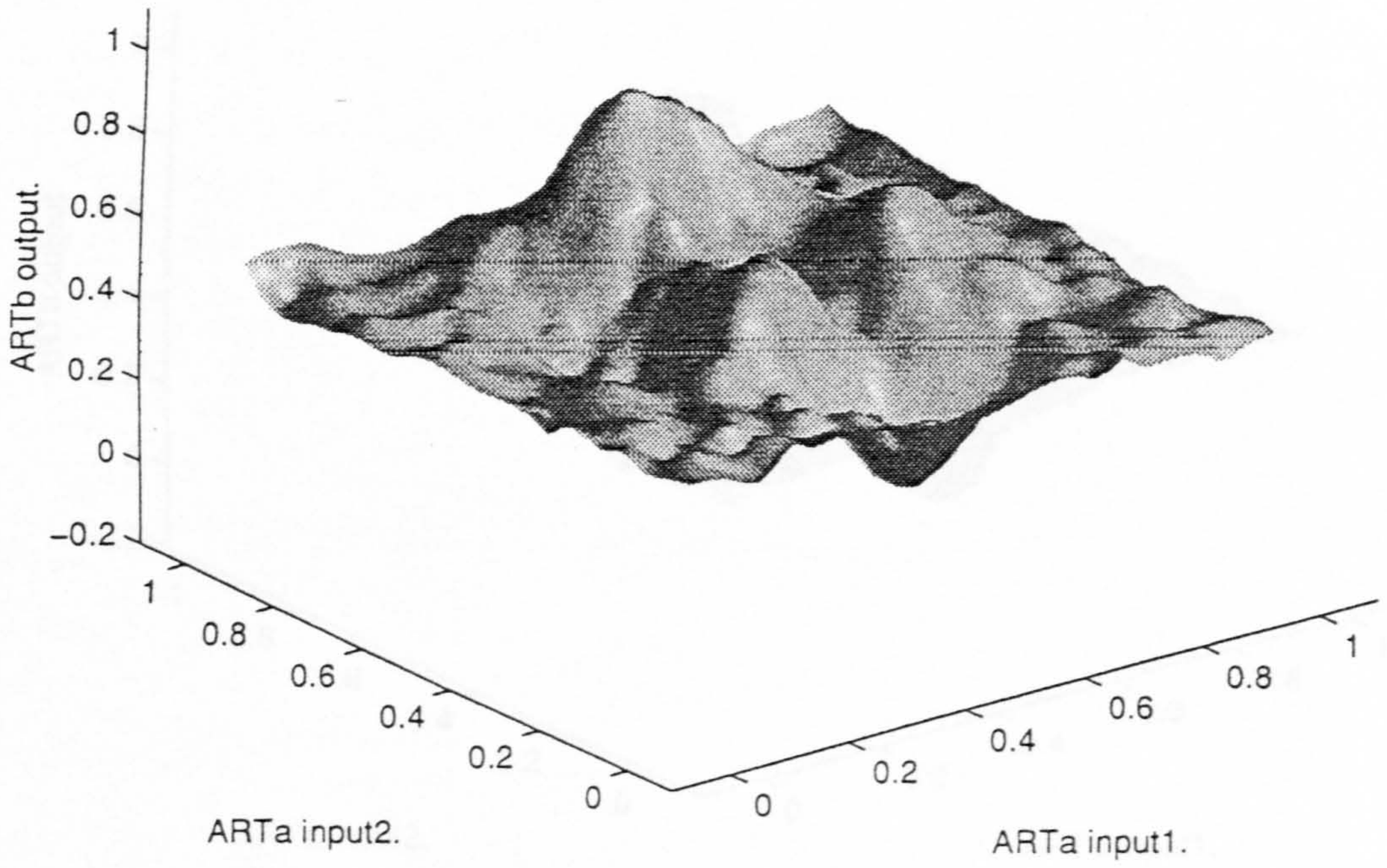


Figure 2.30(a). Fuzzy ARTMAP output after being trained on a noisy version of the non-linear signal shown in Figure 2.29. Both training and testing files consist of 1,000 data points. The network parameters used were $\alpha = 0.001, \rho_a = 0.99, \rho_b = 0.99$ and $\rho_{ab} = 0.9$.

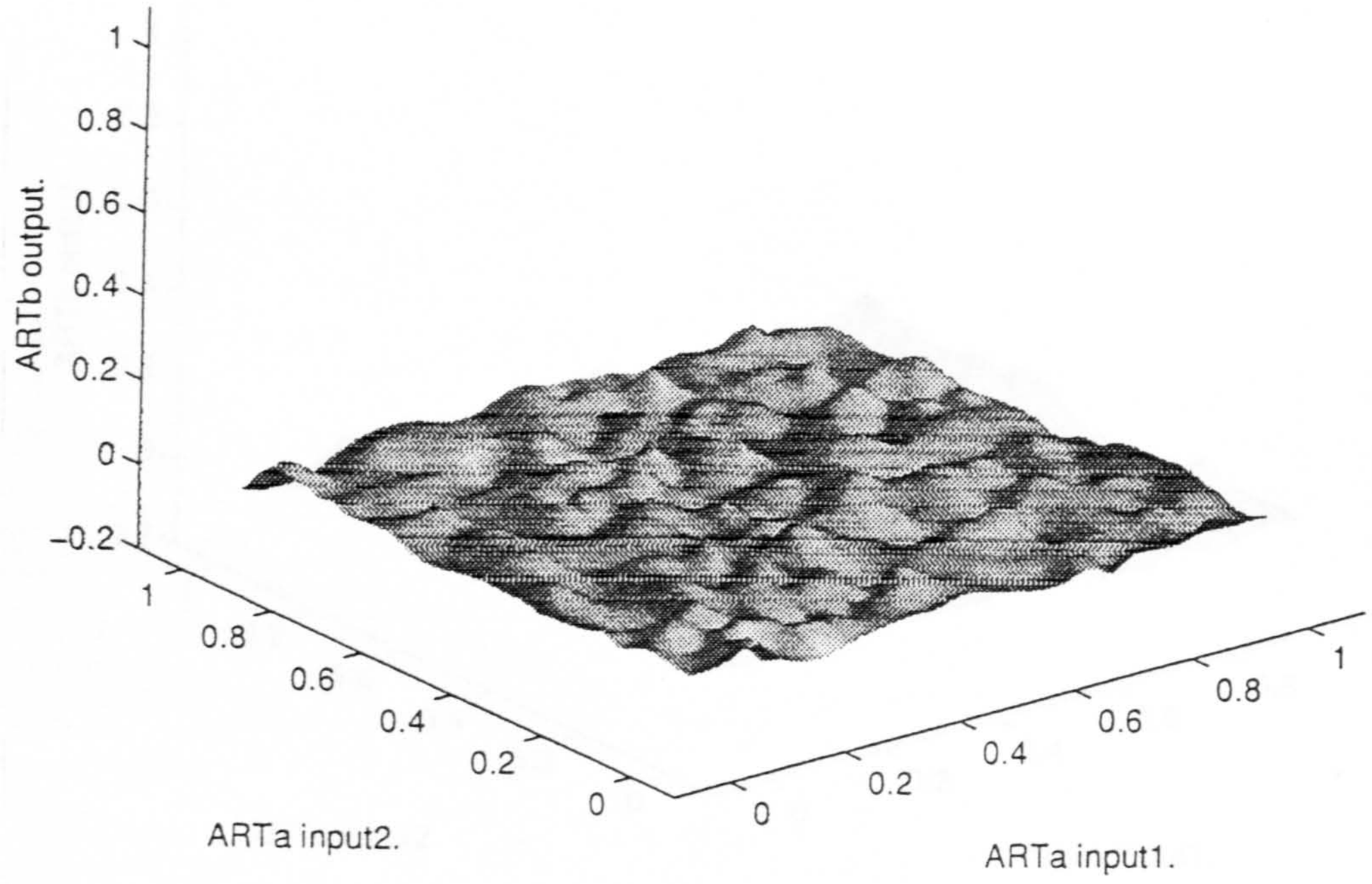


Figure 2.30(b). Fuzzy ARTMAP error profile for simulation 6.

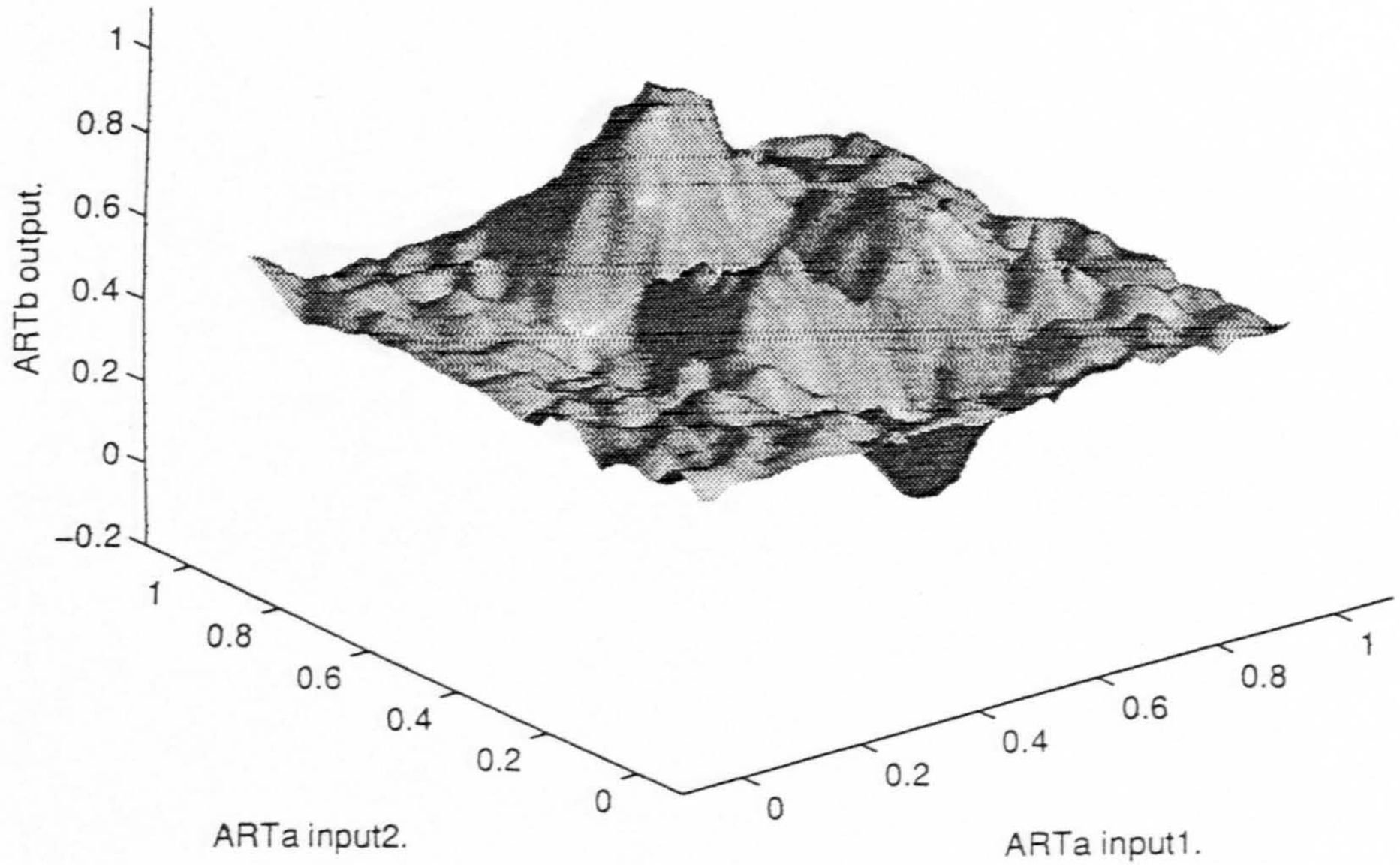


Figure 2.31(a). Fuzzy ARTMAP output for the noisy non-linear signal when trained on 5,000 data points. Test set remains at 1,000 data points.

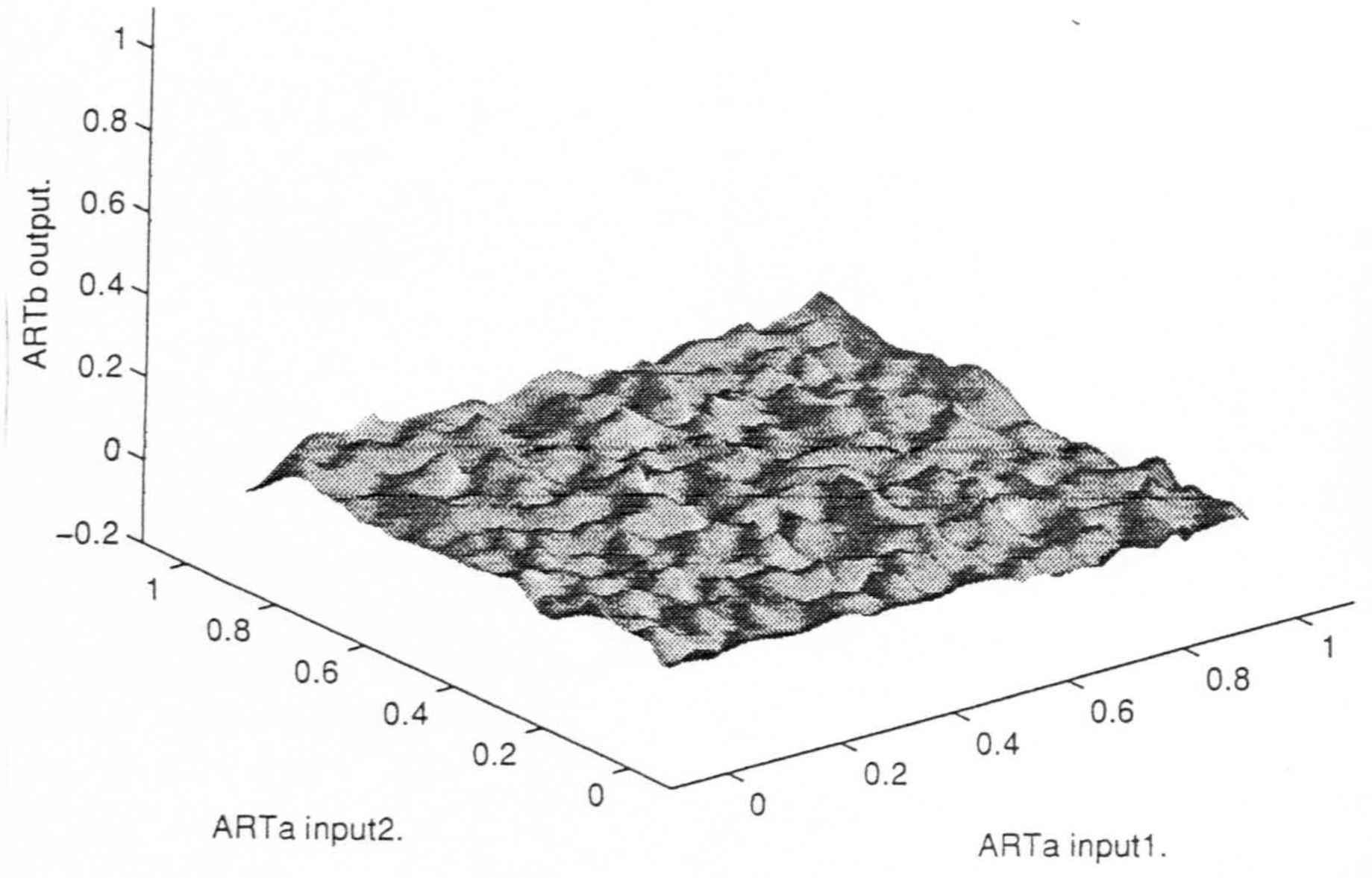


Figure 2.31(b). Error profile for the 5,000 data point run of simulation 7.

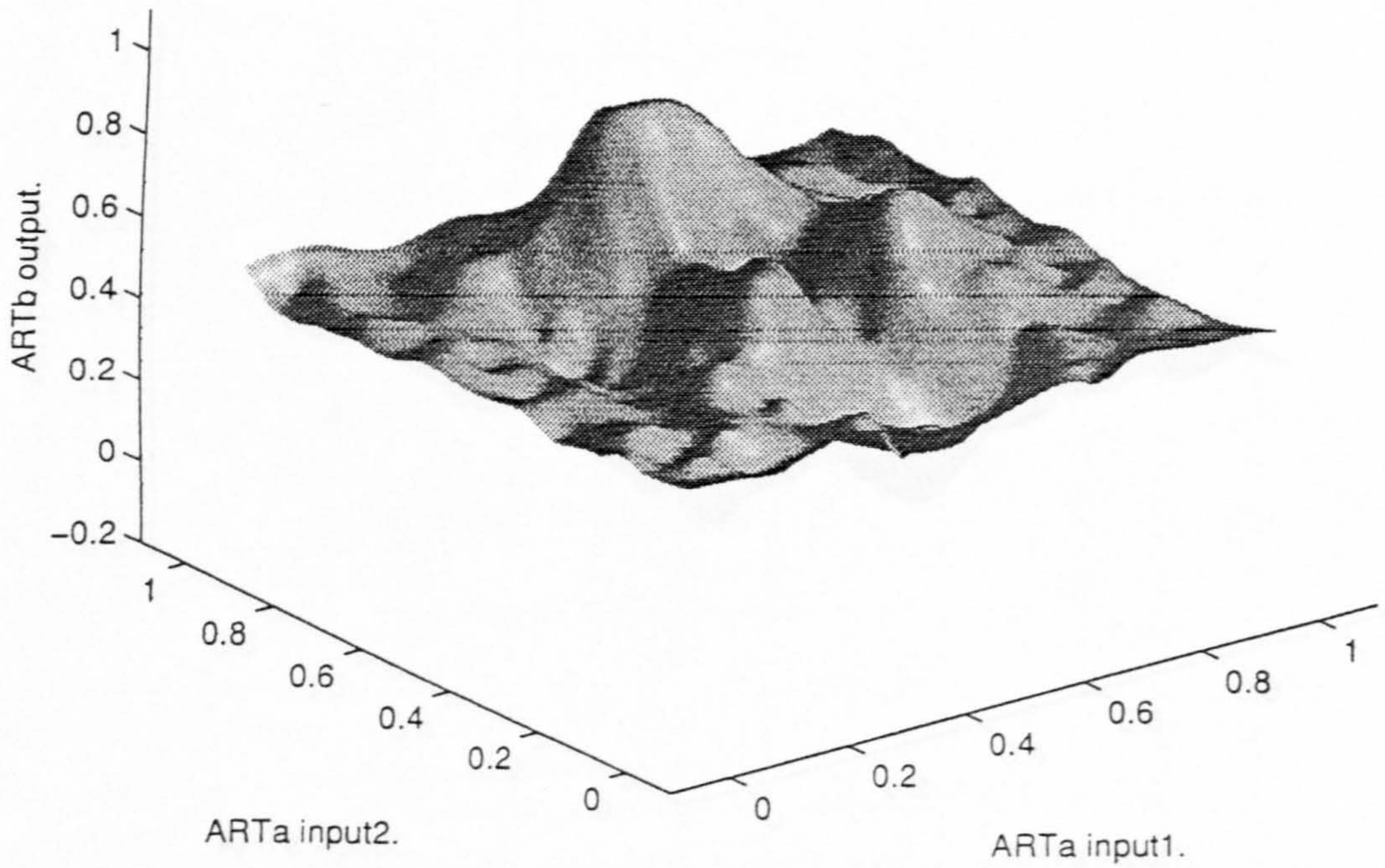


Figure 2.32(a). PROBART output after being trained on a noisy version of the non-linear signal shown in Figure 16. Both training and testing files consist of 1,000 data points.

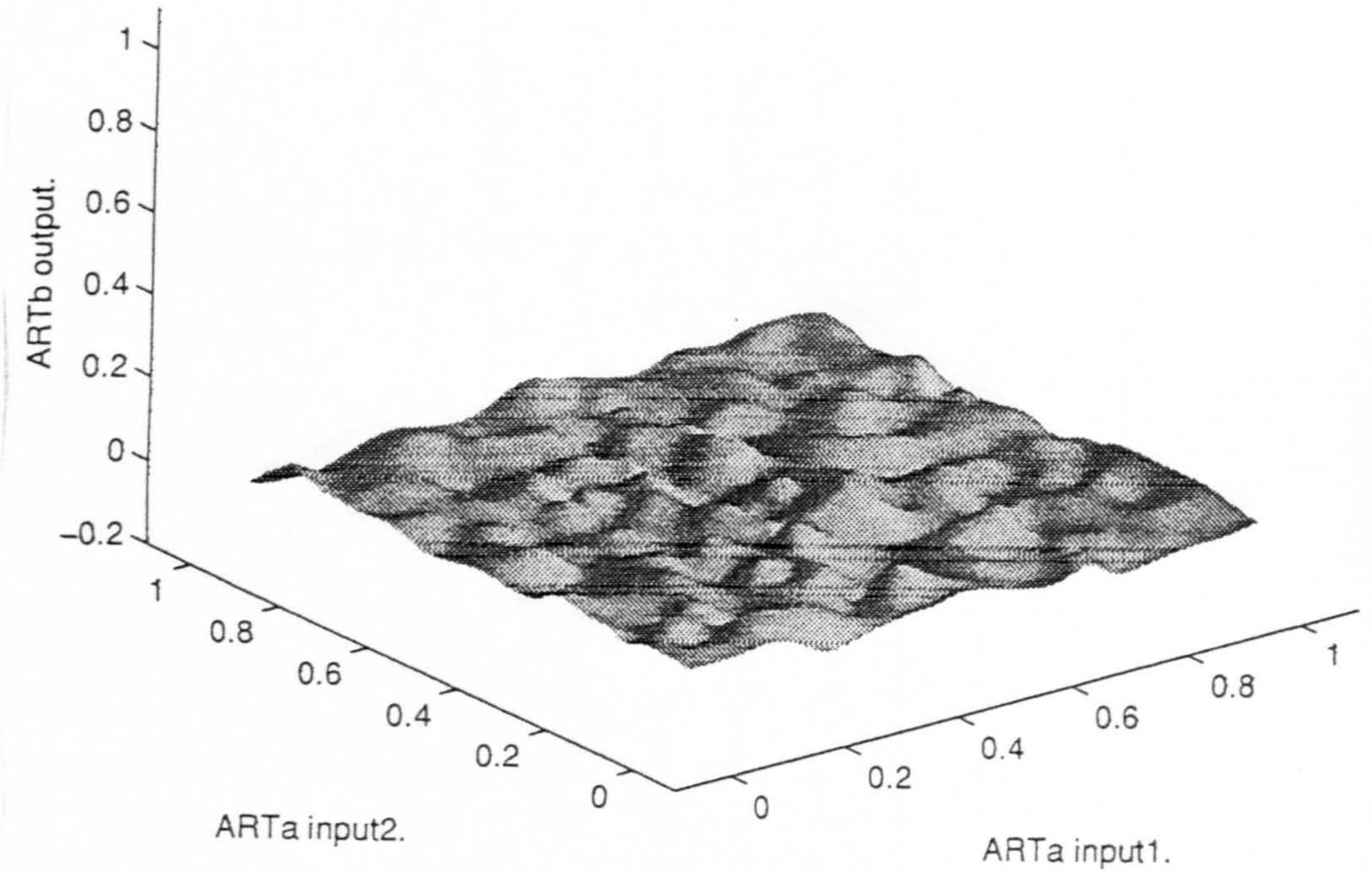


Figure 2.32(b). PROBART error profile for simulation 8.

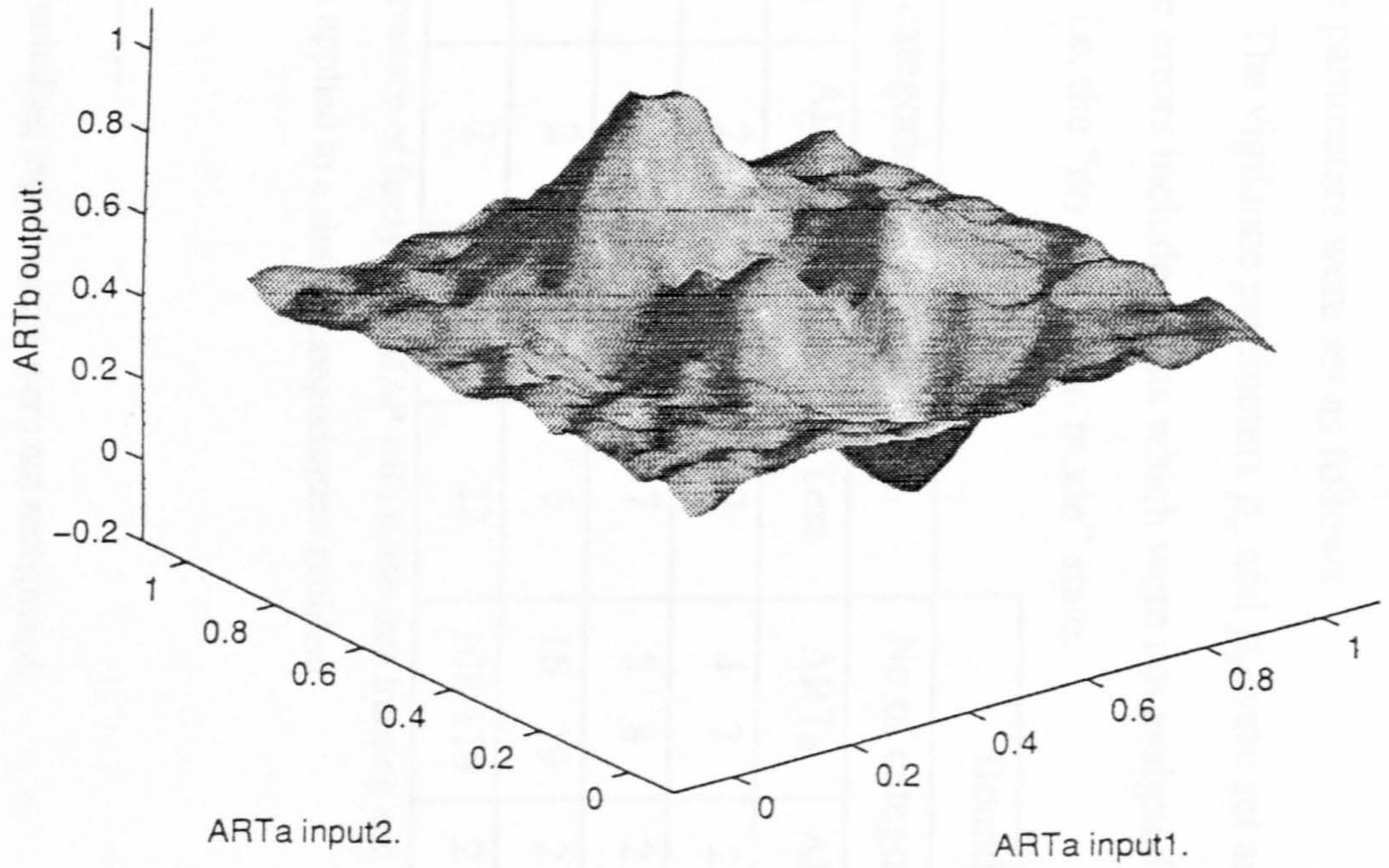


Figure 2.33(a). PROBART output for noisy non-linear signal when trained on 5,000 data points. Test set remains at 1,000 data points.

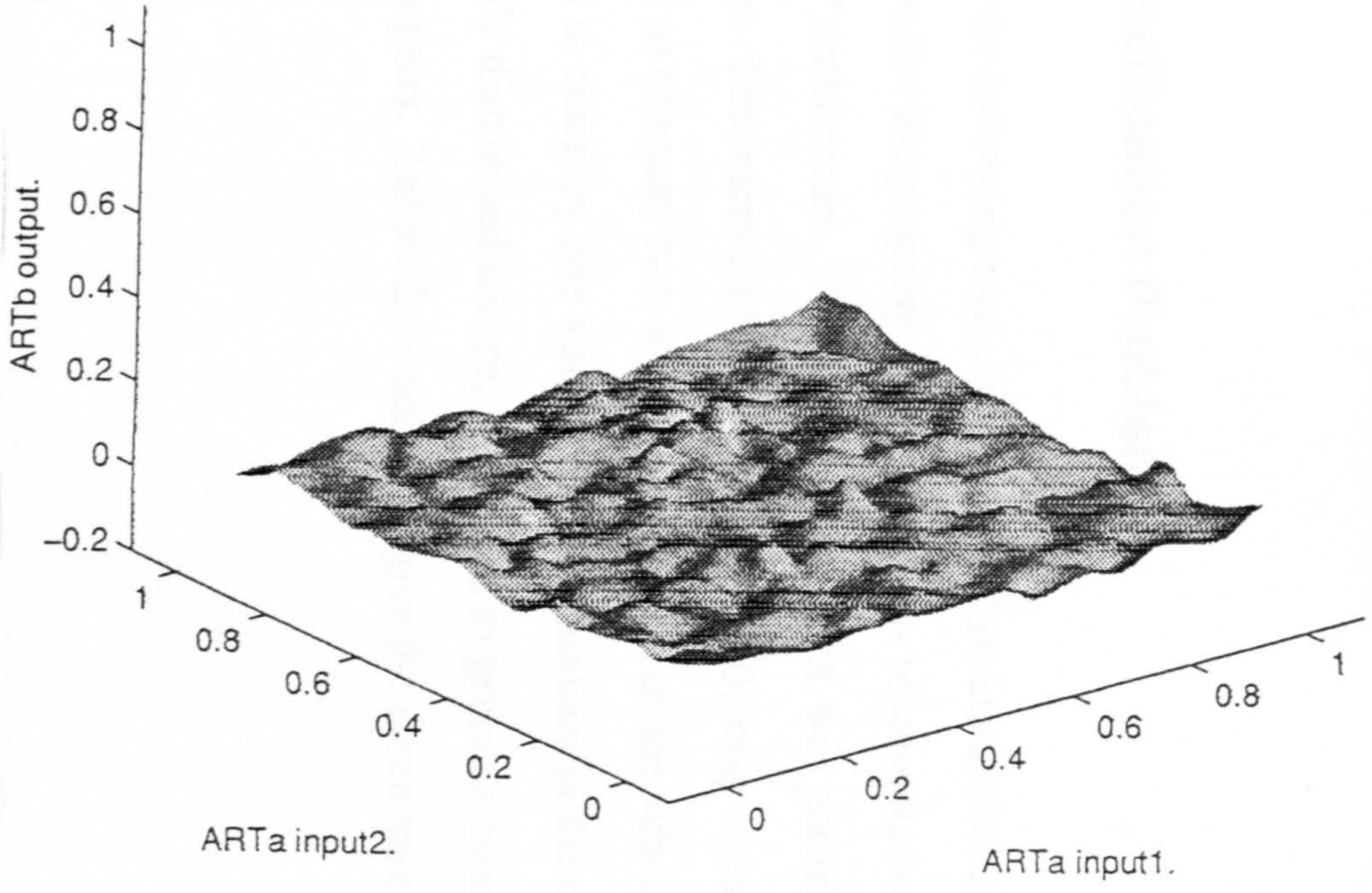


Figure 2.33(b). PROBART error profile for the 5,000 data point run of simulation 9.

2.10 A Simple Classification Problem

In contrast to the above, the following two simulations illustrate the utility of match tracking which confers the property of parsimonious representation on the original fuzzy ARTMAP architecture. The interval $[0,1] \subset \Re$ was partitioned into two categories as follows: The intervals [0,0.3] and [0.7,1.0] map to category one, and (0.3,0.7) maps to category two with the exception of the sub-interval [0.45,0.55] which maps to category one also. Random numbers in the range 0 to 1, drawn from a uniformly distributed source, were used to generate training and testing sets of 1,000 data pairs. Tables 2.14 and 2.15 give the mean results of five trials.

2.10.1 Simulation 10

Fuzzy ARTMAP was trained on 1,000 data pairs and tested on a different testing set of the same size. The parameters were set as follows

$\alpha = 0.001$, and $\rho_{ab} = 0.9$. The vigilance parameters ρ_a and ρ_b were set as given in Table 2.14. Note that the errors include inputs which were not assigned to categories during testing i.e. the "no prediction mode" state.

Vigilance	No of categories		Error[1]		Bounds (Min. Max.)			
					No of categories		Error	
ρ_a, ρ_b	ARTa	ARTb	Train	Test	ARTa	ARTb	Train	Test
0.2	6	2	0	7	4 7	2 2	0 0	0 14
0.5	6	2	0	7	5 8	2 2	0 0	0 14
0.9	17	2	0	5	16 19	2 2	0 0	0 12
0.99	113	2	0	22	108 119	2 2	0 0	5 30

Table 2.14. The mean performance of fuzzy ARTMAP with noise-free training and testing data when applied to a simple categorisation problem.

[1]No. of incorrect categories including inputs which were not recognised.

2.10.2 Simulation 11

PROBART was trained using the same conditions as in simulation 10 but without the map field parameter. The mean results are shown in Table 2.15.

					Bounds (Min. Max)			
Vigilance	No of categories		Error		No of categories		Error	
ρ_a, ρ_b	ARTa	ARTb	Train	Test	ARTa	ARTb	Train	Test
0.2	2	2	300	310	2 2	2 2	289 312	310 310
0.5	3	2	285	308	2 3	2 2	212 312	300 310
0.9	14	2	90	106	12 16	2 2	54 124	66 152
0.99	113	2	6	32	107 116	2 2	2 12	17 44

Table 2.15. The mean performance of PROBART with noise-free training and testing data when applied to a simple categorisation problem.

Simulations 10 and 11 illustrate the points made regarding match tracking in the discussion of simulation 1. Fuzzy ARTMAP is able to represent categories efficiently by varying the vigilance parameter through match tracking. The increased error at high vigilance is accounted for by the narrow scope of categories which cause some patterns to go unrecognised. PROBART behaves as expected with a fixed category size. With low vigilance, category membership frequency causes the higher frequency category (category one) to dominate with an error rate of approximately 30% as predicted from the distribution. At very high vigilance (≥ 0.99) differences between PROBART and fuzzy ARTMAP are negligible as there is little scope for incrementing ρ_a during learning.

2.10.3 A Short Conclusion

It is self-evident that some neural networks do better at certain tasks than others. Often, a specialised network will outperform its more general counterpart but suffers from the disadvantage of requiring *a priori* information pertaining to the learning task. Thus, autonomy is reduced as operator knowledge is built into the network to guide learning. ART-based systems are self-organising and so reduce

the need for intervention. They exhibit attractive properties such as the ability to operate in non-stationary environments and to learn continuously new associations following training, without disrupting previous learning. However, the independence of nodes, as in fuzzy ARTMAP, leads to over learning and reduced generalisation as noisy associations are treated as novel associations in noisy mapping problems. The mechanism of match tracking which allows sub-categories to be resolved in classification problems causes categories to proliferate when noisy mapping approximations are carried out. Rapidly changing regions of a mapping—often resulting from the superposition of noise on the underlying signal—are treated as misclassifications requiring new ART a and ART b category nodes with alternative links via the map field. PROBART goes some way to rectifying this by using probability information, combined from various nodes, to estimate output values. The benefits of using PROBART when dealing with noisy mappings include a reduction in RMSE values, an improved error profile, a sizeable reduction in the number of ARTa category nodes and increased generalisation.

As illustrated, PROBART is also capable of classification and exhibits performance similar to fuzzy ARTMAP at high vigilance. For efficient performance on classification tasks, however, fuzzy ARTMAP is the preferred architecture where classes are resolved accurately using few nodes. As with all tasks the architecture must be matched with the problem and the ART family of networks is no exception.

While PROBART requires fewer nodes than fuzzy ARTMAP to achieve similar performance for a complex mapping task, it has not *solved* the generalisation problem. For example, in the testing phase, some inputs are rejected and consequently no prediction can be made because those inputs are not within the range of any relevant ARTa category. Neither fuzzy ARTMAP nor PROBART construct a mapping using a sum of weighted basis functions. Although this property confers several advantages on the ART family of architectures, the danger is that—under certain conditions—they may become nothing more than

sophisticated look-up tables. One possible solution would be to remove both the ARTa vigilance and "winner-takes-all" dynamics during the testing phase. This would allow a local neighbourhood to be established around the input and a weighted interpolation procedure, depending upon the degree of match between the input and existing ARTa categories, to be carried out to provide an output prediction. The result would be a reduction in the number of inputs which do not lead to a prediction; a problem which increases with higher dimensional space as the input data becomes less densely packed.

The present version of PROBART uses a simple average to calculate the output approximation. This average is made possible owing to the multiple linkages allowed between input categories and output categories. Through the map field frequency counts a rough approximation to the probability distribution of the output values could be made. For a single input category, linkages to multiple output categories could be stated individually together with their respective frequencies. Thus, PROBART could approximate multimodal distributions and thereby remove the one-to-many mapping restriction of fuzzy ARTMAP.

A possible continuous version of PROBART (hence exhibiting generalisation) would have the form $\hat{\mathbf{y}} = \sum_{i=1}^{K} f_1\left(\left\|\mathbf{x} - \mathbf{c}_i^a\right\|\right) \sum_{j=1}^{K_i} f_2\left(w_{ij}^{ab}\right) \mathbf{c}_j^b$ where K signifies the set of nearest ARTa nodes to the input x, $\mathbf{c}_i^a$ and $\mathbf{c}_j^b$ are the node exemplars or centroids of ART a and ARTb nodes respectively, K_i signifies the set of ART b nodes associated with the i th ART a node, $f_1(.)$ is a normalised 'activity' function in the range [0,1], $f_2(.)$ is a normalised weighting in the range [0,1] and w_{ij}^{ab} is the frequency of association between the i th ART a node and the j th ART b node. In the limiting case where the winning ART a node only is chosen with index I, the estimate is given by $\hat{\mathbf{y}} = \sum_{j=1}^{K_I} f_2\left(w_{Ij}^{ab}\right) \mathbf{c}_j^b$ where $f_1\left(\left\|\mathbf{x} - \mathbf{c}_I^a\right\|\right) = 1.0$ and choosing $f_2(.)$ to be the relative frequency gives equation (2.18).

Both Fuzzy ARTMAP and PROBART provide a "rough and ready" method of approximating mappings and can be implemented as on-line versions. The underlying concern of adaptive resonance theory is with pattern classification and recognition but the theory does not entirely preclude mapping as shown. The trade-off between precise mapping and adaptiveness may be worth making for certain applications. One area of application is neurocontrol where precise mapping may be too time-consuming and computationally expensive making some problems untenable. Precise mapping may not be needed because error-correction techniques adapt quickly providing there is an underlying mechanism which allows rapid storage and retrieval of control information. The CMAC (Albus, 1975a,b) is an example of an adaptive memory system which functions somewhat like a look-up table and that allows rapid and flexible adaptation through the use of hash coding to retrieve or store information quickly.

To put the neurocontrol issue into perspective, consider human and animal behaviour. Although constructed from "components" with wide tolerance limits, inherent disturbances and relatively imprecise connectivity, the nervous system is highly adaptive and successful. There is simply not enough time for all the subsystems involved to compute trajectories and apply the techniques of inverse kinematics etc.; movements are made and fine-tuned immediately using multiple sources of feedback.

Consideration of the mapping problem, and some of the issues involved, reveals a more fundamental concern with supervised learning; the very fact that it is *supervised*. This concern will be addressed throughout this thesis but a few words are in order here. Training data is usually in the form of a set of pattern pairs, and there is predefined structure already present—albeit implicitly— in the statement of a learning problem. The supervised model of learning—used extensively in the neural network field is not the only model of learning and, furthermore, does not account for the majority of learning.

For the purposes of system identification, supervised learning methods may provide a model of a system from samples of the system's behaviour. In many cases, however, one black box is being swapped for another. Without analytical information to synthesise a system model, it may be just as difficult—if not more so—to extract relevant information from the collection of weighted connections between nodes in a neural network.

Most criticism can be directed not towards learning mappings from 'raw' data, but towards using supervised learning methods for training controllers. Where does the control information, pertaining to the desired control actions, come from? Observing the input-output behaviour of an unknown system and using supervised learning to develop a mapping (almost certainly not in minimal form) is one thing, but training a neurocontroller with desired control information is another.

Knowing the intermediate control actions to achieve a given control objective—using a neural network trained with supervised learning—implies that the control problem has already been solved in some sense. If this were not the case, then how would the "correct" intermediate control actions be known? For example, using an existing control strategy or modelling an expert reduces to nothing more than transferring control "knowledge" to a neural network platform.

A more desirable and more biologically realistic situation is to have autonomous learning systems which are capable of discovering temporospatial structure and order for themselves with a minimum of *a priori* information except where it is beneficial or easily produced in any given situation. Such autonomous systems would be goal-driven and strive to develop successful behavioural strategies which enabled them to achieve the stated goals. The possible move away from "dim" number-crunching neural network architectures towards more flexible and adaptive structures is discussed in Chapter 3 onwards where an alternative learning model, reinforcement learning, is explored.

Chapter 3 Reinforcement Learning

3.1 Psychology

3.1.1 Behaviourism

Until the early part of the twentieth century, psychology still relied heavily on the process of introspection. However, the practice of self-observation was not entirely satisfactory and the idea of unconscious mental processes became more and more acceptable (Hergenhahn, 1992). The move away from introspective psychology was coupled with an increasing adoption of objective experimental protocols which formed the basis for the newly emerging science of experimental psychology. In 1913 John Watson formulated the concept of *Behaviourism* which treated organisms as "black boxes" (Watson, 1913). Subjective sensations were ignored and psychologists collected data purely through observation of an organism's external behaviour. Watson believed that learning is the most important factor in the development of behaviour patterns and that all human skills, personality traits and motives are learned. Even complex behaviour patterns are believed to consist of sequences of multiple conditionings acquired through continuous learning throughout life (Hebb, 1972). In terms of a neural-network model, action sequences are acquired through on-line supervised learning. As plausible as this theory of human and animal behaviour sounds, it is far too simplified and posits organisms only as passive responders to external stimuli.

The psychologist E. L. Thorndyke (Thorndyke, 1911) postulated that organisms were much more active, and that learning took place through gradual adjustments in behaviour following random actions (stochastic search). Actions which were successful in a particular context were more likely to be repeated in the same context at some future time. Thorndyke used a "puzzle box" in which he placed a single cat. The box had a latch on the door which the cat had to operate correctly

in order to free itself; this required a sequence of actions which had to be learned. Cats learned to "solve the puzzle" by trying sets of actions which became focused on the latch as time progressed. Spontaneous action sequences were generated by the animal and not through an explicit training program of stimulus patterns and desired responses.

Associative learning consists of associating behavioural "atoms" in humans and animals. Experimental procedures have been developed to allow the study of associative learning—involving response conditioning—in controlled environments. There are two main types of associative learning studied in experimental psychology:

- *classical conditioning* which involves conditioning an organism's responses to externally applied stimuli, and

- *operant conditioning* which involves conditioning the type and intensity of spontaneous behaviour generated by an organism.

Although classical and operant conditioning are treated as separate models of human and animal behaviour, the sharp distinction is used as a matter of convenience only; in practice, the division between the two models is much less distinct with behaviour often consisting of a mixture of the two approaches.

To illustrate the transition from a passive stimulus-response model—in terms of both living organisms and neural networks—to a more active model involving exploration and evaluative feedback, these two contrasting behavioural models will be discussed.

3.1.2 Classical Conditioning

The first associative learning model, classical conditioning, is based upon the fact that certain innate or "hardwired" response patterns are already present in an organism from birth. Simple behavioural patterns are elicited by environmental cues without the intervention of "cognition" or internal modelling. Such stereotypical actions, known as *reflex actions* are responses to environmental stimuli which are not learned and follow a simple stimulus-response (S-R) pattern. They have arisen from years of evolutionary development and natural selection (Anderson, J. R., 1995; Barker, 1994). Some simple reflexes have the underlying mechanism in which a sensory neuron transmits a signal directly to a motor neuron via synapses; in other cases, one or more interneurons mediate between the motor and the sensory neurons.

The physiologist, Ivan Pavlov was particularly interested in the *salivary reflex* which he studied in dogs (Pavlov, 1928). The stimulus was invariably meat powder which was placed on the dogs' tongues and elicited a response of salivation. This is a typical example of an involuntary S-R pattern. The natural pairing of stimulus and response provides a basis upon which classical conditioning experiments are carried out, even today. The stimulus in such cases is known as the *unconditioned stimulus* (UCS) and is followed by an *unconditioned response* (UR). So, for the naturally occurring (unlearned) situation: UCS $\rightarrow$ UR.

If a stimulus, previously unconnected with the UCS, is paired with the UCS on a number of occasions, it is found that the new stimulus alone can elicit the response. The new stimulus is called the *conditioned stimulus* (CS) and the response elicited by the CS is known as the conditioned response (CR). In the case of Pavlov's experiments, the CS was a bell and the CR was salivation brought about by the bell. Three phases of conditioning can be distinguished (Barker, 1994) viz.

- the *initial phase*: UCS (taste) + CS (bell) → UR (salivation)

- the *training phase*: (a number of presentations)

  UCS (taste) + CS (bell) → UR (unconditioned salivation)

  \+ CR (conditioned salivation)

- the *testing phase*: CS (bell) → CR (conditioned salivation)

By training using the pattern UCS + CS → UR + CR we end up with the *association* CS → CR. Note that the pattern to be learned (the CR) is not presented to the test animal, it is originally elicited during a natural, inbuilt reflex. If the CS is continually presented without the UCS during conditioning, the response is observed to diminish; this is known as *extinction* (Anderson, J. R., 1995).

Extinction can be incorporated in artificial neural networks as a *forgetting factor* which allows associations between input and response patterns to diminish with time unless reaffirmed during use. The weakening of associations may be important in certain types of neural network which are operating in environments which change over time. Environments with time varying properties are known as *non-stationary*.

3.1.3 Operant Conditioning and Reinforcement

The ideas of active learning in organisms are taken further by the psychologist B. F. Skinner. Skinner developed a standardised methodology for carrying out learning experiments by controlling the environment and isolating a limited number of dependent variables (Barker, 1994).

The most widely known example of an artificial environment is the *Skinner box*: which comprises a laboratory apparatus in which an animal is caged for

conditioning experiments and which typically contains a lever that must be pressed by the animal to gain reward or avoid punishment

The learning theory studied using these experimental conditions is known as *operant conditioning*. A succinct definition of operant conditioning is given by Roberts (1993),

> "A form of conditioning in which learning takes place when reinforcement follows a person's or animal's spontaneous response; also known as instrumental conditioning. For example, a rat exploring its cage might press a lever, and find that a food pellet appears. It will then learn to press the lever in order to obtain food."

There are several key concepts embedded within this definition which are worth expanding upon to provide a basis for ideas developed in later sections when discussing autonomous artificial systems viz.

- *Operant:* This refers to any response which operates on the environment (Barker, 1994).

- *Conditioning:* This is not meant in the classical sense. Here it refers to the modification of internally generated behaviour patterns generated by stochastic search of the environment (exploration). During the course of time, certain behaviour patterns become more probable and others less so.

- *Reinforcement:* Reinforcement can be thought of as evaluative feedback from the environment. Reinforcement can be either *positive* or *negative*. Care must be taken with terminology to avoid confusion. Positive reinforcement can be identified with the idea of "reward" but negative reinforcement is not "punishment"; it is the avoidance of punishment when referring to psychobiological studies. A *reinforcer* is defined as any stimulus applied following a response which has the effect of increasing the probability of that response (Barker, 1994). Positive reinforcement enhances "approach" behaviour while negative reinforcement enhances "avoidance" behaviour. The

term reinforcement refers to the enhancement of behaviour not the informative "direction" (approach/avoid) or survival value. For Skinner, a reinforcement does not necessarily imply a reward. The term *punisher* is used to refer to a stimulus which reduces the probability of a given response; its survival value is identified with punishment.

- *Spontaneous response* refers to the internally generated behaviour patterns actively exhibited by an organism exploring its environment. These are not conditioned by presenting a stimulus to a passive recipient which then responds with an approximation to a desired response specified by the "trainer".

These points have provided, and will continue to provide, biological inspiration for the design of autonomous agents capable of learning about the world and actively adapting to environmental conditions with reduced operator prompting. The brief introduction to relevant psychological ideas presented here illustrates that neural networks and other autonomous systems may benefit from a study of psychobiological ideas. The biological world may, at least, provide inspiration for the design of "intelligent" autonomous agents; better still, it will furnish researchers with mechanisms which provides a basis for artificial counterparts.

In the sections dealing with reinforcement learning in artificial autonomous agents, the term "positive reinforcement" will refer to reward or probability enhancement and the term "negative reinforcement" will refer to punishment or probability reduction. This convention is used in the reinforcement learning literature and will be adhered to here. The contrasting use of terms must be borne in mind when comparing psychobiological and artificial neural network literature.

3.1.4 Shaping

Operant conditioning involves associative learning in which the desired behaviour or increasingly closer approximations to it are followed by a reinforcing stimulus; the animal receives reinforcement depending upon how it responds to aspects of

the controlled environment. Rewards (or punishment) become *associated* with individual actions or sequences of actions.

Classical conditioning does not depend upon the spontaneous generation of behaviour by an organism; instead the response is elicited by an externally applied stimulus which triggers a "hardwired" pattern of behaviour. Conditioning, in this case, consists of forming an association between the naturally occurring stimulus and an artificial stimulus.

In both conditioning methods, associative learning takes place through *shaping* of behaviour; this shaping depends upon a *reinforcement schedule* which is specified as a set of events and contingencies for each type of artificial environment.

Even in the controlled conditions of an artificial environment, both types of associative learning do not occur in isolation; this blurring of boundaries is even more pronounced in natural environments—especially in the case of humans—where learning takes place through association of instinctive urges with socially acceptable outlets, shaping of spontaneous responses with reward / punishment schedules and association of appropriate behavioural responses with environmental cues. Higher level associative learning also takes place where abstract ideas (concepts) are associated.

The ideas of experimental psychology provide motivation for the development of artificial learning systems such as neural networks. However, the study of artificial neural networks is a subject in its own right and does not exist solely to provide a set of abstract explanatory models for observed behaviour.

The division of conditioning into classical and operant modes has an artificial counterpart in neural network learning methods, namely supervised and reinforcement learning; unsupervised learning is more difficult to classify in this twofold scheme but possibly belongs in the second class—the artificial counterpart to operant conditioning—although it is difficult to see where the shaping of behaviour occurs here.

3.2 Automata

Reinforcement learning algorithms are a class of algorithms for *learning automata* (Zeidenburg, 1990). This section will form an introduction to the theory of automata. There are two general classes of automata, viz. *deterministic* and *stochastic*. Stochastic automata are of particular interest in this thesis but deterministic automata provide a convenient starting-point for discussion and generalise naturally to their stochastic counterparts. These two classes will be covered in sections 3.2.2 and 3.2.3. Section 3.2.4 will discuss the concept of learning as it applies to automata.

The theory of stochastic automata can be related meaningfully to the theory of operant conditioning in animals. At a basic level, animals can be modelled as stochastic automata which learn and adapt to the environment. When placed in a new (experimental) environment, animals will exhibit behavioural patterns from a repertoire of actions or action sequences having different probabilities. The relative frequencies of the occurrence of given actions can be changed over time with a reinforcement schedule based upon the theory operant conditioning. Seen from the point of view of stochastic automata which learn, the changes in action frequencies correspond to action probabilities altered by a learning algorithm exposed to training signals.

3.2.1 Introduction: Markov Decision Processes

A useful framework for the formulation of learning problems is that of *Markov Decision Processes* (*Markov Processes*) or *Controlled Markov Chains* (Bailey, 1964; Budnick, 1988; Watkins, 1989). Markov decision processes allow the representation of probabilistic behaviour in an organism or intelligent agent. In its simplest form a Markov decision process involves spontaneous outcomes with no external input. A definition of this form is given by Budnick (1988):

"A **Markov process** is a sequence of experiments in which each experiment has m possible outcomes $E_1, E_2, \ldots, E_m$ and the probability of each outcome depends only upon the outcome of the previous experiment."

Here, *outcome* means action or state and *experiment* a behavioural period or unit in which an action is performed out of a repertoire or action set. Informally, *state* is the current potential for an outcome or set of outcomes depending upon past experience or history. Past history is not stored explicitly, it is represented by the state of a system.

It is convenient to distinguish between states and actions; this is not done in the above definition which uses the general term of "outcomes". Watkins (1989) uses the distinction and it will be used henceforth in this thesis. In a Markov decision process there is a finite set of states, denoted here by S. The transitions between members of S are determined by a transition function, T. If state transitions are determined only by the previous state then the transition function, $T(\mathbf{s})$ where $\mathbf{s} \in S$ can be represented as a matrix of transition probabilities known as the *state transition matrix*. To calculate transition probabilities over a number of steps, the state transition matrix is multiplied by itself that number of times.

States can be thought of as an "internal" representation of behaviour, and actions as an "external" manifestation of behaviour. The finite set of actions is determined probabilistically by the system states when operated upon by an action function, denoted here by A. Figure 3.1 shows the situation schematically.

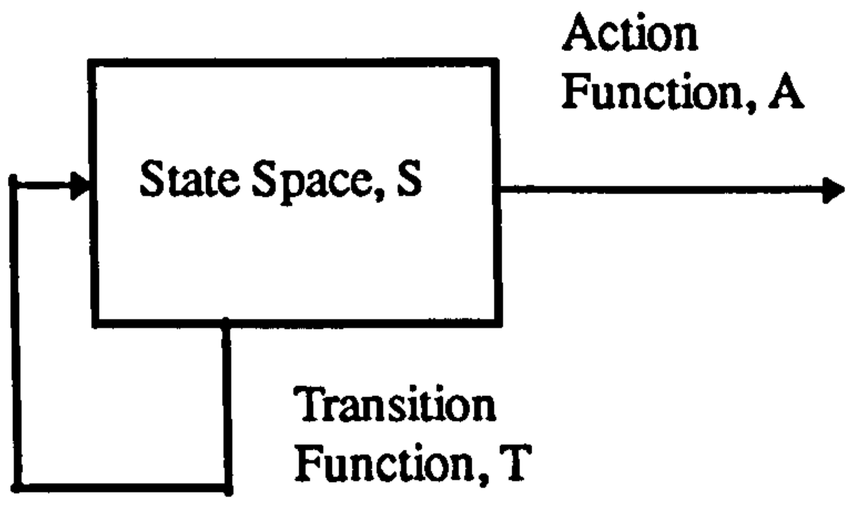


Figure 3.1 A schematic representation of a Markov process. The process changes state according to a transition function and elicits some action.

The previous description can be formalised to include the state-action distinction; following Watkins (1989) an extended definition is given. Thus, a *Markov decision process* consists of four parts:

- a finite state-space, *S*,
- an action function, *A*, which determines probabilistically the action at each discrete time step,
- a transition function, *T*, which determines probabilistically the transition between states of the process, and
- a reward function, *R*, which gives the, possibly probabilistic, reward at each time step.

Note that the extended definition also includes a reward function.

3.2.2 Deterministic Automata

Any system—for example, a neural network an environment and a plant—can be thought of as a black-box with a specified set of inputs and outputs. To simplify further the formulation of systems in terms of automata two conditions may be specified: Changes occur in discrete time and both the input and output sets are finite. The sets of inputs and outputs and discrete time instants, can be represented by X, Y and Z respectively.

So, far this abstract model says nothing about the relationship or mapping between X and Y as a function of Z. Set Z is required because a sequence of inputs from set X may not determine uniquely a sequence of outputs from set Y. If the output sequence is determined for a given input set, regardless of time, the system model is said to be *memoryless*.

For most systems, the input-output mapping depends upon the "history" of the system. For deterministic automata, the system history is represented by a set of *states*. A definition of state is given by (Arbib, 1987):

"The **state of a deterministic system** is some representation of the past activity of the system that is sufficiently detailed to serve as a basis together with the current inputs for determining what the next output and state will be."

The state does not give any information about how it was reached. Such information is redundant and each state provides a compact representation of a set of *equivalent histories* (Minsky, 1967).

The above description can be formalised to give a definition of an automaton (Arbib, 1987):

An *automaton* is specified by three sets X, Y, and S, and two functions T and A, where

(i) X is a finite set, the set of *inputs*;

(ii) Y is a finite set, the set of *outputs;*

(iii) S is the set of *states*; while

(iv) $T: S \times X \rightarrow S$, the *state-transition function* is such that if at any time t the system is in state s and receives input x, the at time t+1 the system will be in state $T(s,x)$: and

(v) $A: S \rightarrow Y$, the *output function*, is such that s always yields output $A(s)$.

The automaton is said to be *finite* if S is a finite set and *deterministic* if the state-transition function, T, uniquely determines the output when a given input is present. In other words, from any state, the future evolution is determined for a specified input sequence provided that state transitions are not random. The case of random state-transition functions is covered in section 3.2.3.

Neural networks, defined formally may be viewed as finite automata (Arbib, 1987). Conversely, the state dependent input-output mapping of a finite automaton can be replicated by a functionally equivalent neural network.

For a non-stochastic neural network, the equivalent deterministic state-transition function can be identified with the set (matrix) of weights, W^* following training. The final mapping $F^*: X \to Y, \quad \mathbf{x} \mapsto \mathbf{y} = F^*(\mathbf{x})$, uses W^* implicitly.

During training, F is also a function of W(t) which will be represented by W_t to show that the weight matrix is parameterised by the time instant, t. At any time instant, t during training, the input-output mapping can be represented by

$F_{W_t}: X \to Y$ where F is the functional $F(W_t, \mathbf{x})$. For $F^* = F(W^*, \mathbf{x})$ the weight matrix W^* is subsumed within the function as constant giving the function $F^*: \mathbf{x} \mapsto F^*(\mathbf{x})$

3.2.3 Stochastic Automata

If the state-transition function is probabilistic, that is, for a given state there exists a set of possible transitions which depend upon a set of associated probabilities, then the transition function does not uniquely determine the transitions and allows a stochastic search of the environment.

For a *stochastic automaton*, both (iv) and (v) of the automaton definition are modified to include the probabilistic nature of stochastic automata. Now, the state does not determine the transition (via the transition map $S \rightarrow S$) for a given input, it determines a set of probabilities governing the transition. A state $s \in S$ has a vector of probabilities, $\mathbf{p}$ associated with it which signify the transition probabilities, given the current state.

3.2.4 Learning Automata

Up to now, nothing has been said about how the state-transition mappings—whether deterministic or stochastic— are specified; indeed, the key idea of learning has been avoided. Learning is essential to the development of autonomous agents if they are to be sufficiently adaptable. Deterministic and stochastic automata may be used if the environment or plant model is known sufficiently well. Otherwise, more sophisticated methods, such as those of neurocontrol, are required. On-line adaptation of the state-transition function (or its neurocontroller equivalent) is carried out as more observations become available. With learning, little or no *a priori* knowledge may be necessary in order to develop a successful control strategy.

Without learning, for input patterns $x_{t-1}, x_t \in X$ at successive time instants $t, t+1 \in Z$ and states $s_i, s_j, s_k \in S$ a state transition can be represented formally by $s_j = T(s_i, x_{t-1}) \rightarrow s_k = T(s_j, x_t)$.

With learning, the set S and the function T are both dependent upon time. For this case, the set of states can be represented by S_{t-1}, S_t, S_{t+1} with the succession of states $s_i \in S_{t-1}, s_j \in S_t, s_k \in S_{t+1}$. Now the transition is represented by $s_j = T_{t-1}(s_i, x_{t-1}) \in S_t \rightarrow s_k = T_t(s_j, x_t) \in S_{t+1}$, where the state-transition function, T is parameterised by time. In some cases, the number of states may be fixed, or

may grow or reduce for some automata providing this is taken into account by the state-transition function.

For a stochastic automaton in which learning has occurred, for a given state s_i, $\mathbf{p}_i(t)$ is not necessarily equal to $\mathbf{p}_i(t+1)$. Automata which are capable of learning are called *learning automata*.

For a learning stochastic automaton, the main task is to "shape" probability distributions for both the state-transition function and the action (output) function respectively. For the latter, some actions will be made more probable, and others less so. The learning process of shaping probability distributions in a learning automaton can be viewed as the artificial counterpart of operant conditioning.

For a *stationary environment*, $S_\tau \rightarrow S^*$, where S^* is the final set of states which are optimal in some sense, that is, the set of states tends to a final set with respect to time ($\tau \rightarrow \infty$). This is a generalisation of the fixed point concept. More formally, $S_\tau = S_{\tau+1} = S^*, \quad \forall \tau \geq \tau^*$ where τ^* is some time instant. For a *non-stationary environment* no such optimal set of states exists.

For a stationary environment, an automaton has to solve an *optimisation problem* where the optimal state set is treated as a "fixed point". More importantly, the state-transition function—also treated as a "fixed point"— determines behaviour. This must be optimised. Formally, $T_\tau \rightarrow T^*, T^* = T^*(s,x)$ and $T_\tau(s,x) = T_{\tau+1}(s,x) = T^*(s,x), \forall \tau \geq \tau^*$. The state-transition function will be optimised through learning. For a non-stationary process, no such *absorbing* "state" exists where all state transition probabilities are fixed indicating an optimal state transition function for an animal or automaton.

3.2.5 What is Meant by 'Optimal'?

The notion of "optimal" behaviour needs to be clarified. Watkins (1989) distinguishes between two types of optimality in learning:

- *optimal learning* where the agent processes information in such a way that ensures that the "best" possible decisions are made at each stage of learning;

- *learning of efficient strategies* where the process of acquiring a strategy is not necessarily optimal but leads eventually to a maximally efficient strategy.

The second notion is weaker than the first and refers to an end-state as opposed to *overall* efficiency. Optimal learning refers to the learning method itself and does not necessarily lead to the acquisition of the maximally efficient strategy; the overall cost to the agent may be too great (Watkins, 1989).

Optimal behaviour, in whatever sense, involves a trade-off between *exploration* and *exploitation.* If too much time is spent in exploring the environment then little time is left for immediate exploitation of acquired learning. Conversely, if too little exploration is carried out then useful alternative behaviours may be missed and time may be wasted exploiting a second-rate strategy. This dilemma is known as the *exploration-exploitation trade-off* (Watkins, 1989). At each moment in time, an agent must decide whether to explore or to exploit. There are no hard and fast rules for this decision process.

The formalisation of optimal learning theory is difficult for two main reasons (Watkins,1989):

- the difficulty of devising proven optimal learning strategies for all but the simplest of artificial problems;

- the requirement of *a priori* assumptions pertaining to probability distributions of encountered environments.

3.2.6 The Road to Reinforcement Learning: Two Algorithms

A subject of this thesis is a particular control strategy of reinforcement learning using two outputs (bang-bang control). The neurocontroller used in this application can be viewed as a stochastic automaton. To prepare further for the later discussion, the idea of learning in stochastic automata will be expanded upon.

There exists a class of learning algorithms for stochastic automata (Zeidenburg, 1990). Here two algorithms of relevance to the discussion will be covered; these are the *linear reward-penalty* algorithm (Narendra and Thatachar, 1974) and the *associative reward-penalty* algorithm (Barto and Anandan, 1985).

The linear reward-penalty (L_{R-P}) deals with a simplified automaton, with a single state (no state-transition function), and only a simple reward-penalty signal as input. The learning problem involves convergence to a final set of action probabilities which specify a desired behavioural repertoire. Again, the connection with operant conditioning is apparent. The situation is illustrated schematically in Figure 3.2.

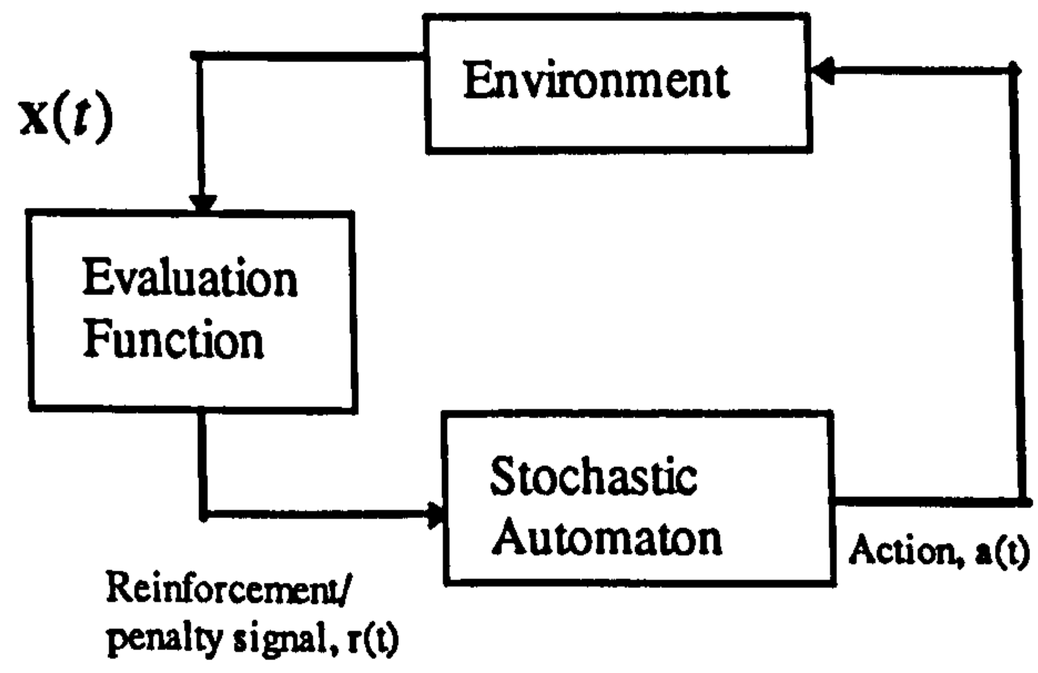


Figure 3.2. The operation of a learning automation in an environment. Actions are directed towards the environment from which feedback is obtained

The set of actions, A is defined as $A = \{a_1,\ldots,a_i,\ldots,a_m\}$ with an associated set of action probabilities $P = \{p_1,\ldots,p_i,\ldots,p_m\}$. This system can be thought of as a stochastic neural network with a set of m nodes. The operation of this system provides a simple model of operant conditioning.

At time instant t, the reinforcement signal r(t) of +1 or -1 is applied to the automaton and action $a(t) = a_i \in A$ is elicited. For r(t)=1 the learning rule is

$p_i(t+1) = p_i(t) + \lambda_R\left(1 - p_i(t)\right)$ for the "winning" action and

$p_j(t+1) = \left(1 - \lambda_R\right)p_j(t)$ for the rest.

For the reward part of the learning rule, the set of action probabilities always sums to unity. For the "punishment" part of the learning rule, r(t)=-1,

$$p_i(t+1) = \left(1 - \lambda_P\right)p_i(t)$$

$$p_j(t+1) = \frac{\lambda_P}{m-1} + \left(1 - \lambda_P\right)p_j(t)$$

It can be shown similarly for the punishment case, that the set of action probabilities sums to unity.

The associative reward-penalty algorithm (A_{R-P}) of Barto and Anandan (1985) extends the utility of the L_{R-P} driven stochastic automata by allowing the *association* of output actions with input vectors from the environment; this is shown schematically in Figure 3.3.

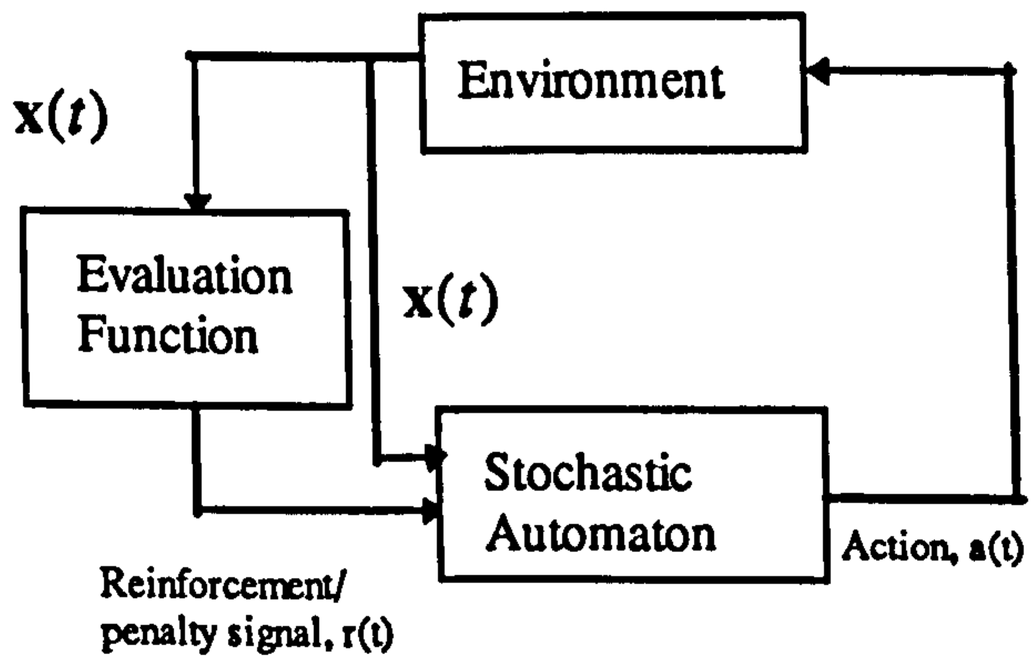


Figure 3.3 The Associative Reward-Penalty automaton of Barto and Anandan (1985).

The automaton can now learn an input-output mapping and associate appropriate actions with stimuli from the environment. The automaton is "taught" to make associations by an evaluation function (critic) which emits reward / penalty signals based upon the current input vector and the most recent action. Formally,

$$r(t) = d\big(\mathbf{x}(t), a(t)\big), \quad d: X \times A \rightarrow \{-1, +1\}.$$

The actual details of the associative reward-penalty algorithm are not important here and are documented elsewhere (Barto and Anandan, 1985) where it is shown that A_{R-P} reduces to L_{R-P} under certain conditions; variations to the basic algorithm can also be found there.

So far, the stochastic automata of Figures 3.2 and 3.3 have been treated as black boxes with no mention of the internal states. It is not the intention of this thesis to discuss these matters further but suffice to say that the black box will be "opened" when discussing the reinforcement-learning based neurocontroller. Section 3.2 has indicated the possibility of automata (or equivalent neural networks) which learn to associate actions with input vectors from an environment and do not require the specification of desired outputs (actions) as with supervised learning. The main advance here is the use of an evaluation function or critic which determines the type of reinforcement (reward or punishment) administered to the learning automaton.

The critic network must be examined next in some detail if the possible accusation that the learning problem has been merely re-located is to be refuted.

3.3 Michie and Chambers' Boxes

It has been stated that to develop a controller, knowledge of system dynamics is required. Furthermore, in addition to the accurate dynamical model, knowledge of the desired system behaviour is needed, usually in the form of an *objective* or *cost* function (Anderson, C. W., 1989). This knowledge is often either unavailable or difficult to obtain. Candidate autonomous neurocontrollers must be able to operate without such *a priori* knowledge and formulate a control strategy on-line given plant input-output data as it arises.

3.3.1 Introduction

Reinforcement learning applied to the *cart-pole* or *inverted pendulum* problem exemplifies the control problem of applying a "naïve" neurocontroller directly to an opaque (black-box) dynamical system. The inverted pendulum problem is an unstable system with dynamics of fundamental importance to the idea of maintaining balance in, for example, walking systems or rocketry. (Anderson, C. W., 1989).

When treating a dynamical system as a black box, using reinforcement learning, the only information available is a state vector, which forms the neurocontroller input and a punishment signal which signifies when control has failed. The desired intermediate control action for each system state is unknown. There is also no explicit objective or cost function to shape controller performance. Neurocontroller learning is based upon failure signals alone.

It should be apparent that temporal information is important for such a control problem and that delays will play a part in the learning of long sequences of actions required to avoid failure. Delayed input information is not available to the neurocontroller which has to apportion "blame" over a sequence of actions depending upon individual "responsibility". This is the *assignment of credit* problem (Anderson, C. W., 1989).

The formulation of reinforcement learning considered in this thesis is that of Barto Sutton and Anderson (1983) which will henceforth be referred to as the BSA model for convenience. The BSA reinforcement learning system provides the basis for an autonomous neurocontroller architecture (Barto, Sutton and Anderson, 1983) which has provided inspiration for a number of modified architectures including those comprising the subject of the remainder of this thesis. Some of the other BSA variants will be covered briefly in section 4.1 together with a few alternative control methods.

3.3.2 The Cart-Pole problem

The starting point for the BSA formulation of reinforcement learning (Barto *et al*, 1983) is the "boxes" adaptive problem solving system of Michie and Chambers (1968a). As the boxes learning system forms the basis for the evolution of the present work, it will be described briefly here.

Following Michie and Chambers (1968a) and Barto *et al* (1983) the cart-pole system problem is used to exemplify some of the characteristics that distinguish neural networks as autonomous learning systems from other available data processing methods. The characteristics of autonomy and adaptability are among the most important. As a test problem, the cart-pole system provides an example of a highly non-linear system involving the characterisation of complex state-space trajectories. Standard solution methods require assumptions about the form of the control force function and an objective function (Anderson, C. W. 1989; Hocking, 1991). Furthermore, such techniques rarely generalise and, thus, require an *a priori* analysis of each dynamical system encountered. Like Barto *et al* (1983), it is assumed in this thesis that the available feedback is of much lower quality than is required for both standard control techniques and for supervised learning techniques. Furthermore, it is believed that similar assumptions can be made about the state-space partitioning problem where any autonomous system will have limited information about the structure of state-space in advance of

experience. Indeed, merely specifying a fixed partitioning *a priori* makes assumptions about the granularity of the resultant control mapping and constrains the available adaptive procedures within a pre-specified temporospatial structure.

The problem posed by Michie and Chambers to illustrate the boxes adaptive learning system consists of a cart constrained to move along a one dimensional track with a pole attached to it. This is illustrated in Figure 3.4. The movement of the pole is constrained within the vertical plane and is represented by the state variables θ and $\dot{\theta}$ signifying the angle of the pole from the vertical and the angular speed of the pole respectively. The movement of the cart is controlled by an impulse force (bang-bang control) in either direction and is represented by the state variables x and $\dot{x}$ which signify the distance from the origin (centre) of the track and the speed of the cart respectively. Thus, there are four state variables representing the whole motion of the cart-pole system. System parameters are given in appendix F which also specifies the physical system and computer simulation details.

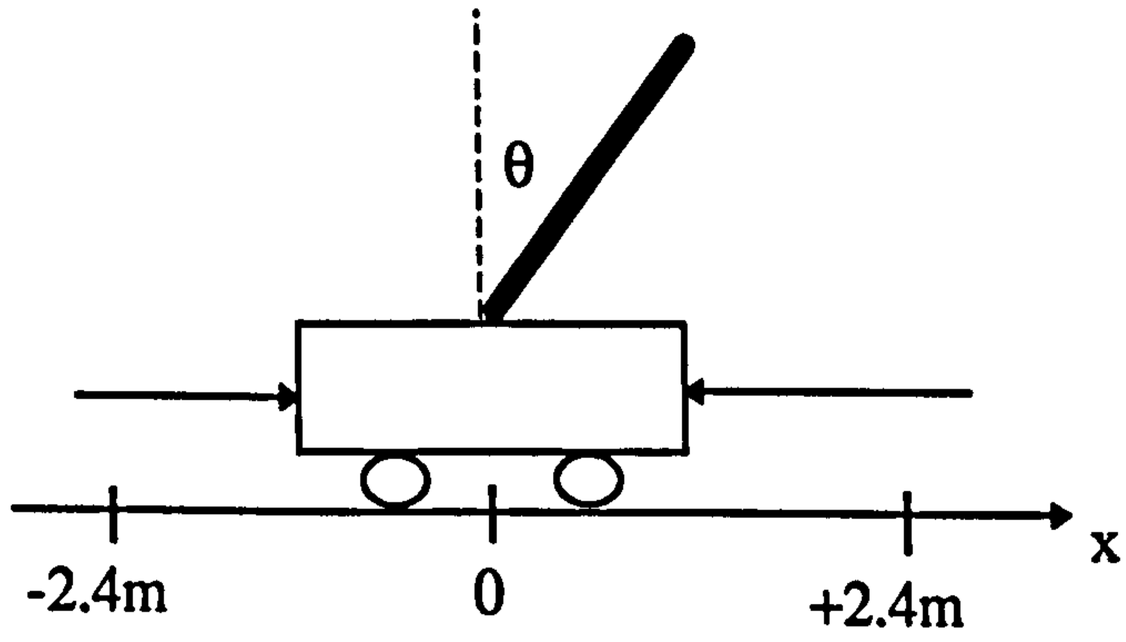


Figure 3.4. The cart-pole system. Motion is constrained within the vertical plane. See the body of the text for details.

Information from the physical system simulation is minimal and does not provide stimulus-response pairs consisting of inputs and desired outputs to be associated. Only the state vector and a coarse failure signal, reflecting the cart-pole status, are supplied to the control system. If the pole falls or the cart hits the track boundaries then a failure signal is sent to the controller and the cart-pole system is reset to its initial conditions to begin a new trial.

3.3.3 Boxes

Under these conditions, the credit assignment problem becomes apparent; there are difficulties concerning the assignment of credit (blame) to individual control actions which, taken together, comprise the state-space trajectory which leads to failure and thus the final failure signal (Barto *et al*, 1983). The boxes system (Michie and Chambers, 1968a) partitions state-space into 225 non-overlapping regions or boxes by quantising the state variables; note that this partitioning is fixed *ab initio* in both the boxes and the BSA systems. Each individual region is independent and is said to contain a local "demon" (Selfridge, 1959) which has to choose a control action of ±N Newtons whenever the state-space trajectory enters the local region.

A global demon has overall control; its task is to decode the state vector, assign its trajectory to individual regions and distribute the failure signal to the local demons. Left/right force decisions are taken on the basis of the utility of these decisions calculated from past failure signals weighted by the time interval from box entry to failure for a given run. Thus, the expected lifetimes of a left or right decision determine the box output at any particular time and the temporally weighted effect of failure on the system is fed back to compute new left/right decision expected lifetimes. The full formulation of the boxes learning system is found in Michie and Chambers, (1968a).

3.3.4 Linearisation

Standard state-space methods can be used to obtain a linear model as an approximation to a non-linear system and to design a closed-loop feedback controller (via pole placement) to control the system within a limited region of state-space (e.g. Friedland, 1987). This control method requires an *a priori* model of the dynamical system, obtained by using the simplification of linearisation to render the problem amenable to linear techniques (e.g. Wiberg, 1971; Banks, 1986). More sophisticated approaches using feedback linearisation,

for example, may extend the neighbourhood of effective control but are still highly dependent on accurate *a priori* models. For many desirable control purposes, however, such models may not be available or may contain too many analytical simplifications which render the proposed control system incapable of following the complex dynamics of the real system under consideration.

To simplify, friction can be neglected, i.e. $\mu_c = \mu_p = 0$. The frictionless equations are linearised by assuming that $\theta(t)$ and $\dot{\theta}(t)$ are small; these assumptions are reasonable given that the pole is to be balanced around an equilibrium point of $\theta = \dot{\theta} = 0$. The simplifications $\sin(\theta) \approx \theta$, $\cos(\theta) \approx 1$ and $\dot{\theta}^2$ can be substituted into equations (F1) and (F2) of Appendix F to give

$$\ddot{\theta} = \frac{g\theta - \dfrac{F}{m_c + m}}{l\left[\dfrac{4}{3} - \dfrac{m}{m_c + m}\right]} \text{ and } \ddot{x} = \frac{F - ml\ddot{\theta}}{m_c + m}. \text{ Rearranging gives}$$

$$\ddot{\theta} = \frac{3(m_c + m)}{l(4m_c + m)} g\theta - \frac{3}{l(4m_c + m)} F \text{ and } \ddot{x} = -\frac{ml}{m_c + m}\ddot{\theta} + \frac{1}{m_c + m} F.$$

Substitution gives $\ddot{x} = -\dfrac{3mg}{(4m_c + m)}\theta + \dfrac{4}{(4m_c + m)} F$. Letting $x_1 = x$, $x_2 = \dot{x}$, $x_3 = \theta$, and $x_4 = \dot{\theta}$, the dynamical equations can be put in matrix form, thus:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\dfrac{3mg}{l(4m_c + m)} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \dfrac{3(m_c + m)g}{l(4m_c + m)} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{4}{(4m_c + m)} \\ 0 \\ -\dfrac{4}{(4m_c + m)} \end{bmatrix} F$$

which is in the standard linear state-space form, $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$ for which a linear controller can be developed (e.g. Friedland, 1987, Ogata, 1990). The intention here, in this thesis, is not simply to develop another controller for a particular non-linear control problem; it is to explore some of the issues for which the cart-pole problem provides a convenient example and to indicate the possibilities of developing flexible, general purpose controllers capable of adapting to a given dynamical system with a minimum of *a priori* information.

3.4 Reinforcement Signals and Traces

3.4.1 Reinforcement

The only feedback information available for neurocontroller learning is a failure signal which is triggered when the state vector crosses preset failure boundaries. For the BSA formulation, the preset failure limits are $\pm 12^{\circ}$ and $\pm 2.4M$. If the pendulum or cart exceed their respective bounds, then a punishment signal is fed back to the neurocontroller.

The reinforcement signal at time, t, denoted by $r(t)$, is characterised by,

$$r(t) = \begin{cases} -1 & \text{when failure occurs} \\ 0 & \text{otherwise} \end{cases}$$

It will become clear that failure alone is an inadequate training signal. This inadequacy is corrected in the original BSA version of reinforcement learning by using *predicted* reinforcement to provide "reward" or positive reinforcement to enhance learning and reduce learning time.

The BSA learning system selects a control action for a given state at each time step. The neurocontroller attempts to learn through experience which action is appropriate for which state and associates state-action pairs in an associative memory network (Figure 3.5).

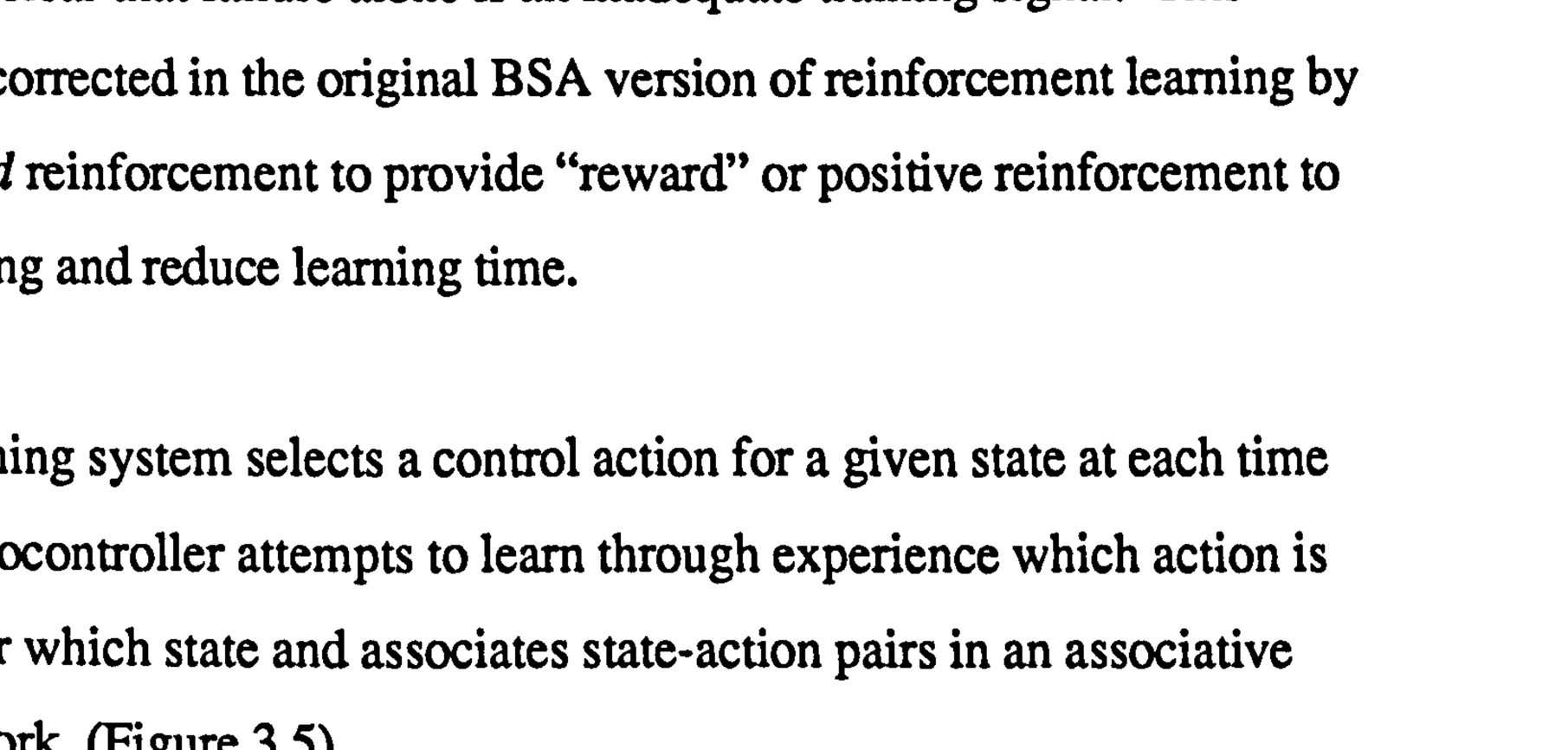


Figure 3.5. A neurocontroller based upon reinforcement learning. Internal details of the neurocontroller will be covered in later sections.

3.4.2 Traces

The problem of delay learning has been mentioned in Chapter 1. There are difficulties in training neural networks to associate input and output patterns owing to system memory. The approach of Hebb (1949) is simplistic in that inputs and outputs are associated by the instantaneous correlation of neural activities. Control of dynamical systems by neural networks often requires that delays are taken into account by the correlation of delayed inputs and outputs.

The boxes system of Michie and Chambers (1968a) circumvents the delay problem by recording what control action was used and when in the form of a "tally". A more biologically plausible system was proposed by Klopf (1972, 1986, 1988) and Sutton and Barto (1981) which postulated the existence of neural activity traces known as *eligibility traces*. These traces are said to indicate when a synapse (weight) is eligible for modification (Levine, 1991).

Incorporation of this mechanism into neural network architectures means that neural activities may be correlated in time, that is, a single input can still influence subsequent behaviour if weighted connections between neurons remain eligible for update after the input has been removed. Eligibility traces are an integral part of Barto, Sutton and Anderson's (1983) reinforcement learning system.

3.5 Temporal Difference Learning

3.5.1 Sequence Prediction

One of the objectives of developing autonomous learning systems is to be able to treat any environment (plant) as a black box and predict future behaviour. Prediction is the most basic form of learning (Sutton, 1988) and is fundamental to survival. Prediction of environmental characteristics arises from the need to establish the utility of different regions of the problem space and to associate appropriate actions with those regions. Often, a heuristic search of problem space is carried out by an intelligent agent to build up an internal representation of salient features.

Autonomous learning also implies an ability to take training examples directly from the "stream of experience"—that is, on-line or causal learning—without the help of a teacher or supervisor. A possible solution to the on-line prediction problem is that of *Temporal difference learning* (TD) which Sutton (1988) defines as "...a class of incremental learning procedures specialised for prediction problems." Temporal difference learning is a subset of the reinforcement learning paradigm; the key concept is that of the temporal difference between successive predictions, hence the name. Two advantages of temporal difference learning are that:

- learning is incremental; handling one piece of data at a time makes computations easier, and
- time delayed data does not have to be stored.

Early approaches to TD learning include Samuel's checkers player (Samuel, 1959) and Barto, Sutton and Anderson's ASE/ACE system (Barto, Sutton and Anderson, 1983); the latter being the motivation for a large part of the subject matter of this thesis. TD methods have also been proposed as models of classical conditioning (e.g. Barto and Sutton, 1982; Sutton and Barto, 1981, 1990; Klopf, 1988). The operation and utility of temporal difference learning methods will be

covered in section 3.5.2; the remainder of this section will provide some background and motivate the subsequent discussion.

TD learning attempts to move interest in artificial and natural learning methods away from the dominant supervised learning paradigm. Supervised learning methods are used extensively for training neural networks (e.g. Haykin, 1994). As discussed in section 1.5, pattern pairs, consisting of an input, x and a desired or actual output, y are presented, often repeatedly, to a neural network. A training set containing numerous examples is used to train a neural network to construct an input output mapping. After training, an input, x is presented to the trained network and elicits a response, $\hat{y}$ predicted by the stored associative mapping. This is a form of system identification and has proved effective for straightforward associative pattern matching.

Problems arise with the supervised learning method when temporal effects have to be taken into account (Myers, 1992). Prediction data is often in the form of a sequence of temporally related events or experiments $E_1, E_2, \ldots, E_N$ such as a time-series. System identification in this case is concerned with discovering the dynamical laws underlying a process to enable prediction and control Techniques using *recurrent neural networks* have been developed to deal with time delays and temporal sequences of events (e.g Elman 1990) but these often involve complex algorithms or network structures.

The sequential learning problem may be cast in terms of supervised learning by treating successive members of a temporal sequence as input-output patterns, thus (E_n, E_{n+1}) forms a set of training patterns. The more general form, $(E_{n-k}, \ldots, E_n, \ldots, E_{n+1})$ where the first k members form an extended input vector, is used to take account of delays.

There is a more fundamental problem with delays; when attempting to control dynamical systems, a sequence of events or control actions may lead to a final outcome or goal where intermediate stages are of little or no importance until the

goal is reached. In such cases, learning cannot take place until the final outcome so that credit (or blame) may be assigned to each of the preceding actions. This can be represented by $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \mathbf{y}_{n+1})$, where $\mathbf{x}_n$ is a vector of observations or control actions and y is the final outcome of a process. This is the *multistep prediction problem* (Sutton, 1988).

A putative learning system will be required to produces a series of predictions $\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_n$ which approximate $\mathbf{y}_{n+1} = \mathbf{y}$. This "end state" problem can also be framed in terms of supervised learning by specifying pattern pairs $(\mathbf{x}_n, \mathbf{y})$. The weights are updated at the end of a temporal sequence; this can be written as

$$w_{n+1} = w_0 + \sum_{t=1}^{n} \Delta w_t \qquad (3.1)$$

where

$$\Delta w_t = \alpha(\mathbf{y} - \hat{\mathbf{y}}_t)\nabla_w \hat{\mathbf{y}}_t \qquad (3.2)$$

(Sutton, 1988). Here, t denotes the time label of all intermediate weight changes prior to the final update at the end of a temporal sequence.

Taking a linear estimator of the form $\hat{\mathbf{y}}_t = \mathbf{w}^t \mathbf{x}_t$ and substituting into equation (3.2) gives the simplest case of an update rule (Sutton, 1988)

$\Delta w_t = \alpha(\mathbf{y} - \mathbf{w}^t \mathbf{x}_t)\mathbf{x}_t$ which is the Widrow-Hoff or delta rule used in the adaline (Widrow, 1960, Widrow and Hoff, 1963). A similar form can be used for the multilayer Perceptron where gradient information is backpropagated through one or more hidden layers.

3.5.2 Temporal Difference Learning

Equation (3.1) can be reformulated in terms of successive predictions, $\mathbf{y}_t$ and $\mathbf{y}_{t+1}$ with

$$\Delta w_t = \alpha(\hat{\mathbf{y}}_{t+1} - \hat{\mathbf{y}}_t)\sum_{k=1}^{t} \nabla_w \hat{\mathbf{y}}_k \qquad (3.3)$$

Details of the derivation leading to Equation (3.3) can be found in Sutton (1988). The weight update can now be computed incrementally as Δw_t only depends upon successive predictions (TD). The advantage is that values of a temporal sequence are not stored.

The key feature of temporal difference learning is that it is the changes in successive predictions which drive learning and not the overall error between the predicted and an actual or desired outcome. Equation (3.3) uses the implicit assumption that the past predictions in the summation are weighted equally. A more general form of the weight update equation is given by

$$\Delta w_t = \alpha\left(\hat{y}_{t+1} - \hat{y}_t\right)\sum_{k=1}^{t} \lambda^{t-k} \nabla_w \hat{y}_k \qquad (3.4)$$

and includes an exponential weighting factor λ, where $0 \leq \lambda \leq 1$ (Sutton, 1988). More recent predictions are weighted more strongly which is in accord with the idea of stimulus traces (see sub-section 3.4.2).

The weighting factor parameterises a family of learning procedures denoted by TD(λ) of the form given in equation (3.4); equation (3.3) is a special case TD(1). Defining,

$e_{t+1} = \sum_{k=1}^{t+1} \lambda^{t+1-k} \nabla_w \hat{y}_k$ gives $\Delta w_t = \alpha\left(\hat{y}_{t+1} - \hat{y}_t\right) e_t$ which leads to the recursive form $e_{t+1} = \nabla_w \hat{y}_{t+1} + \lambda e_t$

(Sutton, 1988).

For λ=0, $\Delta w_t = \alpha\left(\hat{y}_{t+1} - \hat{y}_t\right)\nabla_w \hat{y}_t$ which is similar in form to the adaline (Widrow-Hoff) learning rule but successive predictions are used.

The convergence of TD(0) for absorbing Markov processes is proved in Sutton (1988). An absorbing Markov process has a well-defined end-state. The prediction problems discussed so far assume definite outcomes at the end of a sequence but this is not always the case; indeed, for the cart-pole problem, the desired "outcome" of success demands longer and longer temporal sequences of

states and control outputs as balancing becomes more and more successful. Finite sequences terminated by failure are to be avoided. Prediction problems involving potentially infinite temporal sequences—sequences with no well-defined outcome—are called *infinite-horizon problems* (Sutton, 1988). In such cases, success (failure) is measured by associated costs generated by an environment or process.

Sutton (1988) defines a discounted sum of future costs

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \cdot$$

Defining a predicted future cost and assuming that it is accurate gives

$$\hat{R}_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = r_{t+1} + \sum_{k=1}^{\infty} \gamma^k r_{t+k+1} = r_{t+1} + \sum_{k=0}^{\infty} \gamma^{k+1} r_{t+k+2} = r_t + \gamma \hat{R}_{t+1}$$

For the recursive equation $\hat{R}_t = r_t + \gamma \hat{R}_{t+1}$ it can be assumed that $\hat{R}_t \neq r_t + \gamma \hat{R}_{t+1}$ until convergence and so an error ε can be defined in place of the predictive difference $\left(\hat{y}_{t+1} - \hat{y}_t\right)$ of equation (3.4). Defining the error $\varepsilon = r_{t+1} + \gamma \hat{R}_{t+1} - \hat{R}_t$ gives,

$$\Delta w_t = \alpha\left(r_{t+1} + \gamma R_{t+1} - R_t\right) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w \hat{R}_k \qquad (3.5)$$

as the equivalent weight update rule to equation (3.4) for infinite horizon problems (Sutton, 1988). In this thesis, this form of temporal difference learning equation is referred to as TDIH(λ) to distinguish it from the finite horizon version TD(λ).

For λ=0

$$\Delta w_t = \alpha\left(r_{t+1} + \gamma \hat{R}_{t+1} - \hat{R}_t\right) \nabla_w \hat{R}_k \qquad (3.6)$$

Equation (3.6) is a special case of TDIH(λ), denoted by TDIH(0) which relies only upon successive predictions of reinforcement; an example of TDIH(0) is that of Barto's Sutton's and Anderson's reinforcement learning system (1983) which is the subject of section 3.6.

3.5.3 Q-Learning

Q-learning is a form of reinforcement learning derived from dynamic programming; it is model-free and enables autonomous agents to discover optimal behavioural strategies in Markovian environments (Watkins, 1989; Sutton, Barto and Williams, 1992; Watkins and Daynan, 1992). Q-learning is similar to temporal difference learning in that an agent acts, evaluates the consequences of a particular action immediately (reward or penalty), and proceeds to estimate the value of the subsequent state. A Q-learning agent estimates a real valued function of the current state and action, known as the *value function,* which represents the total expected discounted future reward (Q-value). The objective of Q-learning is to estimate the Q-values for an optimal policy (Watkins and Daynan, 1992).

Q-learning systems have formally proven learning capabilities (Watkins, 1989; Watkins and Daynan, 1992;Sutton, Barto, and Williams, 1992). An agent using Q-learning explores state-space by trying out its repertoire of actions; it builds a map of state values based upon the expected long-term discounted reward.

The main difference between actor-critic and Q-learning systems is that actor-critic learning systems have two distinct sub-systems—one for estimating the long-term utility of each state and one for choosing the optimal action for each state—and compute state and action utilities separately whereas Q-learning systems maintain estimates of combined state-action pair utilities. Q-learning, thus, combines the operations of the actor and critic sub-systems.

Q-learning is said to be conceptually simpler, have a better-developed theory and has been found to converge faster in a number of cases, than reinforcement learning (Sutton, Barto and Williams, 1992). However, the implementation of reinforcement learning of Barto, Sutton, and Anderson (1983) is suited to the incremental structure of ART-based networks. Q-learning will not be considered henceforth in this thesis; further details will be found in Watkins, 1989; Watkins and Daynan, 1992; Sutton, Barto, and Williams, 1992.

3.6 The BSA Reinforcement Learning System

3.6.1 The Associative Search Element (ASE)

The BSA implementation (Barto *et al*, 1983) uses the following quantisation of state-space:

i) x: $-2.4m \le x < -0.8m \le x \le +0.8m < x \le +2.4m$,

ii) θ: $-12° \le \theta < -6° \le \theta < -1° \le \theta < +1 \le \theta < +6° \le \theta \le +12°$,

iii) $\dot{x}$: $\dot{x} < -0.5m/s \le \dot{x} \le +0.5m/s < \dot{x}$

iv) $\dot{\theta}$: $\dot{\theta} < -50°/s \le \dot{\theta} \le +50°/s < \dot{\theta}$

This collection of intervals results in a state-space partition of 162 distinct regions. A decoder system (see Figure 3.6) assigns a unique output line to each state-space region. This set of decoder outputs forms the unit input vector to the single ASE processing element. During processing, a state vector enters the decoder which switches on the appropriate input line to the ASE which subsequently issues a control action depending upon the current system state.

To avoid confusion between the original ASE /ACE notation and the original ART notation, the ASE / ACE notation has been modified and consequently differs from that used in the original paper of Barto *et al*, (1983).

The ASE control output is computed by

$$y(t) = f\left[\sum_{i=1}^{n} z_i(t)x_i(t) + \varepsilon(t)\right] \qquad (3.7)$$

where $y(t)$ is the output at time t, $z_i(t)$ is the scalar weight value of the i^{th} ASE input line at time t, $x_i(t)$ is the activation of the i^{th} ASE input line, $\varepsilon(t) \sim N(0,1)$ is Gaussian noise derived from a zero mean source with unit variance and

$$f(x) = \begin{cases} +1 & for\ x \ge 0 \\ -1 & for\ x < 0 \end{cases}$$

gives the activation function of the ASE element which signifies the right and left control actions respectively.

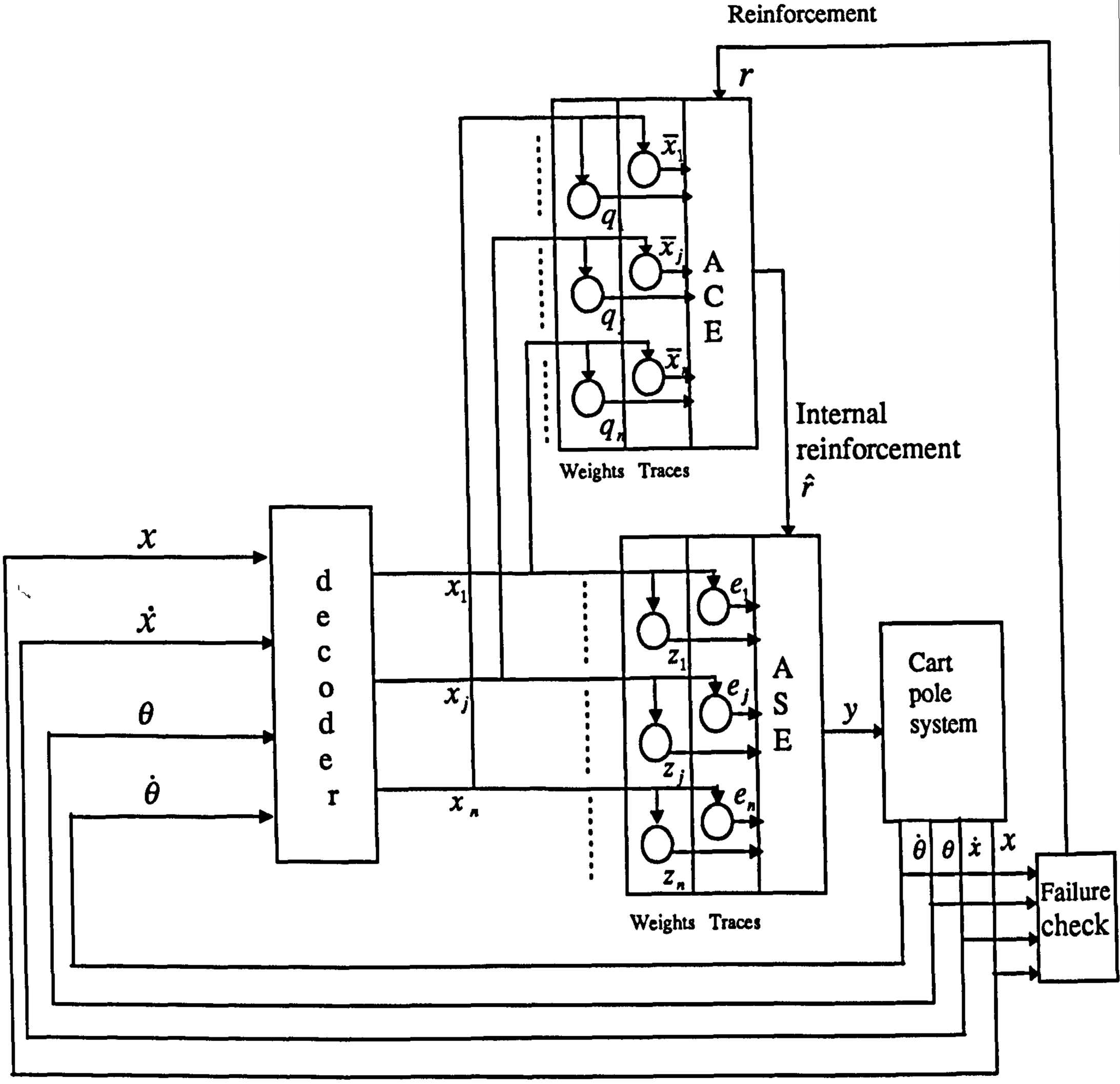


Figure 3.6. A neurocontroller based upon reinforcement learning. Both the original associative search element (ASE) and the adaptive critic element (ACE) of Barto *et al* (1983) have been retained. The independence of the decoder from the ASE / ACE subsystems makes it a focus for possible modifications (After Barto *et al.*, 1983).

The BSA implementation uses a standard basis of 162 unit vectors of 162 entries; when the i^{th} input line is active, the basis vector signifying the ASE input vector consists of all zero entries except for a "one" at the i^{th} entry. The decoder is a sub-system of the whole control system which lends itself to useful modification. This allows the properties of the controller to be modified whilst retaining the

functionality of the ASE and ACE sub-units. Various methods of state-space partitioning become possible (e.g. Lin and Kim, 1991) including self-organisation through experience as considered in this thesis. Thus, the *a priori* partitioning of state-space, as given in the original formulation, is a sufficient but not a necessary condition for using the ASE / ACE system.

From equation (3.7) it can be seen that, at a given time, τ

$$y(\tau) = f[z_k(\tau) + \varepsilon(\tau)] \qquad k \in \{1,\ldots,162\}$$

where k is the index of the input line. The weight, $z_i(\tau)$ signifies the direction in which the control force is applied at time, τ depending on the result when added to $\varepsilon(\tau)$ the random perturbation also at time, τ.

The ASE weight evolution equation, for the i^{th} input line is given by

$$z_i(t+1) = z_i(t) + a\hat{r}(t)e_i(t), \qquad (3.8)$$

where $\hat{r}(t)$ is the real valued reinforcement at time t, $e_i(t)$ is the "eligibility" at time t of input pathway i and a is the positive rate of change constant for z_i which determines the magnitude of change in z_i with respect to the reinforcement signal. The term 'reinforcement' has already been mentioned and, for the ASE unit operating alone, is given the value of 0 throughout a trial until failure occurs when it becomes equal to -1.

Eligibility is derived from the work of Klopf (Klopf, 1986, 1988) and represents the temporal weighting of the reinforcement signal in the derivation of the weight change. In a series of modifications to the Hebbian model (Hebb, 1949), Klopf suggests that, "instead of correlating approximately simultaneous pre- and post-synaptic signal levels, earlier pre-synaptic signal levels should be correlated with later post-synaptic signal levels." (Klopf, 1988). Klopf considers *changes* in levels to be more important but here we are concerned with the signal levels and delay effects. This is consistent with a solution of the credit assignment problem which requires temporally adjusted weight updates for distributing credit or blame

to state-space partitions traversed by an evolving state-space trajectory. The eligibility update equation is given by:

$$e_i(t+1) = \delta e_i(t) + (1-\delta) y(t) x_i(t) \qquad (3.9)$$

where δ, $0 \le \delta < 1$ is a constant determining the eligibility trace decay rate. This linear difference equation gives an exponentially decaying eligibility trace which maximally contributes to weight updates when the given input line is activated recently with respect to the reinforcement signal. Without stimulation via conjunction of pre- and post-synaptic activity reflected in equation (3.9), the eligibility trace passively decays. This is Hebbian learning (Hebb, 1949) with passive decay. The inclusion of the term *y(t)* ensures that information regarding the direction of the force is included in the weighting which reflects the expected lifetime and desirability of a particular control force. Consequently, actions which were made relatively long ago, with respect to eventual failure, merit little change to their expected lifetimes and, thus, exert little influence on the outcome.

3.6.2 A Non-linear Evaluation Function

The ASE element forms the *action network* and is formally equivalent to the boxes system. In theory, this action system could implement a linear neurocontroller without quantization of the state-space. A linear neurocontroller using the adaline element learned to balance the pole using operator modelling (Widrow and Smith, 1963). For an autonomous system—not using supervised learning or operator modelling—this would require an *evaluation function* which evaluated the consequences of each action on-line.

In practice, the ASE could not learn such a linear control mapping without knowing the desired output for each input state using a *linear* evaluation function because no such linear function exists. The evaluation function would have to be non-linear (Anderson, C. W., 1989) and so precludes the use of linear neural networks to develop neurocontrollers without pre-processing. The BSA implementation uses linear neural elements but does not violate the non-linearity

requirement because the quantization of state-space is a form of pre-processing which transforms the original variables into a form which allows a single linear element to solve the control problem. The evaluation function of the BSA system uses the quantization of state-space and constructs a look-up table of system states (Boxes) and their current evaluations regarding reinforcement. Using an evaluation function allows a continuous reinforcement signal to be used instead of the crude failure signal. The more informative signal consequently improves the learning rate.

Other methods of solving the control problem will be discussed in section 4.2.10 onwards including Anderson's non-linear action and control elements, each consisting of two layers (Anderson, C. W., 1989).

That any evaluation function for the cart-pole system is non-linear can be seen by examining the angle failure criterion alone (Anderson, C. W., 1989). Consider an evaluation function using the BSA failure criterion of $\pm 12^\circ$ and using positive and negative reinforcement of +1 and -1 respectively at the extremes. Traversing the evaluation function angular range between failure at -12° through the "successful" region to failure at $+12^\circ$ indicates that a linear function (hyperplane) to solve the problem does not exist. Figure 3.7 shows a simple hypothetical evaluation function which is clearly non-linear.

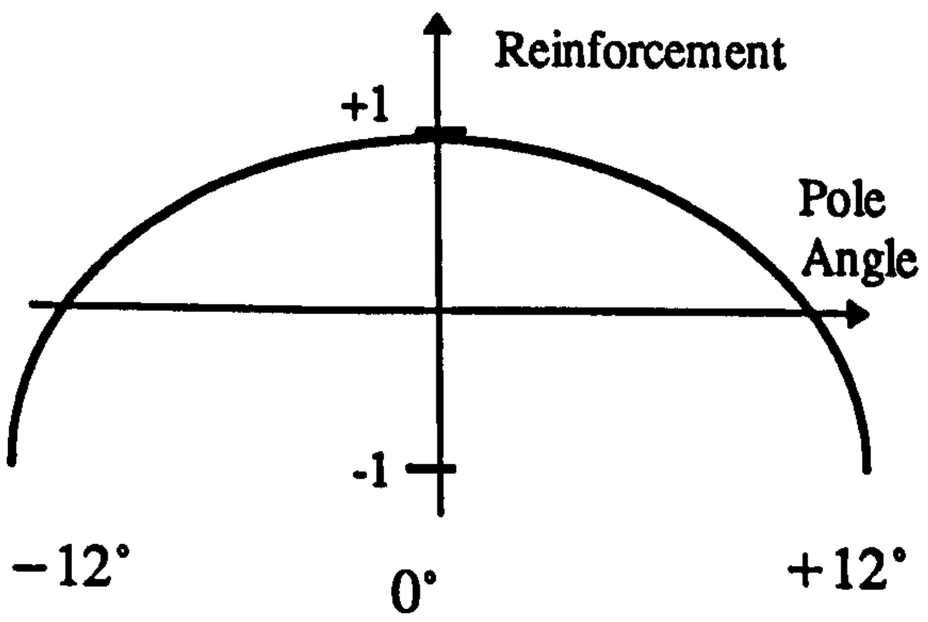


Figure 3.7 A possible evaluation function for reinforcement given the pole angle. At the two extremes of pole angle, the reinforcement is -1. At the balance point, reinforcement is +1. No linear decision boundary exists (a single point) between positive and negative reinforcement.

3.6.3 The Adaptive Critic Element (ACE)

The ACE is similar in structure to the ASE (see Figure 3.6) and computes an expected or predicted reinforcement signal given the current state vector and external reinforcement from the system; the predicted reinforcement is continuous unlike the external reinforcement signal and allows learning throughout a trial. Thus, the combined ASE / ACE system is not a purely failure driven system. The prediction of expected reinforcement is given by

$$p(t) = \sum_{i}^{n} q_i(t) x_i(t) \qquad (3.10)$$

where $q_i(t)$ is the weight for the i^{th} input line and $x_i(t)$ is the input signal for that line as before.

The learning rule is given by

$$q_i(t+1) = q_i(t) + b\hat{r}(t)\bar{x}_i(t) \qquad (3.11)$$

where b, $b > 0$ is a constant which determines the rate of change of learning in q_i, $\hat{r}(t)$ is the predicted reinforcement and $\bar{x}_i(t)$ is a trace of the activity of the input variable x_i.

This trace, unlike the eligibility trace, does not take into account the control action chosen by the system for the region of state-space. It is given by:

$$\bar{x}_i(t+1) = \lambda\bar{x}_i(t) + (1-\lambda)x_i(t) \qquad (3.12)$$

where λ, $0 \leq \lambda < 1$, is a rate of change constant. Although similar in form to the eligibility trace, it provides a record of the activity of the input line x_i alone during the trial to determine whether or not the particular input line contributes to the prediction. With the present protocol of selecting a single input line, equation (3.10) becomes $p(\tau) = q_k(\tau)$ at time τ where the weight q_i reflects the prediction of failure for a given control action elicited by entering the region of state-space coded for by input line k.

A distributed version of equation (3.10) might also be used where multiple input lines, $x_i(t)$, are activated to varying degrees, in the range zero to one, and thus

weight the prediction contributions to give a final prediction of reinforcement; this possibility is mentioned in Barto *et al* (1983). A novel distributed architecture, FUZBOX, is discussed in section 5.5.

The predicted reinforcement is given by

$$\hat{r}(t) = r(t) + \gamma p(t) - p(t-1) \qquad (3.13)$$

where r(t) is the external reinforcement, $r(t) \in \{0,-1\}$, and γ, $0 < \gamma \leq 1$, is a discounting factor. The discounting factor is required to prevent the reinforcement from becoming self-sustaining. To see this, consider $\gamma = 1$ and $p(t) = p(t-1)$ at some time, t. If failure has not yet occurred, equation (3.13) gives $\hat{r}(t) = 0 + p(t) - p(t-1) = 0$

Now, from equation (3.11), $q_i(t+1) = q_i(t) + b\hat{r}(t)\bar{x}_i(t) = q_i(t)$ $:\hat{r}(t) = 0$ for some node, i, so that $p(t+1) = q_i(t+1) = q_i(t) = p(t)$ if node i is chosen again. Thus, the prediction for a particular node becomes self-sustaining.

When r(t)=0, (failure has not yet occurred) a smaller prediction of failure, p(t)>p(t-1), (e.g. -0.8 > -0.9) signifying a transition from a region of higher expected failure to a region of lower expected failure, constitutes a positive reinforcement.

When r(t)=-1 (failure), p(t)=0 (no present prediction) and equation (3.13) becomes $\hat{r}(\tau) = -1 - p(\tau - 1)$. Thus, the degree of prediction of failure is taken into account and fully predicted failure is not penalised.

For the reinforcement learning system just described, the weights can be viewed as representing probabilities (although not normalised) stored in stochastic automata which determine the next action given a particular state.

This is an approach to reinforcement learning in a specific way. For a more standard introduction see Barto, Bradtke and Singh (1995).

3.7 Simulations

3.7.1 Replication

The BSA system was implemented as detailed in Barto, Sutton and Anderson (1983) for comparison purposes. A series of runs was carried out. Each run consisted of a sequence of trials, the cart-pole state was reset to $x = \dot{x} = \theta = \dot{\theta} = 0$ at the beginning of each trial. The ASE / ACE parameters were set as follows: a=1,000, b=0.5, δ=0.9, γ=0.95, λ=0.8. The cart-pole simulation details are given in appendix F.

A summary of the results of 100 runs is given in Table 3.1. A mean trial count of 106 trials required for convergence concurs with the results of Barto, Sutton and Anderson (1983) in which 10 runs were carried out up to a maximum of 100 trials. At 100 trials the BSA results show an average balance time of approximately. 1600 seconds (80,000 time steps) indicating that the system had learned to balance the pole.

mean	min	max	SD
106.09	33	917	133.073

Table 3.1 Mean results for 100 runs for the replication studies of the original BSA system.

The min trials and max trials figures indicate the minimum number of trials to convergence and maximum number of trials to convergence respectively during the set of 100 runs. Note the large variation between a run which converges within 33 trials and one which took 917 trials to converge.

The standard deviation figure of approximately 133 shows a large variance and indicates that the convergence rates are not grouped tightly; The variability of convergence is confirmed further in Table 3.2 which shows the first 10 runs of the replication simulation.

seed	1	2	3	4	5	6	7	8	9	10
trials	71	130	50	53	50	121	141	83	57	82

Table 3.2 The convergence times for the first ten runs of the replication studies.

To get the figures for 100 runs, one anomalous run was removed because it failed to converge within 10,000 trials and it appeared that the system could not recover from "bad" strategy choices.

Individual runs were qualitatively similar to those of the original BSA implementation; Figure 3.8 shows the characteristic slow start, with many early failures, followed by a rapid rise in performance.

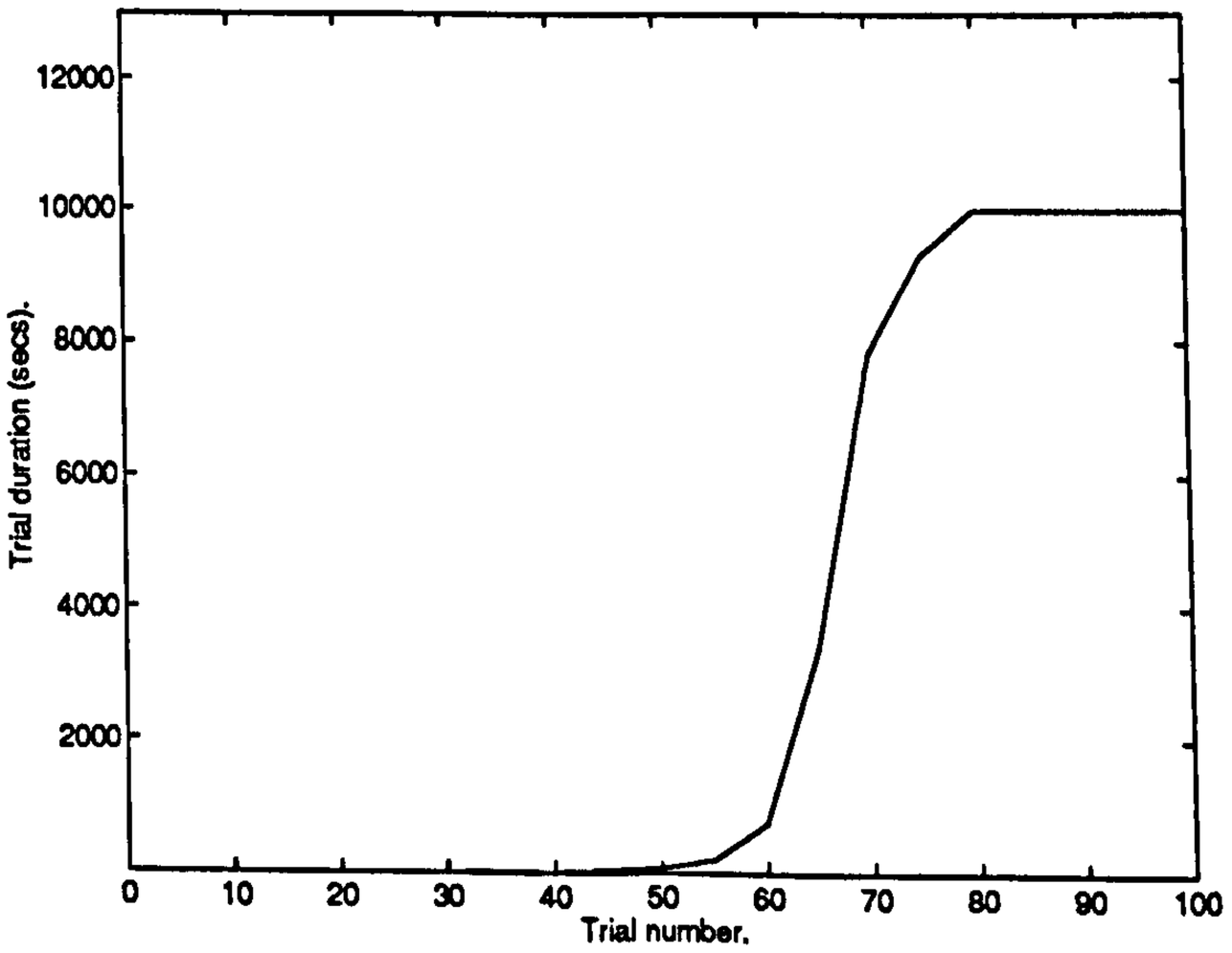


Figure 3.8 A typical run showing ASE/ ACE reinforcement learning performance over 100 trials.

Figure 3.9 displays the same results without averaging across bins of five (see Barto, Sutton and Anderson, 1983). In this raw form, it is readily apparent that learning is not monotonic with trial durations dropping down to lower levels as time proceeds.

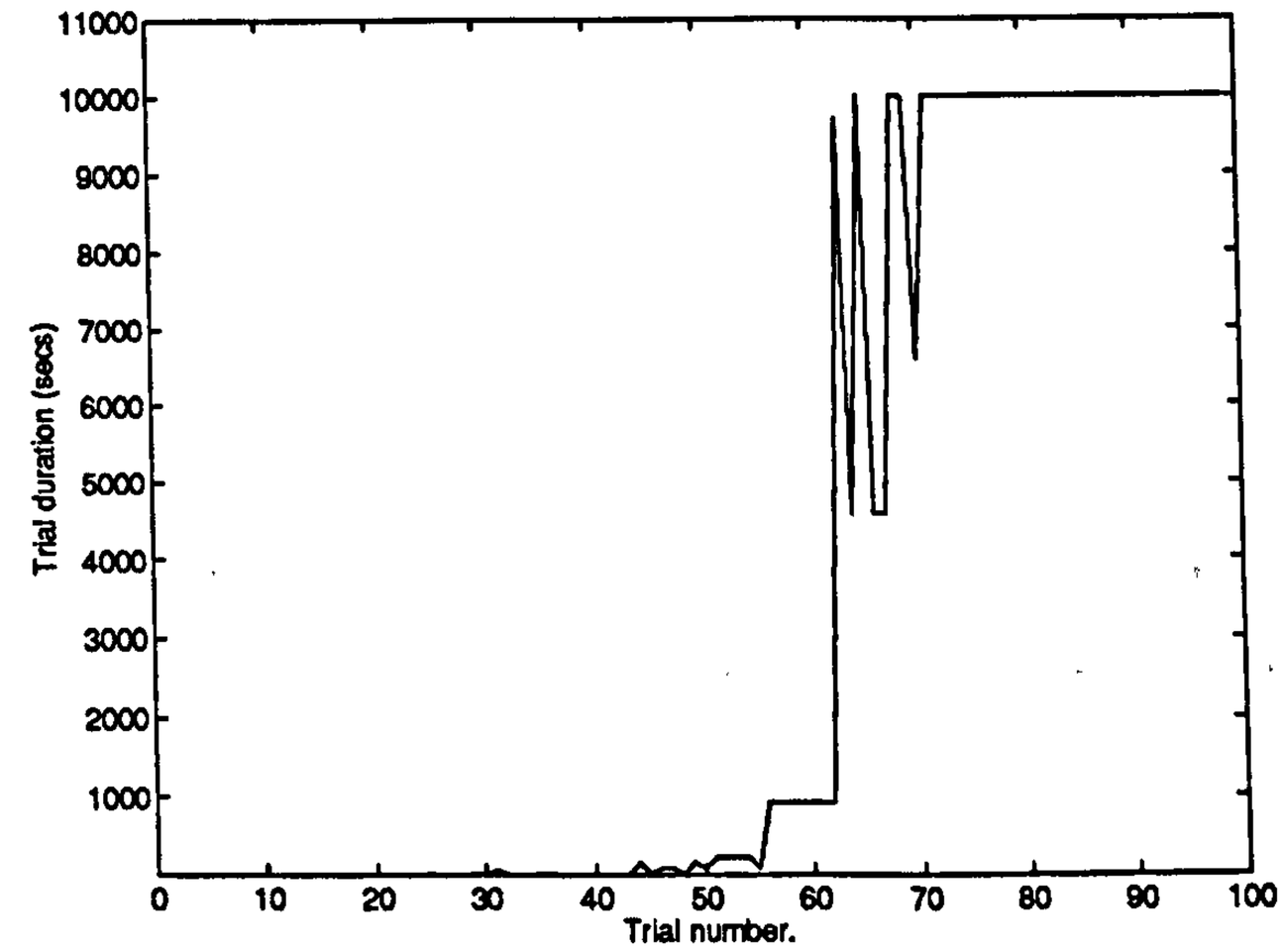


Figure 3.9 The run of Figure 3.8 without averaging to show the non-monotonic nature of learning.

Figure 3.10 shows the incremental usage of boxes during learning. The monotonic increase in the number of boxes recruited continues until a sufficient coverage of state-space is achieved. For this run, a total of 129 boxes, out of the maximum of 162, was used.

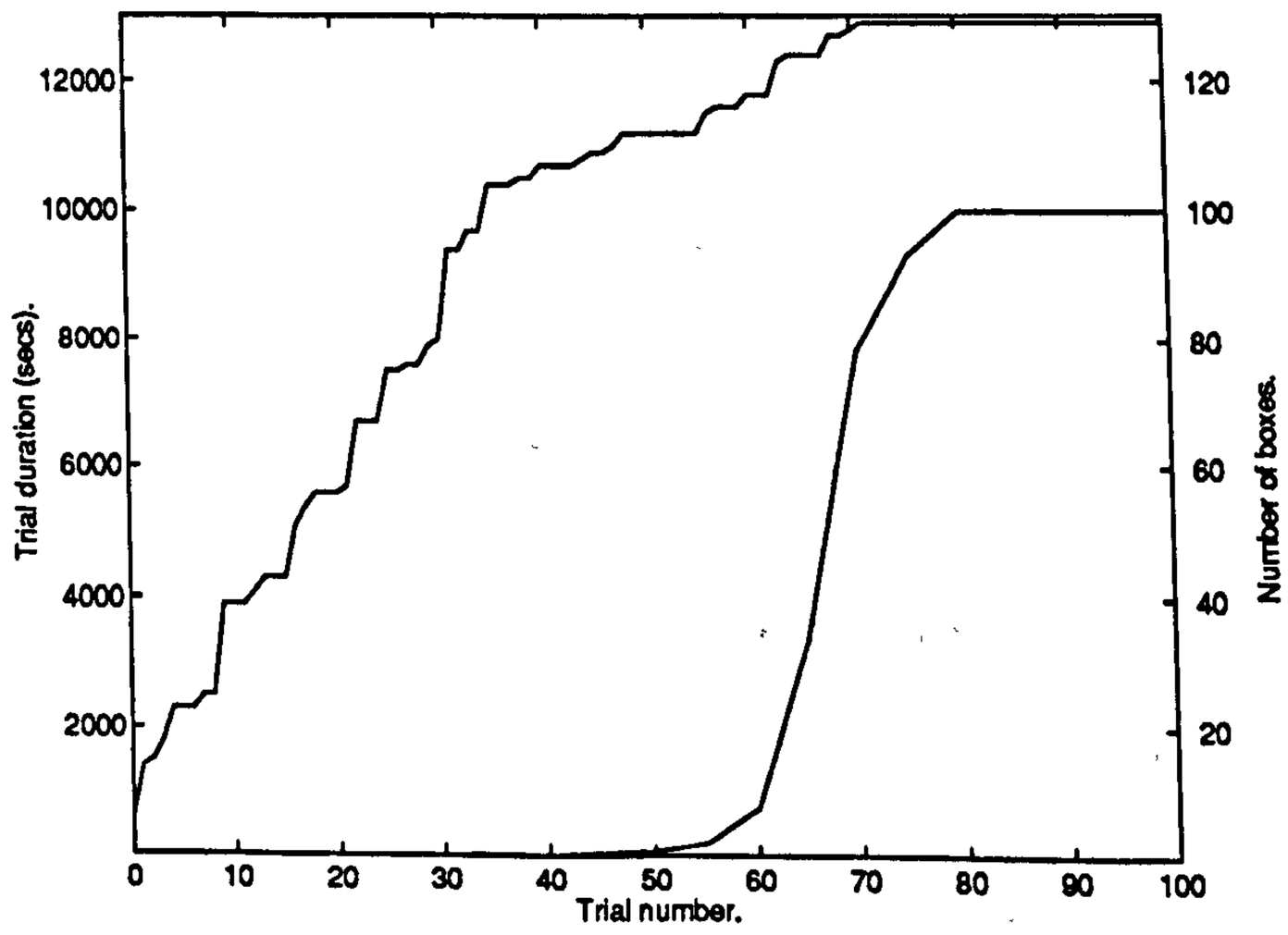
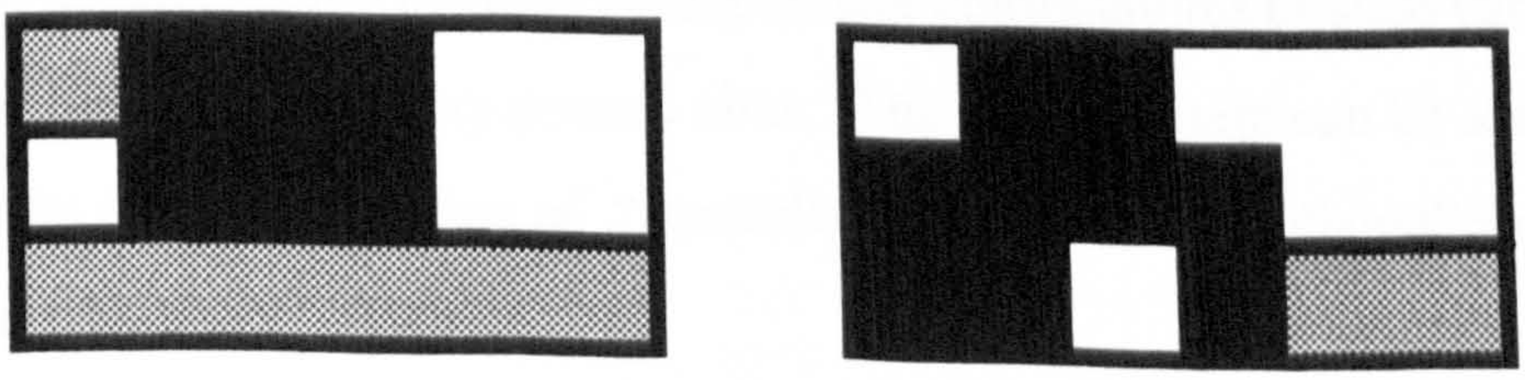


Figure 3.10 The incremental use of boxes with learning for the ASE /ACE system for the run of Figure 3.8

3.7.2 Box Usage.

The run details give useful information about the performance of the BSA implementation of a reinforcement learning system However, this is a black box approach and does not give any indication of how the control strategy is represented across state space for any particular run. At this point it is instructive to "open the box" and look at a particular control strategy implementation.

The BSA system can be envisaged as a crude rule-base which specifies a mapping between states and actions. Figure 3.11 shows a trained system in graphical form. Only a subset of the state-space regions are shown for illustrative purposes. The information is stored as a set of 162 "rules" with four antecedent propositions—one for each of the state variables—and a consequent action specifying a positive or negative force. To represent the five dimensional information in two dimensions, the cart position and cart acceleration are used as parameters to specify one of nine quantised phase planes involving the angle and angular acceleration. The box shadings indicate that a box has not been entered at any time or, if it has been entered, the direction of the force specified by the control policy. The phase plane shown in Figure 3.11 consists of 3X6 or 18 boxes arising from the fact that the pole angle is quantised into six regions and the angular velocity into three regions. The cart distance and cart velocity have been fixed giving one of nine possible phase planes.



(a) (b)

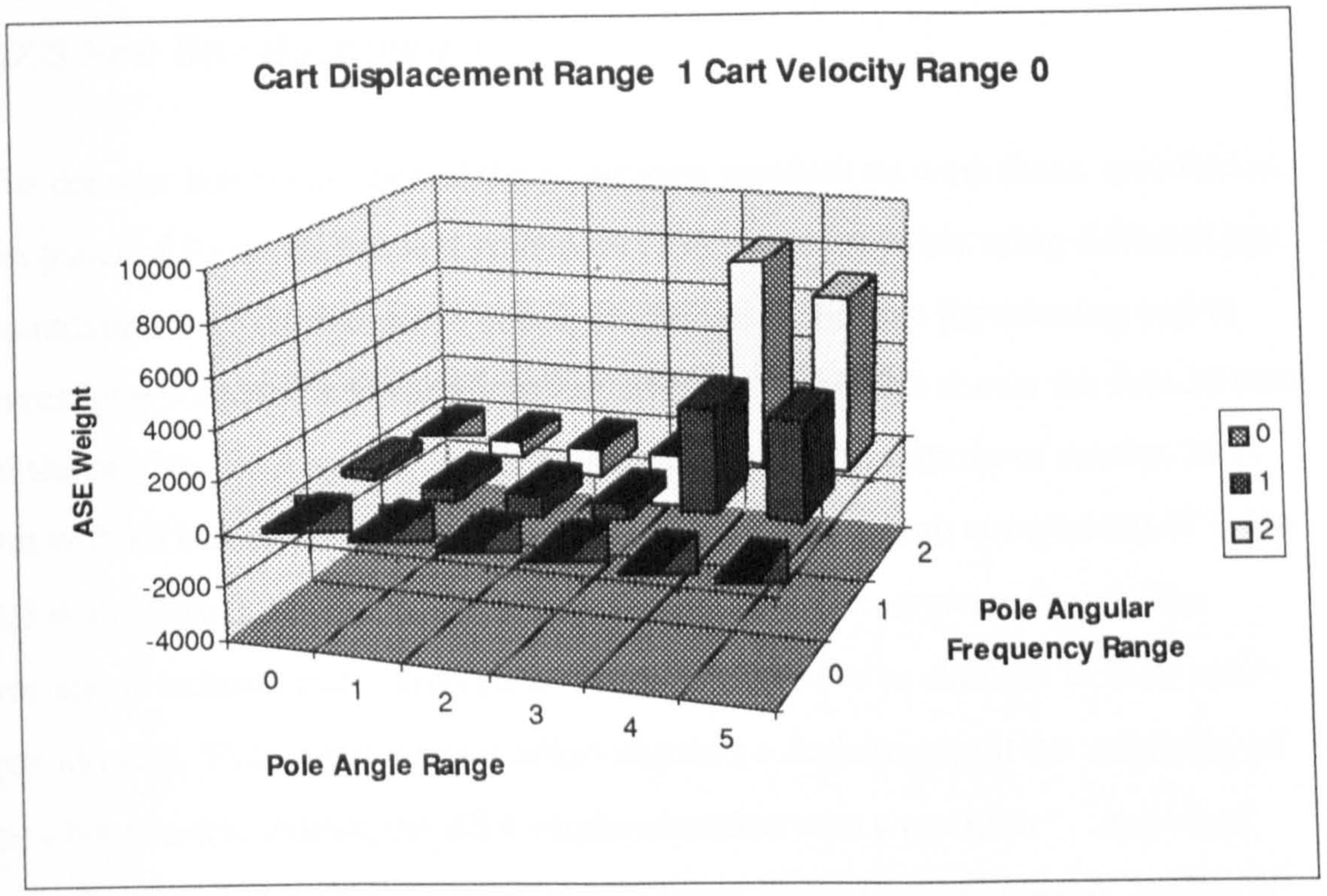
Figure 3.11 Two quantised phase planes showing control rules after training. Black and white regions indicate left and right control forces respectively. Grey regions indicate regions of state-space not yet explored for this run.

Figure 3.12 shows the magnitudes of the weights for the state space region (a) illustrated in figure 3.11; the increased detail gives a better idea of what is happening. Areas of state space which have not been entered may have impossible combinations of state variables.

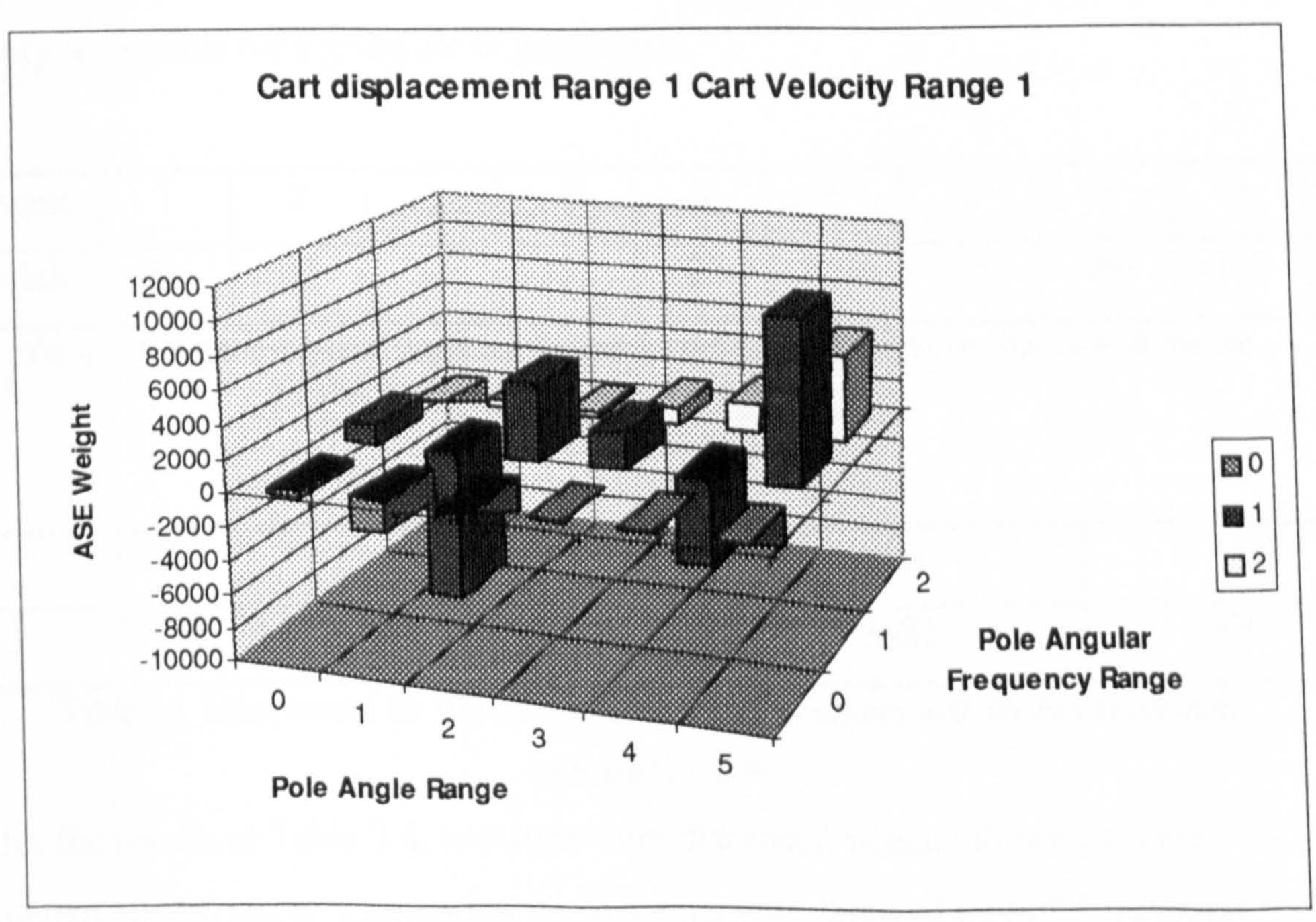
Michie and Chambers (1968b) distinguish between "informed" or "uninformed" and "decisive" or "indecisive" boxes. These labels are given on the basis of the amount of information accumulated and the strength of the left or right decision. They noted that the cart-pole problem exhibited considerable symmetry; this fact was used to extract information about the nature of learning with the 'boxes' system (Michie and Chambers, 1968a).

For the initial random configuration, there was 50% symmetry when it came to left / right decision making. The final configuration for a single run exhibited 84% symmetry for informed boxes and 50% symmetry for the uninformed boxes. Thus, learning allows the boxes system to order information and build a structured representation of state space. Note that the internal representation in this case is transparent, that is, it is directly accessible by an observer. There is no need for a, possibly complicated, mapping between weight space and state space.

Informed and decisive regions of state space are where a left / right decision is essential for the maintenance of an adequate control strategy. Regions which are informed and yet indecisive are "don't care" regions which are not so important. Uninformed regions signify difficult or impossible combinations of state variables which often entail contradictory control aims. The boxes system can be seen as building up its own classification of "controllable" and "uncontrollable" states (Michie and Chambers, 1968b).



(a) ASE weight values for the box usage diagram of Figure 3.11. Note the variation in weight magnitudes between boxes making the same direction "decision".



(b) Box usage for a phase plane adjacent to that of (a). this phase plane results from a shift of only one cart velocity range.

Figure 3.12 Box usage graphs extending the information given in Figure 3.11. The codes "0,1,..." form an index of the state-space regions used by the boxes system e.g. 00 denotes the first box of the phase-plane and 52 denotes the eighteenth.

3.7.3 New Box Boundaries

The decoder box boundaries of the replication simulations were those specified in the paper of Barto, Sutton and Anderson (1983). Simulations using different box boundaries were carried out to anticipate the use of a fuzzy partitioning and to investigate robustness to variations in partitions. Table 3.3 shows the first 10 runs of the original BSA system using the parameters and conditions of section 3.7.1 but with all box dimensions increased by 10%. A run by run comparison of Table 3.3 with Table 3.2 reveals significant differences in convergence times. The variations indicate that convergence times are sensitive to changes in state space partitioning. Fixing partitions *a priori* requires a decision about the suitability of box boundaries; indeed, the BSA implementation uses a partition "...based on specific knowledge of the control task" (Barto, Sutton and Anderson, 1983). If problem specific knowledge is not available, much experimental work may be required to optimise the state space partition and, even then, the partition may only be suitable for a given set of parameters.

seed	1	2	3	4	5	6	7	8	9	10
trials	426	47	50	125	85	2794	56	269	112	61

Table 3.3 The convergence times for the first 10 runs of the replication studies with the box boundaries increased by 10%

mean	min	max.	SD
295.4	21	3881	670.736

Table 3.4 Mean results for 100 runs for the replication studies with the box boundaries increased by 10%

for the results of Table 3.4, two runs were discarded which did not converge within 10,000 trials. Comparing these results with those of table 3.1 indicates that performance is affected by even a small change in the box boundaries. The mean convergence "time" has almost tripled whilst the variance has increased considerably.

To anticipate the simulations of section 5.5.6 using a fuzzy partitioning, the state space was set up using the following unoptimised partitioning:

i) x: $-2.4m \le x < -1.6m \le x < -0.7m \le x < +0.7m \le x < +1.6m \le x < +2.4m$,

ii) θ: $-12° \le \theta < -6° \le \theta < -1° \le \theta < +1 \le \theta < +6° \le \theta \le +12°$,

iii) $\dot{x}$: $\dot{x} \le -2.0m/s < \dot{x} \le -0.5m/s < \dot{x} \le +0.5m/s < \dot{x} \le +2.0m/s < \dot{x}$

iv) $\dot{\theta}$: $\dot{\theta} < -50°/s \le \dot{\theta} < -10°/s \le \dot{\theta} < +10°/s \le \dot{\theta} < +50°/s \le \dot{\theta}$

This partitioning is what would result if a fuzzy partitioning was used in winner-takes-all mode, that is, if the partition boundary between two fuzzy boxes was taken where the fuzzy membership functions crossed. The fuzzy boxes system, FUZBOX, is covered in section 5.5.

Table 3.5 shows the first ten result obtained using the same conditions of the previous two simulations with the new partitioning. The difference in convergence time for the same random number seed is large in some cases; run 6 failed to converge within 10,000 trials.

seed	1	2	3	4	5	6	7	8	9	10
trials	55	276	374	3707	70	-	1239	53	51	178

Table 35 The convergence times for the first 10 runs of the replication studies using the new 625 box partitioning

Using a fuzzy partitioning, which has the discrete partition described above as its limit, the results are radically different as shown in section 5.5.6. The difference in convergence times observed when using a form of distributed representation indicates that distribution of information across neighbouring boxes may be a useful characteristic to confer upon an autonomous learning system because learning is accelerated.

Distributed representation systems are discussed in section 5.5 where learned information is used to inform neighbouring boxes and control information is obtained from more than one box and combined to give a resultant. The remainder of chapter 3 will continue to look at non-distributed systems and their properties; this is commensurate with the original BSA reinforcement learning implementation.

3.7.4 Discussion

The ASE / ACE implementation of reinforcement learning is very effective as can be seen from the results and it is difficult to see how the actual learning mechanisms should be modified to improve upon it. However, the decoder is functionally isolated from the ASE / ACE modules and provides a focus for modification.

The original decoder (Barto, Sutton, and Anderson, 1983) is preset by the user according to empirically derived principles. This is not entirely satisfactory for an autonomous system which should be able to develop its own quantisation of state-space through experience. This is the subject of Chapter 4 which introduces and evaluates a novel self-organising decoder, EUCART, (Marriott and Harrison, 1995, 1996).

One noticeable characteristic of the BSA reinforcement learning system is that learning is not monotonic. Learning consists of exploration of the state-space using a stochastic search technique; this is exploration. During exploitation of control strategies, the state-space trajectory may drift into neighbouring regions of state-space which have not yet been explored; this may happen because random perturbations force the trajectory out of control regions which are only weakly established. The exploration-exploitation gives rise to characteristic "plateau and drop" behaviour where successful control appears to be established and lost.

Chapter 4 EUCART and the EUCART-BSA Hybrid

4.1 Background

This section provides an overview of the problem and a critique of some current approaches. The development of a novel architecture is, thus motivated.

4.1.1 The Decoder Subsystem: Pre-processing

In the original BSA implementation, the decoder is specified by using a fixed mapping between the partitioned state-space and the input lines. Another decoder scheme (Lin and Kim, 1991) uses the cerebellar model articulation controller (CMAC) of Albus (Albus, 1975a, 1975b, 1979; Tolle and Ersu, 1992) with a fixed number of memory locations and an efficient mapping which maps only states which are used, to locations in the CMAC controller. The distribution of state-space information across the locations leads to a degree of overlap and, consequently, some ability to generalise about regions of state-space not yet traversed. The large state-space is mapped to smaller storage space using the state variables as an address key (Lin and Kim, 1991) for the decoder. This compression avoids allocation of storage for large regions of state-space which are not used.

The decoder provides a sub-unit replete with possibilities for modification. Decoder modules can be designed which implement various mappings between state-space and the ASE / ACE controller sub-systems. If the decoder co-domain consists of independent input lines as in the original BSA implementation, then the possibility of increasing network size by exploring state-space presents itself. The addition of new input lines, representing newly traversed areas of state-space, will not conflict with the previously established input lines to the ASE / ACE and their

corresponding weight and trace values. Although the input lines are independent, the state-space regions represented by these lines may overlap and temporarily disrupt the mapping; this phenomenon is considered in section 4.2.

4.1.2 An ART-based decoder

From equations (3.7) and (3.10), for some k,

$$x_i(t) = \begin{cases} 1 & for \quad i = k \\ 0 & for \quad \forall i \neq k, \end{cases}$$

means that y(t) and p(t) depend upon one input line only. This decoupling of the x_i allows the addition of new input lines without disruption of the established output and prediction values, for the existing lines, which would occur if more than one input line contributed to the calculation. Thus, decoders which dynamically partition the state-space, using whatever method, can be easily linked to the ASE / ACE sub-systems provided that the coded state-space regions have unique input lines. This method is highly dependent on experience and is flexible in that new regions of state-space encountered under different initial conditions or disturbances can be accounted for by allocating new storage areas (nodes) which contain the traces and expected lifetime / prediction values for the newly encountered state-space region.

A distributed representation of *y(t)* and *p(t)* of equations (3.7) and (3.10) respectively is possible if input line conflicts are avoided by allocating input line activity according to the degree of node membership (e.g. Zhang and Grant, 1992). Here, the single activated input line convention of Barto *et al*, (1983) is adopted for compatibility between the original ASE / ACE formulation and winner-takes-all dynamics.

4.1.3 Other Approaches Using ASE / ACE type modules

A multilayer non-linear network, with sub-systems operationally similar to the ASE / ACE subsystems, was developed by Anderson, C. W. (1989) to address some of the shortcomings of neurocontroller architectures.

Anderson acknowledged the problems of control such as unavailability of dynamical information and the credit assignment problem, but also stressed the non-linearity of the evaluation function (see section 3.6.2). This means that, although it is possible to use a linear control force function, there is no way to train a linear controller directly from experience; a previously developed control law is required to train the neurocontroller which defeats the object of using a neural network and precludes autonomous operation. The situation is much worse when the plant dynamics are not known and a controller cannot be designed to provide examples of desired neurocontroller behaviour.

The BSA system gets around this problem by using pre-processing to decouple the system states to provide a look-up table of neurocontroller actions; this approach requires the *a priori* use of pre-processing by the user which, again, reduces neurocontroller autonomy.

Anderson (1989) proposes the use of a non-linear neurocontroller to allow pre-processing to be included in the control process itself. The ASE / ACE counterparts of Anderson's system are known as the *action network* and *evaluation network* respectively.

The action network is non-linear and is capable of learning a control force function of the form: $F_t = b_1\theta_t + b_2\dot{\theta}_t + b_3 h_t + b_4 \dot{h}_h$ using Anderson's notation for the state variables. An alternative method of generating a linear controller, using genetic algorithms (Howell, 1994), is discussed in section 4.1.5.

The evaluation network is also non-linear and is capable of learning the non-linear evaluation function.

Training is carried out using a variant of the backpropagation algorithm which allows the hidden units to learn by circumventing the credit-assignment problem. The errors backpropagated to the hidden units of the evaluation function were derived from the evaluation network's output whereas the errors backpropagated to the action network hidden units combines that error with action information. Anderson first shows that using a single layer network gives poor results, even though the action net can learn a strategy, because the evaluation function is non-linear and so the action network doesn't "know how to learn" such a strategy.

Results for the two layer network—using the cart-pole simulation—show that a much longer learning time is required when compared to the original BSA system. The increase in learning time is accounted for by the advantage of increased generalisation across state-space. This illustrates the trade-off between generality and learning speed encountered in many control problems. The richer experience of the Anderson network makes it more robust in that the network represents a non-linear function as opposed to a piecewise look-up table.

For Anderson's system, approximately 10,000 trials were required to balance the pole for about 7,000 steps (140 seconds). An *a priori* choice of boxes is not required but at a cost to performance.

Lin and Kim (1991) use the CMAC network (Albus, 1975a, 1975b, 1979; Tolle and Ersu, 1992) to form the state-space decoder for an ASE/ACE unit-based learning system. The CMAC network distributes individual state boxes and their corresponding ASE/ACE weight and trace values across CMAC storage locations. Instead of assigning state and related information to a single location, it is shared between overlapping locations such that, for a given input repeated immediately after learning, the information will be accurately reproduced. Repeated learning experiences of this type lead eventually to a distributed representation of the control surface.

The CMAC reinforcement learning system was tested using the standard cart-pole simulation in one of two modes: for mode one, the cart-pole system was reset to zero initial conditions after each failure, for mode two the state was reset to a random value. The results of simulations using these two modes of operation were compared with those of the original BSA study (Barto, Sutton and Anderson, 1983) and Anderson's multilayer neural network system (Anderson, 1989).

The CMAC based system was found to consistently outperform both of the systems used for comparison (Lin and Kim, 1991) and illustrated the effect of varying the CMAC storage capacity. The memory storage requirements (in terms of locations) could be reduced below those of the original BSA study because there are only a few critical states concentrated within a small region. The advantages of using a CMAC decoder are a reduction in storage requirements and an increase in learning rate through generalisation of state information. No memory capability is wasted on uninformative states. Interpolation of information across locations reduces both the storage overheads and leads to the observed increased learning rate.

Note that distribution of information throughout the network in this case involves individual quantised states being "spread" across the set of storage locations; it does not mean that actions and predicted outputs are composites produced by combining values associated with several states; individual weight and trace values are used by the learning system following retrieval from the CMAC memory.

Santiago and Werbos (1994) use a method related to reinforcement learning and known as dual heuristic programming (DHP) to solve the cart-pole problem. The DHP network consists of four components:

- an *action network*, which issues the control actions;
- a *critic network*, which evaluates the utility of performing given actions;

- a *model network*, which performs one-step-ahead prediction of system states, and
- a *utility function* which is used to modify the critic network.

The success criterion for the DHP network was defined as being able to balance the pole for 30 minutes (1800 seconds). According to this criterion, the average balance time over 11 runs was 31.8 trials to success. The 11 runs used 4 different pole lengths. The results were compared against a set generated using backpropagation through time and the DHP network was found consistently to outperform the comparison system.

4.1.4 Other Approaches Using ASE / ACE: A Critique

Anderson's non-linear system appears to confer robustness of learning at the expense of training speed; robustness is desirable in control applications but at what cost? The use of a feedforward network trained with a modified backpropagation method (gradient descent) reduces network flexibility.

The network size and configuration has to remain fixed once specified; no nodes can be added or removed during operation to adjust the network according to experience in state-space. This may lead to sub-optimal solutions in that a control or evaluation surface may be under- or over-represented.

The network is also opaque in the sense that the distributed stored representation does not easily yield information to a user or expert. Extraction of explicit operational information from feedforward neural networks, such as the multilayer Perceptron, is not easy (Ma, Harrison and Kennedy, 1995) and requires specialised construction to facilitate rule extraction (Brown and Harris, 1994).

The CMAC interpolation method is not of the first type postulated by Barto, Sutton and Anderson which would involve "overlapping sets of output pathways"

(Barto, Sutton, and Anderson, 1983). It is of the second type involving associative memory networks "...in which dispersed rather than localised patterns of activity encode information." (Barto, Sutton, and Anderson, 1983).

The distribution method of the second type, although conferring advantages upon the learning system—like the CMAC based system—does not facilitate ease of information retrieval; the system still remains opaque to a user or expert. The original boxes (Michie and Chambers, 1968a) and BSA (Barto, Sutton and Anderson, 1983) approaches formed a crude rule-base from which dynamical information could be obtained easily. For the CMAC based system, the output would have to be reconstructed for a given input and the rules generated using a black box (based only on input / output behaviour) approach (Ma, Harrison and Kennedy, 1995); this would lead to a combinatorial explosion of trying different combinations of inputs to generate the rule base even though there are a limited number of storage locations. The main reason is that the storage locations do not have a direct one-to-one relationship with a rule-base—the candidate rules (boxes) being distributed throughout the CMAC memory.

The EUCART-based reinforcement learning system has some of the characteristics of transparency with nodes directly representing sets of related states (closed balls) with associated weight and trace information. These state sets can be seen as "micro-rules" giving dynamical information about small regions of state-space. The use of localised as opposed to distributed information eases the problem of information extraction and, as explored in this thesis, makes structural alteration of the neurocontroller feasible through pruning and addition of nodes.

The EUCART rules, although transparent, are not sufficiently general; an investigation into the feasibility of "lumping and splitting" (Michie and Chambers, 1968b) is required. Such a scheme of rule generalisation or specialisation is possible because of the non-distributed nature of the decoder module. The lack of distribution of the second type, however, does not preclude the possibility of

distribution of the first type which is compatible with both generalisation and transparent information representation; the latter forming a one-to-one relationship with a rule-base.

The fuzzy version of the BSA boxes system, named FUZBOX here, combines the advantage of a distributed representation (of the first type) with the utility of a transparent control mapping in the form of fuzzy rules; the rules encapsulate easily interpreted dynamical information and allow combinations of rule information across states to facilitate learning and control action. The FUZBOX system is discussed in detail in chapter 5 where it is compared with other systems including the EUCART-based system.

4.1.5 Alternative Approaches

As discussed in section 3.3.4, the cart-pole system can be linearised so that standard state-space methods can be applied. The fact that a linear controller exists means that a linear neural network such as the adaline can be used as a neurocontroller. The adaline provides an early example of neurocontrol (Widrow and Smith, 1963; Widrow, 1987) using operator modelling where an existing controller or human operator is used to provide the training data. Supervised learning is required when linear neural networks are used because the control evaluation function is non-linear. A visually supervised version of the linear adaline controller has been developed (Tolat and Widrow, 1988)

The linear cart-pole model is controllable using proportional plus derivative control (Picton, 1994) where the control force is proportional to the error and the derivative of the error between the actual and desired outputs. The output is a linear function of the cart position and the pole angle and their derivatives. Bang-bang control can be used; it is time optimal as shown by Pontryagin's Maximum

Principle (Hocking, 1991). A suitable control system consists of proportional plus derivative control and a hard limiter (Picton, 1994).

Defining a parameter vector, $\mathbf{w} = [w_1 \quad w_2 \quad w_3 \quad w_4]'$ and a state vector $\mathbf{x} = [x_1 \quad x_2 \quad x_3 \quad x_4]' = [x \quad \dot{x} \quad \theta \quad \dot{\theta}]'$ the form of a bang-bang control solution can be stated as $u = \text{sgn}(\mathbf{w}'\mathbf{x})$ which can be solved using an adaline with a hard limiter on the output (Widrow and Smith, 1963; Widrow, 1987). The adaline is more suitable than the Perceptron (Rosenblatt, 1962) owing to the requirements of the Perceptron Convergence Theorem. If the separating boundary of the two classes representing the discrete outputs is not defined precisely by the data (rendering the problem not linearly separable) , then the Perceptron will not converge. The adaline will cope well with noisy data and give the best linear approximation. The adaline solution defines a switching surface and generalises after being trained using a borderline exemplar set.

The adaline controller was trained, during a training phase, using a teaching controller (Widrow and Smith, 1963; Widrow, 1987). The linearised differential equations representing the cart-pole system were given by $\ddot{\theta} = \frac{3g}{4l}\theta - \frac{3}{4lM}F$ and $\ddot{x} = \frac{1}{M}F$ where assumptions were made that damping was negligible, and that the pole has no effect upon the cart motion. The teaching controller was of the form

$u = -2.0\dot{\theta} - 1.0\theta + 1.0\dot{x} + 1.0x$ The state variables were individually coded using a 6 bit binary code giving a 24 bit binary input vector. Although successful, such linear systems are limited and have very limited autonomy; they are restricted to linear system models and have to be trained using an operator or existing controller.

Criticisms of inflexibility and possible sub-optimal learning by Anderson's non-linear reinforcement learning system (Anderson, 1989), are addressed with a new architecture based upon Q-learning. The Q-Learning system with hidden unit

restart (Anderson, 1993) still uses a fixed structure net but allows the reuse of existing nodes. The goal of supervised learning is to obtain a compact representation with good generalisation. The Q-learning network uses an incremental gradient based search (similar to backpropagation) and gives initial fast learning with localised experience; the network is not allowed to generalise too widely in the beginning. There are a number of fixed units which are trained at every step. During the gradient based search, units are not added or removed; changes to network topology are made by restarting the least useful unit.

An example of plant modelling (forward modelling) applied to the cart-pole problem is the temporal difference approach of Jordan and Jacobs (1990). They attempt to model the system by using the error between the actual and predicted plant outputs to drive a backpropagation algorithm. The idea is to find an adequate model and use this model to train a neurocontroller. The simulation protocol is similar to that of Barto, Sutton and Anderson (1983) but with three important differences:

- disturbances (white noise) were derived from the environment and not from the controller
- the forces applied were real valued, not binary, and
- after failure, the cart-pole simulation was set to a random value, not to the origin.

The temporal difference algorithm of Sutton (1988) was used to learn the system model; this is a prediction problem. A variant was developed in which the learning of the forward model and the controller proceeded simultaneously.

The results of 20 runs were stated. Of the 20 runs, 18 found an adequate control configuration and 2 fell into local minima. A set of six runs was illustrated and had a minimum and maximum run length of approximately 250 and 1400 trials respectively.

The learning times were longer than those for the original BSA study. However, learning speed is not necessarily the best guide to performance. The fact that a random starting position was used following failure is likely to mean that the controller is more robust to changes in starting conditions.

The approach of Connell and Utgoff (1987) attempts to build up a "map" of the environment through experience. For the cart-pole problem, training information is weak. Physical system and control constraints makes inevitable regions of state-space where the recovery of control is not possible; these "doomed" regions of state-space must be avoided at all costs. The goals of an autonomous learning system are to identify and to avoid these undesirable states. The fundamental idea is to build up a potential map of state-space including the "hot spots".

The CART system of Connell and Utgoff (1987) consists of four elements which work together to construct the potential map:

- *problem generator:* which initialises the cart pole system for a new trial; the cart-pole system is reset to a small random perturbation away from the equilibrium point where the cart is centred on the track and the pole is vertical.

- *performance element:* which chooses a control action—a left or right push—at each time step; the choice is either to repeat the last action or to carry out the opposite action. The decision is based upon the angle between two vectors, the *gradient* and *extended* vectors. The gradient vector—indicating the direction of the desirable state-space trajectory—is computed first followed by the extended vector which shows the direction of the state-space trajectory if the last action is repeated. The objective is to move to a more desirable state by following the gradient "downhill". If the angle between the two vectors indicates that the current action policy is reducing the undesirability of being in this region of state-space then continue, else change the action;

- *learning element*: which estimates the desirability of states given the 5-dimensional training instances of the state vector coupled with a +1 or -1 label which constitutes the reinforcement. The learning element interpolates between the training instances to build up a surface in 5-dimensional space;

- *critic:* which supplies information to the learning element in the form of the state vector and label pair. The process is initiated using a short-cut which sets all state variables to zero and labels this point as desirable. When the pole falls, the final state is labelled as an undesirable state. Between these two extremes, the algorithm runs until the pole is balanced for greater than 100 time steps. The algorithm is to back-up to the state which occurred 50 time steps prior to failure and keep on backing-up until a state is found from which at least 3 of the state variables decrease in magnitude; the resulting point is then labelled as desirable.

The numeric parameters of 100, 50 and 3 featured in the critic are empirically derived. An automatic method of deriving such parameters would be desirable (Connell and Utgoff, 1987).

An approach, based upon drive-reinforcement theory and related to temporal difference learning has been developed by Morgan, Patterson and Klopf (1990). Temporal differences of predicted reinforcement are compared and used to control learning at the single neuron level. A network consisting of two neurons, one for each force direction, has a series of inputs or "drives" which represent a prediction of eventual reinforcement. A change in a drive level represents a change in predicted reinforcement.

The main difference between this approach and the BSA approach is that only those drives which have recently changed are reinforced thus rendering fewer drives eligible for reinforcement. It was claimed that for one particular run, only a single trial was required to learn a successful control strategy without failure. For this run, parameters controlling the learning rate were set *a priori*. For other

values of the learning rate parameters failures occurred. The state-space partitioning is fixed *a priori* for the drive-reinforcement system in a similar manner to the boxes or BSA systems.

An alternative method to neurocontrol is provided by the field of *genetic algorithms* (e.g. Goldberg, 1989). Genetic algorithms are another example of artificial learning methods inspired by the biological world. Briefly, information relevant to a problem is coded as a string of bits called a *chromosome*. An initial *population* of different chromosomes is generated randomly which provides a starting point for the "breeding" process. Changes in the population take place over distinct time periods, called *generations*, which give rise to new chromosomes. The changes—with mechanisms analogous to natural mechanisms observed in the science of molecular biology—take place through such processes as *crossover*, *recombination* and *mutation*.

At each generation, a new population of chromosomes is produced which represent a set of candidate solutions to the original problem. The number of each different type of chromosome is determined by *differential reproduction* between chromosomes which depends upon individual *fitness* parameters. Thus, a chromosome with a higher fitness value will tend to "breed" more rapidly than a chromosome with a lower one. The fitness value is determined by a chromosome's suitability as a solution to the original problem. After a number of generations, a population will exist which may contain an acceptable solution to the problem. If this is not the case, then more breeding cycles are required.

Control problems may be rendered solvable by genetic algorithm methods by coding the weights of a candidate neurocontroller to give a chromosome template (e.g. Wieland, 1991, Maricic, 1991) or by coding the parameters of a known system model to give the template (Howell, 1994). The cart-pole problem has been solved using both of these methods (Wieland, 1991; Maricic, 1991; Howell, 1994).

Howell (1994) used genetic algorithms in system identification to find suitable sets of parameters for two forms of controller model. The first form of controller model was specified to solve the simpler two state problem of balancing a pendulum constrained by the differential equation

$$\frac{4}{3}ml^2\ddot{\theta} = mgl\sin(\theta) + mlu\cos(\theta).$$

The optimal control problem involves using a genetic algorithm to find the parameters

$\alpha_0,\ldots,\alpha_5$ and $\beta_0,\ldots,\beta_5$ which specify a continuous controller of the form

$$u = \frac{\alpha_0 + \alpha_1 x_1 + \alpha_2 x_1^2 + \alpha_3 x_2 + \alpha_4 x_1 x_2 + \alpha_5 x_2^2}{\beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2 + \beta_4 x_1 x_2 + \beta_5 x_2^2}$$

such that the cost function $J = \sum_{t=0}^{5} x_1^2(t) + x_2^2(t) + u^2(t)$ is minimised.

The 12 control system parameters are discretised by coding them as 16 bit binary numbers covering the range -50 to +50; the resultant 192 bit string is the chromosome.

It was concluded that the genetic algorithm derived controller performed better in terms of cost than a controller obtained using a linearisation of the system (Howell, 1994).

When the cart dynamics were included (giving a four state problem) a controller of the form $u = \sum_{i=1}^{4} \alpha_i x_i + \sum_{j=1}^{4} \beta_j x_j^2 + \sum_{k=1}^{4}\sum_{l=1}^{4} \gamma_{kl} x_k x_l$ was used. The parameters $\gamma_{kl} = 0$ for some of the terms giving 14 parameters in total. The cost function in this case is $J = \sum_{t=0}^{5} x_1^2(t) + x_2^2(t) + x_3^2(t) + x_4^2(t) + u^2(t)$

Again, the developed controller was found to outperform a standard linear controller (Howell, 1994). However, the drawback is that the controller learns control over one region of state-space and not others; this is a gain scheduling problem for which Howell suggests evolving a controller for each chosen region

and storing the parameters. Overall control will then be maintained by switching between the different controllers.

Maricic (1991) offers another evolutionary approach to the cart-pole problem similar to Wieland's (1991) in which a neural network is optimised using genetic algorithms. The size and connectivity of the neural network is fixed; one node acts as the output. In effect, the genetic algorithm 'breeds' neural networks which signify the phenotype. The genotype, consisting of coded weights, is altered using genetic operators such as crossover and mutation. The neural network output is discretised in the range [-1.0,+1.0]. Results show that adequate control is obtained but the technique is not directly comparable to that of Barto, Sutton and Anderson (1983) or that developed in this thesis.

While neural network systems utilising the unsupervised learning method require neither explicit pattern pairs nor evaluative feedback *per se* to operate effectively, they are only able to organise input patterns by means of clustering methods and have no intrinsic means for adjusting control actions on the basis of environmental responses. External learning mechanisms have to be incorporated into candidate self-organising controllers based upon such clustering networks. These external mechanisms can, for example, involve the use of stimulus-response pattern pairs, a cost-function or scalar evaluative feedback.

A self-organising controller, based upon a Kohonen topology conserving network, was developed to learn the control actions of a teacher in supervised learning mode (Ritter *et al*, 1992). A variant akin to reinforcement learning, using only a reward signal based upon a specified cost function, has also been developed (Ritter *et al*, 1992). In both cases Kohonen's original learning algorithm (Kohonen, 1989) has been extended to incorporate an output value for each node of the network lattice. In the supervised case, the cart-pole problem has been solved by a teacher external to the network which acts as a look-up table following training. Although this method obviates the need for re-calculation of output values, the requirement for an external teacher limits the autonomy of the network.

The variant removes the requirement for an external teacher and computes desired outputs on the basis of a generalised reward signal derived from a system specific cost function. The network no longer has access to desired control outputs and forms a continuous mapping between state-space and control output space, with the control outputs being determined via a stochastic search process. During the search, the stored output value for a particular lattice node is allowed to converge to a desirable control action.

Both the supervised and the variant topology conserving controllers have a planar network lattice structure which is fixed *ab initio*. This places a restriction on the information capacity of the network through the determination of the state-space resolution by the size of the lattice. In other words, with too few nodes the control hypersurface will be coarsely defined. Too many nodes may reduce the parsimony of the network depending upon the size of the local update region with respect to the granularity of state-space coverage.

Fuzzy approaches, although relevant here, will be discussed in chapter 5 where the application of fuzzy techniques is reviewed briefly.

4.2 EUCART Description

4.2.1 Introduction

To solve a non-linear control problem using a neurocontroller, a method of representing state-space is required which allows the association of control actions with distinct state-space regions; the regions may overlap but they must be distinctly identifiable so that unique outputs may be assigned to them. A convenient set of methods of representing state-space that is compatible with the incremental learning paradigm involves Euclidean clustering (e.g. Kohonen, 1989, 1995). Individual states are assigned to a cluster and, in some cases, new clusters may be added as required. Euclidean clustering methods provide a convenient way of assigning cluster membership by comparing the distance between an input vector and various categories stored by the system. Category assignment based upon the Euclidean distance between inputs and category centres, or prototypes, results in a partitioning of state-space as shown in Figure 4.1 if winner-takes-all dynamics are used.

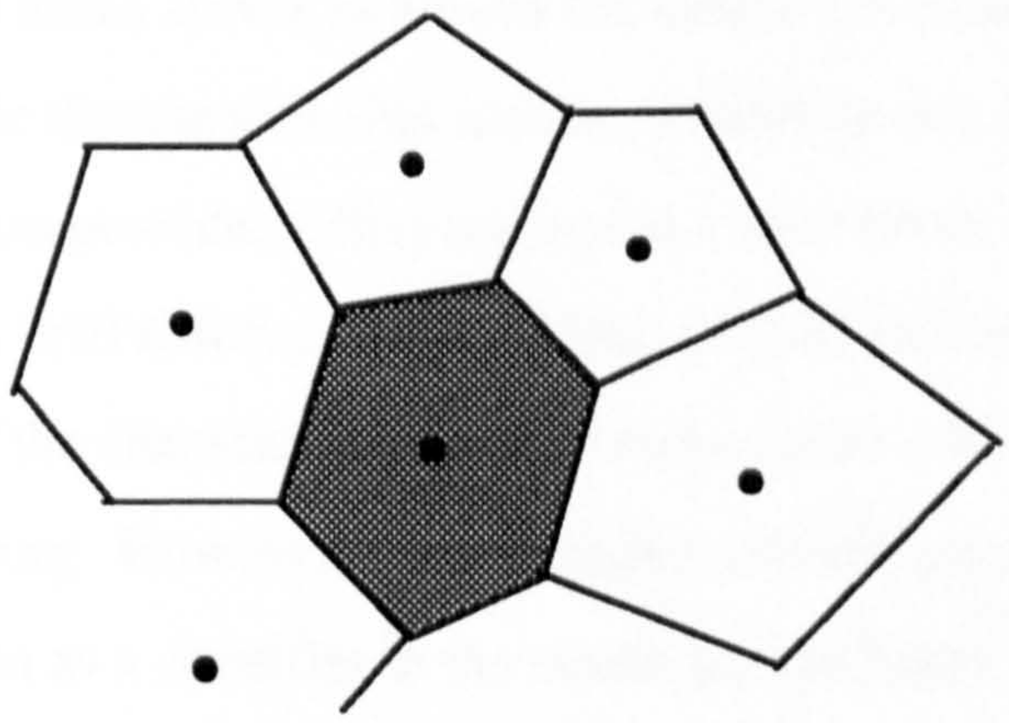


Figure 4.1. Using Euclidean clustering with winner-takes-all dynamics results in a state-space partitioning that consists of irregular convex regions. The regions are comprise intersecting hyperplanes that represent the decision surface between neighbouring category centres.

Inputs are assigned to the category represented by the nearest category centre; this results in a unique assignment for each input unless two or more category centres are equidistant from an input vector, which is unlikely. Category centres may be represented by nodes within the neurocontroller. Following competition

between category nodes, either an overall winner or a selection of active nodes may be chosen to compute a control output. In the latter case, a method of weighting the nodal contributions is required; the weighting is usually a function of the category node activation.

Here, a novel Euclidean clustering method, inspired by some of the attractive properties of fuzzy ART (Carpenter, Grossberg and Rosen, 1991), is presented that overcomes the problem of *category drift* (Moore, 1989) and allows incremental learning of state-space information without supervision. This method is compatible with the BSA formulation of reinforcement learning, and is implemented as a state-space decoder that replaces the fixed structure of Barto *et al*, (1983).

4.2.2 Fuzzy and Euclidean Clustering

The fuzzy ART system has many desirable properties of which a subset can be abstracted for the purpose of designing a state-space decoder. Framing this subset in Euclidean terms serves as a basis for further developments in decoding schemes. As will be discussed in this section, emulating one particular aspect of fuzzy ART operation provides a first attempt at a solution to the problem of Euclidean category drift (Moore, 1989). Other properties framed in Euclidean terms lose some of the characteristics which make fuzzy ART particularly good at unsupervised learning. However, the Euclidean network presented here is not designed to function as a classifier in the sense that the fuzzy ART system is, and further development would produce a closer functional relationship between the fuzzy and Euclidean clustering schemes if required. The object is not simply to have a Euclidean form of fuzzy ART, if that were indeed possible; fundamental differences in Euclidean and fuzzy metrics restrict operational correspondences in networks to functional analogies.

4.2.3 The EUCART System

EUCART (Marriott and Harrison, 1995, 1996) is a novel Euclidean self-organising state-space decoder based loosely on Fuzzy ART (Carpenter, Grossberg and Rosen, 1991), hence the name, from EUClidean ART. Its purpose is autonomously to structure state-space so that the ASE / ACE sub-units may associate control actions with individual state-space regions through reinforcement learning. The main property of fuzzy ART incorporated into EUCART is the *category growth property* which prevents category templates or prototypes from wandering (Figure 4.2). The category growth property allows the new category to incorporate the region of state-space encompassed previously by the category by expanding outwards towards the input vector up to a maximum possible extent. The existing members of the cluster will always remain within the category.

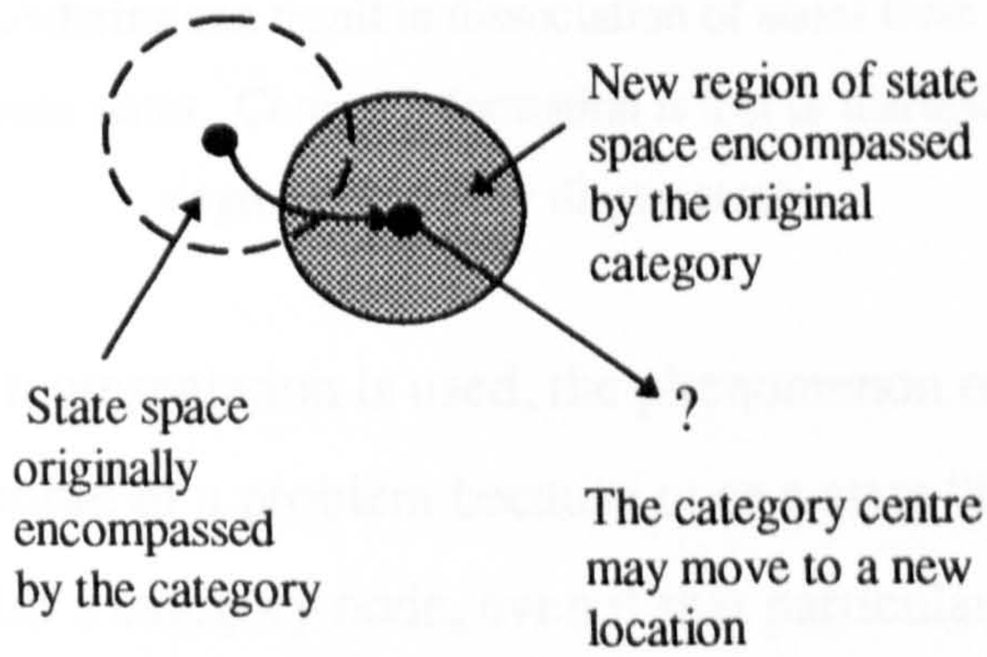


Figure 4.2. A schematic illustration of the phenomenon of category wandering. As a given category centre is updated by a stream of inputs assigned to the category that it represents, the location of the category centre moves and encompasses a new region of state-space. Consequently states that belonged to a particular category may not belong to it any longer.

Category drift may cause degradation of performance in control applications where information is lost through the movement of categories. For example, it is possible that some states have control actions associated with them when assigned to a particular category, but lose these associations when the category moves. Consequently, either no control actions are available for such "displaced" states, or , different control actions are assigned following the category movement. For winner-takes-all dynamics, gains in using the category growth property may be

offset by the plasticity of category assignment where states are reassigned to nearer category centres during learning. Using the category growth property prevents the case where states lose any associated output altogether by ensuring that states always remain members of the category extent of one or more category nodes once they come within the category boundaries; where there is multiple membership, an overall winner may be chosen. The problem of state dissociation is shown in Figure 4.3.

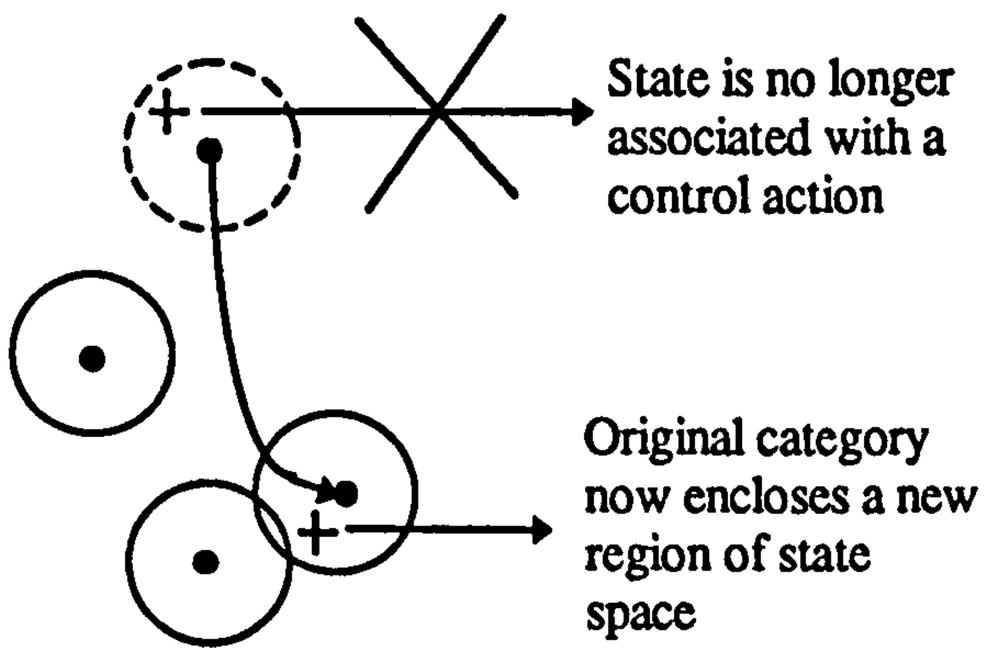


Figure 4.3 Category wandering can result in dissociation of states from control actions that are learned responses to these states. Control information is lost or disrupted depending upon the degree of category displacement.

Where a distributed representation is used, the phenomenon of category wandering presents more of a problem because once a state "informs" a control output associated with a category node, even if that particular category node is not the overall winner next time, it still contributes proportionately to the output because states may belong to one or more categories. Thus, when states are dissociated from categories that they have informed (i.e. they have modified associated category information) when these states or regions of state-space are reactivated, information relating to the control output is lost. This is illustrated in Figure 4.4.

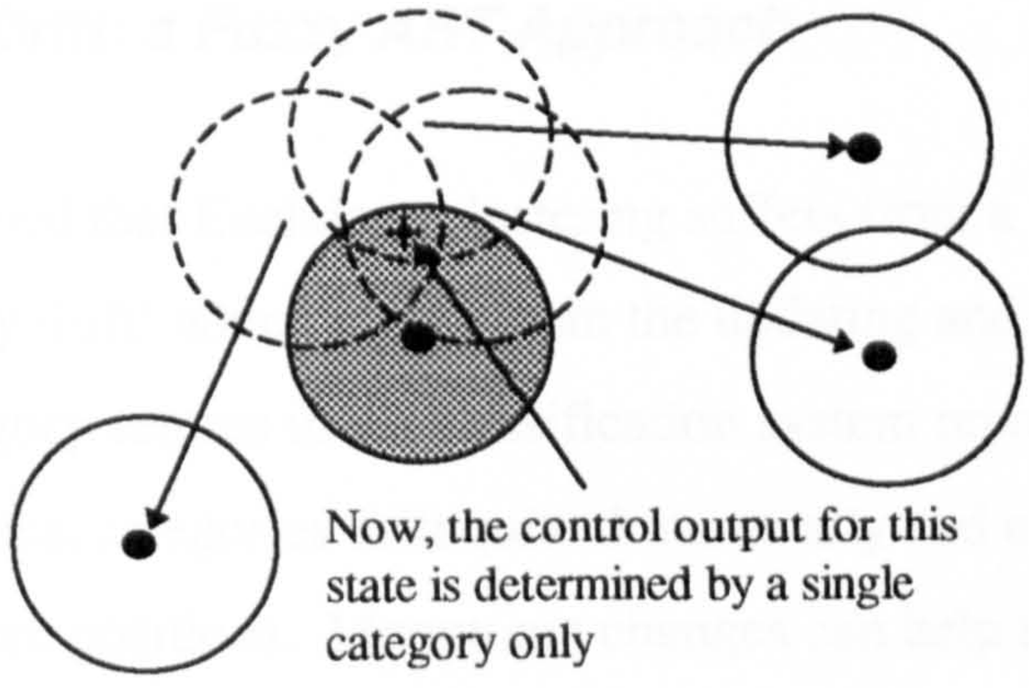


Figure 4.4. The effect of category wandering when a distributed representation is used. Categories that previously contributed to the control output no longer have any effect. The category growth property ensures that, for a distributed representation, once a region of state-space contributes to a category and its associated control action, it will always continue to do so.

The EUCART system retains the ARTfields F_0, F_1 and F_2 but differs in the dynamic operation of the latter two, especially in the case of the matching field, F_1. Unlike Fuzzy ART, EUCART does not use complement coding. The input is given by $\mathbf{I} = (I_1,\ldots,I_M)$,. a subset of the real line, , $\forall i = 1,\ldots,M$.

Note that dropping complement coding removes the unit normalisation restriction. That is, inputs do not have to remain within the unit hypercube. However, EUCART inputs have to be within a fixed input space (hypercuboid) of specified dimension. In this paper we confine the inputs within the unit hypercube, $[0,1]^M$ for convenience. Each F_2 node represents a class or category of inputs and operates in winner-takes-all mode as before.

Associated with each F_2 node j is a set of adaptive weights $\mathbf{w}_j = (w_{j1}, w_{j2},\ldots,w_{j2M}), \quad \forall j = 1,\ldots,N$. These weights store the network LTM traces in a form allowing emulation of a fuzzy ART property which prevents category wandering. Two vectors of length M are stored as a single vector representing the minimum and maximum extents of category growth in elements 1 to M and M+1 to 2M respectively. Before discussing this mechanism it instructive to look at both Euclidean and fuzzy categorisation in more detail.

4.2.4 Category Drift: a Fuzzy ART Approach

It has been mentioned that Euclidean clustering suffers from a phenomenon known as 'category drift' which results from the updating and subsequent movement of category centres in the classification system representation of input space. In some cases, categories drift quite dramatically and even re-occupy previous class centre positions. Monotonic changes can help to rectify this problem but can introduce problems of a different type (Moore, 1989). Fuzzy ART gets around this problem by using complement coding. Complement coding allows a category to grow by incorporating previously enclosed space within the new category extent. The category growth property of fuzzy ART ensures that categories do not drift and occupy different areas of state-space.

The weight vector of a fuzzy ART category is given by

$$\mathbf{w}_j = (\mathbf{u}_j, \mathbf{v}_j^c)$$

where $\mathbf{u}_j$ and $\mathbf{v}_j^c$ are non-complement coded and complement coded

respectively. Note that $\mathbf{v}_j^c$ is not necessarily the complement coded form of $\mathbf{u}_j$. When the new category is first created, $\mathbf{v}_j^c$ is the complement coded form of $\mathbf{u}_j$ but, during operation, sometimes only $\mathbf{u}_j$ is replaced by the new input following the fuzzy AND operation and at other times, $\mathbf{v}_j^c$ is replaced depending upon the new input vector.

As illustrated in Figure 4.5, $\mathbf{u}_j$ and $\mathbf{v}_j^c$, for a two dimensional system, represent the extent of the current category if the size of the rectangle R_j is defined as

$$|R_j| = |\mathbf{v}_j - \mathbf{u}_j|$$

If a new input $\mathbf{a}$ is used to update the J^{th} winning category of extent $|R_J|$, the updating operation, signified by $\oplus$, gives

$$|R_J \oplus \mathbf{a}| = |(\mathbf{a} \vee \mathbf{v}_J) - (\mathbf{a} \wedge \mathbf{u}_J)|.$$

This comes from applying the FCFR learning equation (β=1.0), using input $\mathbf{a}$, to the weight vector $\mathbf{w}_J$.

If the new category $|R_J \oplus \mathbf{a}|$ is too large then fuzzy ART resets and searches for a new category. It can be shown (Carpenter Grossberg and Rosen, 1991) that, for an M dimensional input space,

$$|R_J \oplus \mathbf{a}| \leq M(1-\rho) \qquad (4.1)$$

Thus, the growth of a particular category is limited by the vigilance parameter as would be expected.

An important point to note is that the weight values do not signify the centre of a category in the normal sense of clustering procedures; they signify the 'extent' values which allow the growth of a category to include previously encompassed values of weight space. Categories do not move but grow until a specified upper limit of category size is reached.

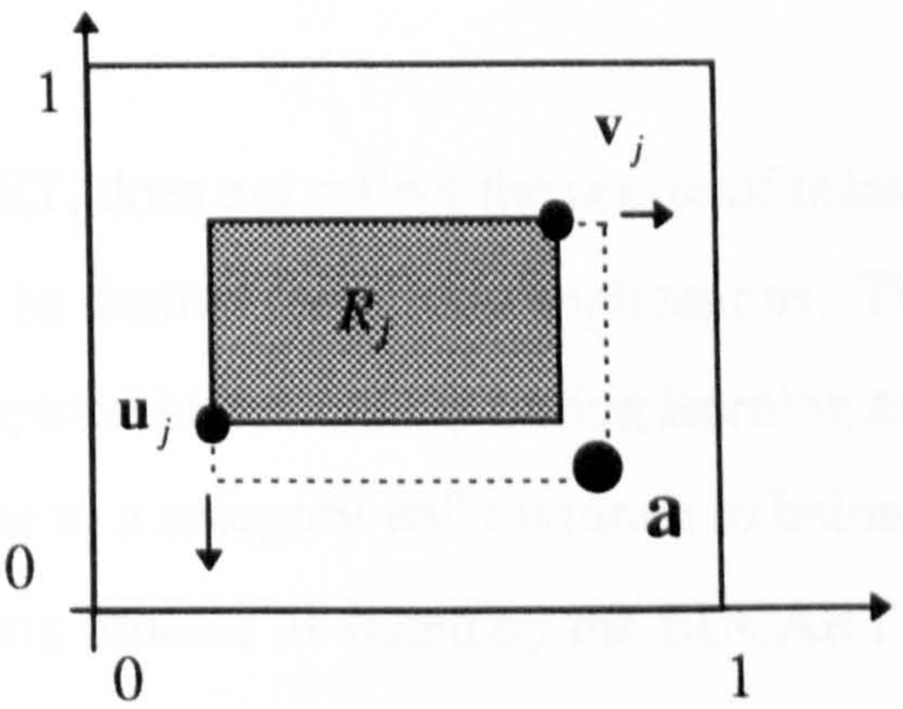


Figure 4.5. Fuzzy ART clustering and category growth illustrating the category growth property. (After Carpenter, Grossberg and Rosen, 1991). Categories are represented by 'extent markers' and not centres; the latter are not meaningful when using the L^1 norm which is more suited to fuzzy operations than is the Euclidean norm.

4.2.5 EUCART Categories

EUCART weight values are given by $\mathbf{w}_j^E = (\mathbf{u}_j^E, \mathbf{v}_j^E)$, where $\mathbf{u}_j^E$ and $\mathbf{v}_j^E$ are the minimum and maximum category extent markers with components given by $u_{jk}^E(t+1) = \min(u_{jk}^E(t), a_k)$, and $v_j^k(t+1) = \max(v_j^k(t), a_k)$ respectively. These 'poles' moving in 'opposite' directions in M-dimensional hyperspace delimit hyperspace categories whose extent is given by

$$e_j = \left\| \mathbf{v}_j^E - \mathbf{u}_j^E \right\|,$$

where $\| \, . \, \|$ is the Euclidean or L^2 norm (Euclidean distance between two vectors); see Figure 4.6. Analogously to equation (4.1) the category growth criterion for normalised Euclidean space becomes

$$e_j \leq \frac{\sqrt{M}}{2}(1-\rho_E) \qquad (4.2)$$

where $\sqrt{M}$ is the maximum possible Euclidean distance between points in $[0,1]^M$ and ρ_E is the vigilance parameter of EUCART. Analogously to fuzzy ART, for high vigilance, i.e. $\rho_E \to 1$, the resulting categories are very small and for low vigilance, i.e. $\rho_E \to 0$ they are very large.

The centre of a category is given by

$$\mathbf{c}_j = \frac{1}{2}(\mathbf{u}_j^E + \mathbf{v}_j^E)$$

and, as with fuzzy ART, does not reflect the centre of mass (centroid) of the category, which may be desired for certain applications. The centre of mass of a category may be computed incrementally during learning as required. All input vectors that contribute to a category will continue to belong to that category throughout the learning process as stated by the EUCART Category Composition Theorem.

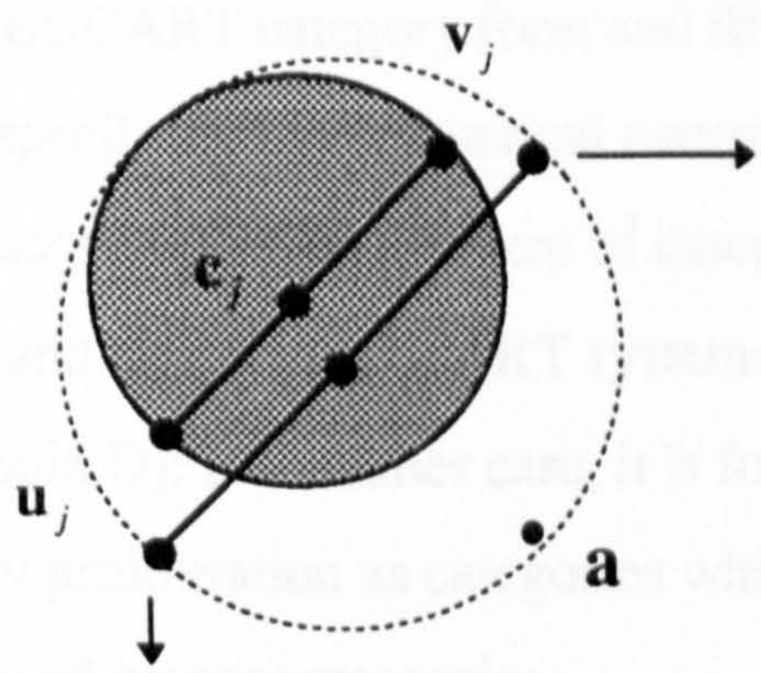


Figure 4.6. EUCART clustering using the category extent markers analogous to those of Fuzzy ART. When a new input extends the category boundaries, the category centre will also change. However, the subset of input space encompassed by the previous category remains within the new category.

An input, $\mathbf{I}$ is said to be a member of the i^{th} category if

$$\|\mathbf{c}_i - \mathbf{I}\| \leq \frac{\sqrt{M}}{2}(1-\rho_E)$$

which forms the EUCART match criterion. This form of match criterion, unlike equation (2.14) of fuzzy ART does not take into account the absolute magnitude of the input vector. In the context of state-space partitioning this is not particularly important as the main focus of interest is upon absolute distances from a state-space exemplar. In pattern recognition tasks, however, the absolute magnitude must be taken into account so that patterns are matched according to the degree of correlation between them. For example, an input may be closer to a signal category in terms of absolute magnitude but have a much smaller correlation in terms of vector direction, i.e. dot product. Thus, a category with a smaller exemplar vector magnitude but nearer in terms of angle may well be the desired category. This important point is reflected in fuzzy ART by the dual choice and matching functions, and the search mechanism. Note that choosing a winner in terms of distance alone is not equivalent to finding the largest net input by using the dot product, unless the exemplar weights are normalised.

4.2.6 EUCART Category Containment

Section 4.2.6 describes the EUCART category form and illustrates the use of category extent markers in specifying hyperspherical categories analogous to the hyperrectangular ones of fuzzy ART. The problem of category drift was outlined in sections 4.2.3 and 4.2.4. and also occurs in ART systems which do not use complement coding (Appendix D); in the latter case, it is for a slightly different reason and leads to category proliferation as categories which have drifted towards the origin are replaced by new categories.

One of the major justifications for an architecture such as EUCART is that it both confers the benefits of using a Euclidean metric and prevents category drift. EUCART prevents category drift in the sense that once inputs are assigned to a category, they always remain within the confines of that category. Even though

the category boundaries change, no input, once categorised, is ever "left behind" by a wandering category.

The hyperspherical category growth is not monotonic in the sense that established hypervolumes are necessarily contained in later hypervolumes. However, as proved in the EUCART Category Composition Theorem of this thesis, all members of a given category remain within the boundaries of that category from the moment of categorisation onwards if fast learning ($\beta=1$) is used. This property is important for problems where actions or outputs are associated with regions of input space or state-space.

A semi-formal proof of the EUCART Category Composition theorem is developed in Appendix G and will forms the basis for the Centroid Inclusion Theorem of section 4.2.13 which shows that the inclusion of centroid information—concerning the distribution of inputs within a category—within the EUCART architecture is possible.

The EUCART Category Composition Theorem :

All inputs that are members of a given hyperspherical category remain within that category throughout the category growth process and beyond.

In other words, for an input space (state-space), X

$\forall \mathbf{x}_\tau \in X$ for some time, τ and some hyperspherical category, $C_\tau^{(i)}$ at time τ,

$\mathbf{x}_\tau \in C_\tau^{(i)} \Rightarrow \mathbf{x}_\tau \in C_{\tau+n}^{(i)}, \quad \forall n > 0$

4.2.7. The Fuzzy Choice and Matching Functions Revisited

The fuzzy ART choice function of equation (2.13) approximates to

$$T_j(\mathbf{I}) \approx \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{|\mathbf{w}_j|} \qquad \text{with } \alpha \to 0.$$

This measures the extent to which $\mathbf{w}_j$ is a fuzzy subset of $\mathbf{I}$ (Zadeh, 1965; Kosko, 1992).

Where $\mathbf{w}_j$ is a fuzzy subset of $\mathbf{I}$, $\frac{|\mathbf{I} \wedge \mathbf{w}_j|}{|\mathbf{w}_j|} = 1$ and equation (2.13) becomes

$$T_j(\mathbf{I}) = T_j(\mathbf{w}_j) = \frac{|\mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} \qquad (4.3)$$

with $T_j(\mathbf{I}) \approx 1$ as $\alpha \to 0$.

$|\mathbf{w}_j|$ can be maximised up to a maximum value at $\mathbf{w}_j = \mathbf{I}$ to give the highest choice function for fuzzy subsets of $\mathbf{I}$ through the monotonic increasing property (M.I.P.) of equation (4.3); this property is proved informally in appendix B. The fuzzy ART match function of equation (2.14) has a value of

$$\frac{|\mathbf{I} \wedge \mathbf{w}_j|}{|\mathbf{I}|} = \frac{|\mathbf{w}_j|}{|\mathbf{I}|} \qquad (4.4)$$

for $\mathbf{w}_j$ a fuzzy subset of $\mathbf{I}$. So, for this special case, given the set of fuzzy subsets of $\mathbf{I}$, denoted by $\Omega_{\mathbf{I}}$, for $\mathbf{w}_j, \mathbf{w}_k \in \Omega_{\mathbf{I}}$

$$T_k(\mathbf{I}) \geq T_j(\mathbf{I}), \quad \Rightarrow \quad |\mathbf{w}_k| \geq |\mathbf{w}_j|, \quad \forall j \neq k$$

by the M. I. P.

Now, for $|\mathbf{w}_k| = \max_j \{|\mathbf{w}_j|\}$, if reset occurs, no more matches can be found since, from equation (4.3)

$|\mathbf{w}_j| \leq |\mathbf{w}_k| \Rightarrow \frac{|\mathbf{w}_j|}{|\mathbf{I}|} \leq \frac{|\mathbf{w}_k|}{|\mathbf{I}|}$, and, $\frac{|\mathbf{w}_k|}{|\mathbf{I}|} < \rho \Rightarrow \frac{|\mathbf{w}_j|}{|\mathbf{I}|} < \rho, \quad \forall \mathbf{w}_j \in \Omega_{\mathbf{I}}$ for some $\mathbf{w}_k \in \Omega_{\mathbf{I}}$.

Thus, no further search for fuzzy subsets is required. Other searches will follow for cases when $\mathbf{w}_j \notin \Omega_{\mathbf{I}}$.

If EUCART is given the choice function

$T_j^E(\mathbf{I}) = 1 - \dfrac{\|\mathbf{c}_j - \mathbf{I}\|}{\sqrt{M}}$, where $\mathbf{c}_j = \dfrac{1}{2}\left(\mathbf{u}_j^E + \mathbf{v}_j^E\right)$ as before, for a normalised space, $[0,1]^M$ then $T_j^E(\mathbf{I}) \in [0,1]$ and a match criterion based purely upon absolute distance $\|\mathbf{c}_j - \mathbf{I}\| \leq \dfrac{\sqrt{M}}{2}(1-\rho_E)$ removes the necessity of further search if the criterion is not fulfilled. In other words, $T_k^E(\mathbf{I}) \geq T_j^E(\mathbf{I}) \Rightarrow \|\mathbf{c}_j - \mathbf{I}\| \geq \|\mathbf{c}_k - \mathbf{I}\|$, $\forall j \neq k$

and failing the match criterion gives $\|\mathbf{c}_j - \mathbf{I}\| \geq \|\mathbf{c}_k - \mathbf{I}\| > \dfrac{\sqrt{M}}{2}(1-\rho_E)$ which implies that $\|\mathbf{c}_j - \mathbf{I}\| > \dfrac{\sqrt{M}}{2}(1-\rho_E), \quad \forall j$

Thus, the search for a new winner is not required because no better match, in the sense of Euclidean distance, can be found. The simplified choice and match functions of EUCART do not take the magnitude of the input vector into account and so remove the need for a more complex search pattern. The more complex search is required to find a new input with roughly the same relative spatial pattern regardless of the absolute magnitude. These simplified dynamics suffice in the present control context because stored patterns reflect state values and correlation of an input with its canonical or exemplar state is based purely upon absolute distances. As mentioned above, more sophisticated pattern clustering, using a Euclidean metric, requires some form of angle or dot product measure to assess input correlation with stored exemplars to improve clustering properties. This correlation measure allows matching independently of signal magnitude, For example, in the clustering of visual data, it is desirable that patterns can be clustered within a sample space of varying background illumination, with respect to relative reflectance patterns.

4.2.8 EUCART Learning

Learning in EUCART is analogous to learning in fuzzy ART and LTM changes are made according to

$$\mathbf{u}_J^{(Enew)} = \beta_E(\mathbf{I} \wedge \mathbf{u}_J^{E(old)}) + (1-\beta_E)\mathbf{u}_J^{E(old)} \qquad (4.5a)$$

and,

$$\mathbf{v}_J^{E(new)} = \beta_E(\mathbf{I} \wedge \mathbf{v}_J^{E(old)}) + (1-\beta_E)\mathbf{v}_J^{E(old)} \qquad (4.5b)$$

where $\wedge$ and $\vee$ are the fuzzy AND and OR operators respectively (Zadeh, 1965). Equations (4.5a) and (4.5b) are used to find the new min and max category extent markers respectively provided that the category growth criterion of equation (4.2) is not violated by the updated extent markers; if this criterion is violated then the update is not carried out. The *Fast-Commit-Fast-Recode* and *Fast-Commit-Slow-Recode* options are retained in EUCART.

4.2.9 A Comparison with Fuzzy ART

Figure 4.7 illustrates the fuzzy ART choice function of equation (2.13) as a function of the two exemplar weights for a two-dimensional non complement-coded input. Where both weight values are less than their respective input values, the choice function has a plateau of approximately unity. Outside of this plateau region, the choice function decays with increasing weight magnitude.

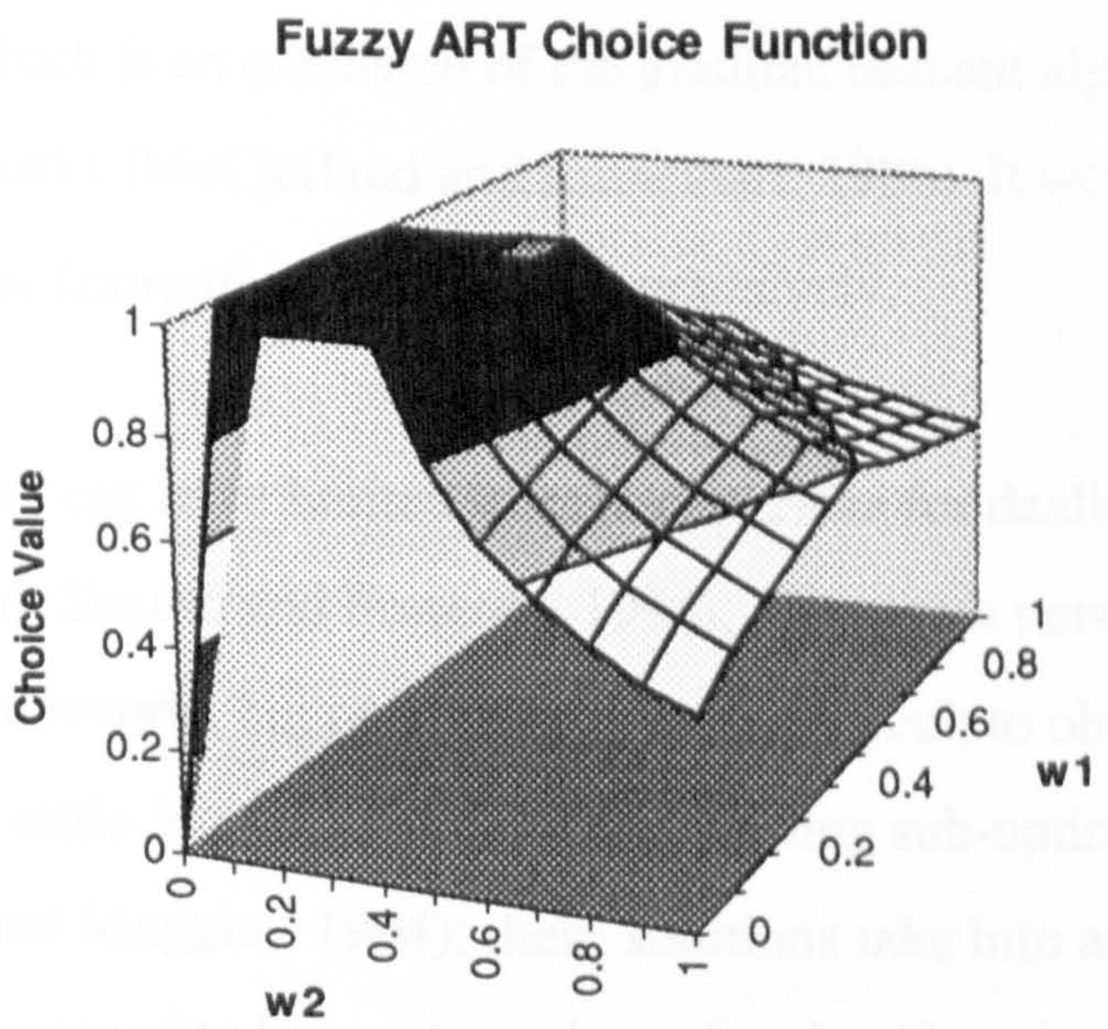


Figure 4.7 An illustration of the fuzzy ART choice function for a two dimensional weight space.

The Euclidean choice function of EUCART is more representative of state-space and does not suffer from the "near discontinuity" between the plateau region and the remainder of weight space. Any function of the fuzzy ART choice function would have a small range within a region of state-space determined by an input; this would not be representative of the variation in distance between the input vector and the weight vector because the choice function is a non-linear function of that distance. Problems might arise when weighting node contributions on the basis of the distance between a given input and neighbouring nodes when using a distributed representation of information within a network.

4.2.10 A Comparison with Other Architectures

Static networks—networks with no recurrent connections—have a finite impulse response and cannot store information for an indefinite amount of time.

Recurrent neural networks have an internal state that is capable of representing contextual information. Learning algorithms for recurrent networks are mostly generalisations of existing learning algorithms for static networks; an example is the *backpropagation through time* algorithm (Werbos, 1990; Srinivasan, Prasad and Rao, 1994) which is an extension of the gradient descent algorithm for feedforward networks (McClelland and Rumelhart, 1986) It works by storing unit activations and computing the gradient recursively.

Recurrent networks can have limited storage capacities for dealing with input sequences (Bengio, Simard and Frasconi, 1994). Recurrent networks generally outperform static networks but optimality is more difficult to obtain (Picton, 1994) and readily settle into local minima representing sub-optimal solutions (Bianchini, Gori and Maggini, 1994); these solutions take into account short term dependencies as opposed to longer term dependencies (Bengio, Simard and Frasconi, 1994). The difficulties involved in training recurrent neural networks are possibly responsible for their slow adoption in the field of control engineering.

A partially recurrent network, the *Elman network*, (Elman, 1990) is slowly increasing in popularity for system modelling and identification (Picton, 1994); it is especially useful for modelling dynamical systems with temporal correlation and delay effects. An Elman network consists of three layers plus a context layer (Figure 4.8). Only the middle layer is recurrent which gives the architecture advantages over fully recurrent nets. A modified form of backpropagation can be used to train it (Elman, 1990; Picton, 1994)

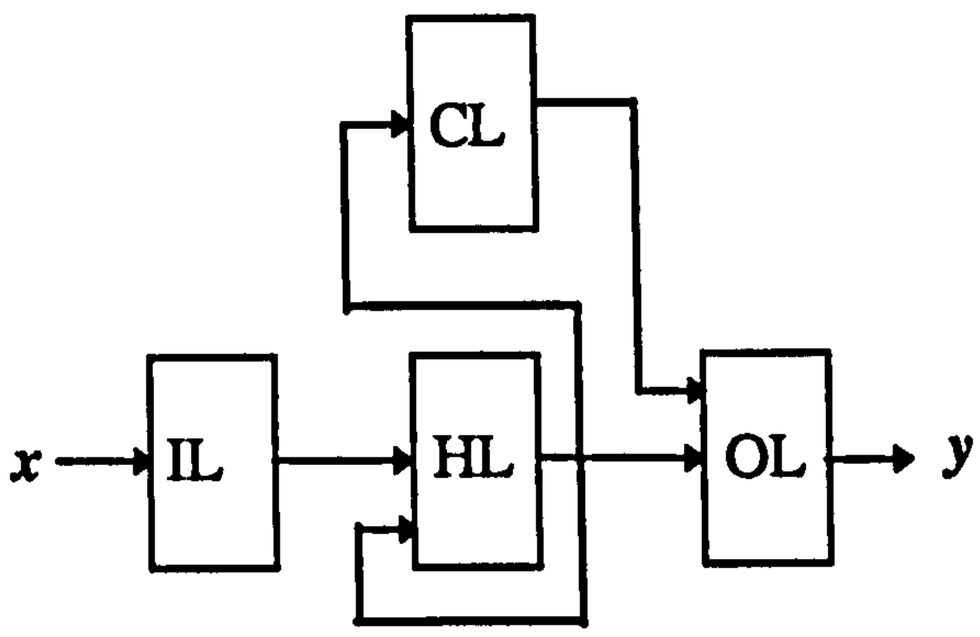


Figure 4.8. An Elman network which consists of three feedforward layers and a context layer which provides a delay of one sampling period.

Connections to context units are fixed and provide a delay of one sampling period. Outputs of the hidden layer represent the state which is then fed to the context units. The output of the context units is fed to the output layer. The output of the net represents a function of the current state and the previous state.

It is difficult to see how such a system will take into account long-term dependencies such as those encountered with failure-driven systems with expanding time horizons.

4.2.11. Cascade Correlation

An example of an incremental learning system with self-organising capabilities is the *Cascade-correlation* architecture of Fahlman and Lebiere (1990). It was developed because current learning algorithms—mostly based around backpropagation using feedforward networks—were seen as slow. The lack of speed was thought, in part, to be responsible for the lack of widespread application of neural networks.

The main problem with algorithms such as backpropagation is that *all* weights have to change and it is difficult to add or adapt individual nodes. The use of global learning often results in wasted effort through inefficiency of adaptation. The cascade-correlation network was developed to overcome these difficulties.

There are two key ideas that underlie the cascade-correlation neural network viz. A *cascaded* architecture: where nodes are added incrementally as required and the maximised *correlation* between the outputs of new units and the residual error signal. New units are added to the network, one by one, when required . The weights on the incoming connections are fixed and adaptation occurs only on the output connection weights.

The learning strategy is incremental and allows progression to high-order feature detection; learning begins with no hidden units and direct input-output training. After a number of training cycles (off-line), a minimum error level is approached asymptotically at which there is no significant reduction in error for each extra cycle. If the error is less than or equal to a prescribed limit the process stops, otherwise, a hidden unit is added to reduce the residual error and the process is repeated.

More than one new hidden unit can be used in the form of a *candidate pool* where a winner is chosen on the basis of residual error reduction.

When a new node is added, the input weights are frozen and the node output weights are trained so as to reduce the residual error. Local training is used instead of global training with the new node being trained separately form the rest of the network to reduce the residual error without disrupting the remainder of the network. The network weights are frozen whilst a suitable set of input weights for the new unit are found, then the new unit input weights are frozen and full network training is resumed.

The many advantages of the cascade-correlation network over "standard" feedforward networks—such as the multilayer Perceptron—using backpropagation include (Fahlman and Lebiere, 1990):

1. the network size / topology is not required in advance,
2. there is a sizeable reduction in training time;
3. high order feature detectors can be constructed without "slowdown"
4. the cascade-correlation architecture can be used for incremental learning in which new information is added to the trained net
5. only one layer is trained at a time
6. no backpropagation of error signals is required; weighted connections are unidirectional (biologically plausible)
7. there is no interaction of new node candidates; each candidate sees input and output and the connections are limited, thus making parallel implementation possible.

Other incremental learning architectures using a single layer are not good for some problems which require higher-order feature detectors with interconnected non-linear layers (Fahlman and Lebiere, 1990).

The cascade-correlation architecture does offer advantages over architectures such as the multilayer Perceptron and in certain mapping tasks will outperform it. However, certain fundamental objections remain when considering the use of this architecture for control problems. The cascade correlation architecture uses a

modified form of the gradient descent method which is off-line and requires numerous passes through a pre-specified training set. Supervised learning is also required to provide an output vector—for each input vector—so that the residual error can be computed to provide a training signal for the addition of new nodes. Furthermore, although nodes can be added to the cascade correlation network, it is difficult to see how nodes can be removed to reduce the load caused by under-utilised nodes.

4.2.12 Fritzke's Growing Cell Structures

Fritzke (1991,1993, 1994) proposes an incremental self-organising network, the *growing cell structures* (GCS) which, like Kohonen's SOM, maintains a topological relationship between network nodes and the input space.

The main difference between the SOM and the GCS is that for the SOM, the structure and size of the network are predetermined, but for the GCS, nodes may be added or removed. The GCS network is an improvement over the SOM (Fritzke, 1993) and has both unsupervised and supervised versions.

Cells or nodes of the GCS network are arranged in R^N and are not restricted to a planar structure. The topological relationship between the cells is maintained by an algorithm which constructs a topological *complex* from hypertetrahedral *simplices* according to the distribution of input data; simplices may be added or removed as required.

Like the SOM, the GCS net uses competitive learning to find the nearest node—in terms of Euclidean distance—for each input presentation. The winning node will be a member of at least one simplex. The set of nodes comprising the GCS network defines a Voronoi tessellation; each node is responsible for a Voronoi region within which it is the winner. The winning node is updated to the

maximum extent and the direct topological neighbours of the simplex are updated to a lesser extent.

Activity traces are maintained for each cell; the winning node trace is increased whilst all others are decreased. The maintenance of traces allows the underlying probability distribution to be estimated by calculating the relative signal frequency of the cells, that is, the ratio of the cell trace to the total trace. Where the trace is high, a new cell may be added and the trace values redistributed. Where it is low, cells can be removed. The algorithm for the addition and removal of cells maintains the topological structure of the complex, i.e. it will always consist of connected hypertetrahedral simplices.

The supervised version uses a radial basis function centred on each of the cell exemplars (weight vectors). It was bench-marked against a MLP and a cascade correlation network using the two spirals problem (Carpenter *et al* 1992). Results showed that the GCS network required many fewer training epochs. This result could be misleading, however because the computational complexity of each epoch for each of the networks may not be comparable.

4.2.13 Why Use EUCART?

Like the cascade correlation and GCS networks, EUCART does not require the size and topology to be specified in advance of training. Learning in EUCART is incremental and adapts to incorporate new information on-line for an indefinite period; there is no distinguishable training phase as with many network architectures such as the multilayer Perceptron and many of its derivatives.

The EUCART structure of individual nodes means that processing is strictly local and only one node is involved at any one time where winner-takes-all processing is used; Backpropagation of error signals during global learning is not required.

The direct topological relationship between the EUCART node locations and the state-space regions they represent, means that complex spatial information regarding node connections is not required; node neighbourhoods are simply chosen on the basis of Euclidean distance.

EUCART has a simple structure when compared to the GCS network. Is all the complexity of the GCS network required for this task? There may be no need for the topological information in this case. The GCS network requires the supervised learning variant if it is to be used to solve the cart-pole problem.

It is possible that the GCS network could be used as a self-organising decoder in the way that EUCART is. Even though the topological structure is maintained by the GCS network it will still suffer from category drift because the state-space categories would be based around the exemplar vectors represented by the node centroids. Although EUCART suffers from the problem of state reallocation when winner-takes-all competition is used, a distributed version, which combines information from more than one node, would reduce the impact because all categories to which a given state belongs will be used to produce the output. This is not the case with the GCS network which does not maintain the property of the EUCART category composition theorem and so would continue to suffer from the state reallocation problem even with distribution of state information. The GCS has no category extent markers to delimit the set of input vectors belonging to each category; thus there is no way of telling whether or not a given input actually belongs to a given category, owing to a set of inputs which have determined the extent markers, or is assigned to a category on the grounds that the category centre has moved the category to where the given input vector is located.

To some extent, EUCART is a "rough and ready" method of obtaining a state-space partitioning; it is certainly not the last word on self-organising nets, nor is it meant to be. In effect, it is a working algorithm shell to which improvements may be made and from which variations may be developed.

One possible variation on EUCART may be to use direction information when comparing input and exemplar vectors; this would be analogous to the match condition of fuzzy ART which is separate from the choice function. By comparing directional information, EUCART could maintain clusters based upon groupings of spatial patterns, that is, patterns not simply related by absolute Euclidean distance. A typical pattern matching cycle would consist of finding the nearest exemplar to the input pattern and then checking for a match in terms of spatial information. If the nearest exemplar pattern did not match sufficiently, then the node could be inhibited and a search triggered to find the next nearest stored pattern which may be a closer spatial match.

It is acknowledged that centroid information of the type found in competitive vector quantising and probabilistic neural network architectures, might be of importance for a large number of tasks (Specht, 1990; Lim and Harrison, 1995); for the quantisation of state-space considered here, this issue might not be critical where optimal state-space partitioning for control does not depend directly upon the distribution of input data. For example, successful control applications have been achieved with both pre-set and self-organising schemes (e.g. Barto, Sutton and Anderson, 1983; Hormel, 1990; Ritter *et al*, 1992).

However, it can be postulated that the use of centroid information may give an improved partitioning of state-space and, consequently, improve the control. The next section explores the option of adding centroid information to alter the operating characteristics of EUCART to make it more suitable for other application areas and, possibly, improve control performance.

4.2.14 Adding Centroids

The inclusion of centroid information often allows a category to represent the contained data in a more meaningful way. For example, a Gaussian kernel

function could be placed at the "true" centre of the category (the centroid) to indicate set membership. More simply, category membership could be made proportional to the distance between an input and the relevant category centroid. The following theorem shows that the EUCART category centroid will always remain within the category and so will be meaningfully associated with that category. The centroid can then be used in any algorithm which requires centroid information.

The EUCART Centroid Inclusion Theorem:

A EUCART category centroid , $\mathbf{w}^c(t)$, defined by the incremental learning rule

$$\mathbf{w}_i^c(t+1) = \mathbf{w}_i^c(t) + \alpha\left(\mathbf{I}(t) - \mathbf{w}_i^c(t)\right) \qquad (4.6)$$

where α is a learning constant such that $0 \le \alpha \le 1$ will always remain within the hyperspherical category, $S_t^{(i)}$ for all t.

Proof:

Assume, without loss of generality, that $\mathbf{w}_i^c(t) \in H_t^{(i)}$. That the input is contained in the hyperrectangle $\mathbf{I}(t) \in H_{t+1}^{(i)}$ is ensured by the category growth process. The hyperrectangle, $H_{t+1}^{(i)}$ is a convex set. Let $\mathbf{w}_i^c(t)$ and $\mathbf{I}(t)$ be the endpoints of a line defined by $\mathbf{r}(\eta) = \mathbf{w}_i^c(t) + \eta\left(\mathbf{I}(t) - \mathbf{w}_i^c(t)\right)$ (4.7)

where $0 \le \eta \le 1$. Setting η=0,1 gives the extremes $\mathbf{w}_i^c(t)$ and $\mathbf{I}(t)$ respectively. From lemma 3 of Appendix G, $H_t^{(i)} \subseteq H_{t+1}^{(i)}$ and from the definition of a convex set, $\mathbf{r}(\eta) \in H_{t+1}^{(i)}, \forall \eta \in [0,1]$

For some time, t equation (4.6) can be identified with equation (4.7) and for an arbitrary value η=α so

$$\mathbf{w}_i^c(t+1) = \mathbf{r}(\eta) = \mathbf{w}_i^c(t) + \eta\left(\mathbf{I}(t) - \mathbf{w}_i^c(t)\right) \in H_{t+1}^{(i)}.$$

Thus, $\mathbf{w}_i^c(t) \in H_t^{(i)} \Rightarrow \mathbf{w}_i^c(t+1) \in H_{t+1}^{(i)}$, and taking, $\mathbf{w}_i^c(1) = \mathbf{I}(1)$ by the principle of mathematical induction, $\mathbf{w}_i^c(\tau) \in H_\tau^{(i)}, \forall \tau$ ■

4.3 EUCART-BSA Hybrid

Section 4.3 presents simulation results showing the application of EUCART to the solution of a non-linear control problem (the cart-pole problem) using reinforcement learning. First, EUCART is applied in the form discussed in section 4.2. Second, a modified version of EUCART, EUCART with nearest neighbour priming, is proposed in an attempt to improve performance further. The results are compared to those of the original BSA implementation and some properties of the EUCART-based neurocontroller are discussed.

4.3.1 Hybrid Description

This section describes the simulations carried out using the EUCART state-space decoder in place of the fixed state-space decoder of Barto *et al.* (1983). The first objective is to confirm that the idea of decoupling the state-space representation task and the control action learning task is tenable. If so, the original BSA reinforcement learning implementation can be retained and different neurocontroller architectures could be developed through modifications to the decoder. The second objective is to develop a decoder which does not require *a priori* state-space structuring and can organise state-space information autonomously through experience. The achievement of the second objective is a sufficient condition for achievement of the first and indicates the possibility of other decoder architectures.

The EUCART-BSA hybrid consists of the EUCART decoder with nodes linked to stored ASE / ACE information. Each node now takes the place of a single box which can determine its own extent, and nodes may be added as required.

Using EUCART as a self-organising decoder coupled with the BSA implementation of reinforcement learning, gives a neurocontroller with the following benefits:

- minimal supervision requirements
- incremental partitioning of state-space,
- "quick" partitioning on-line,
- category overlap: basis for distribution of control information
- "once a member always a member" anticipation of a distributed representation ensuring that inputs always contribute to their allotted category.

4.3.2 Simulation 1: the Basic Hybrid System

Simulations, following the method of Barto *et al* (1983) comprising 10 runs of 600 trials each, were carried out. As in the BSA implementation, the state vector was reset to $x = \dot{x} = \theta = \dot{\theta} = 0$ after each trial. The simulation conditions and parameters were similar to those in the BSA implementation except for a few minor changes necessitated by the new approach. First, runs were not terminated when the trial of a particular run first reached the ceiling of 500,000 time steps of 0.02 seconds (approximately 2.8 hours of simulated time). Learning was still occurring in some cases and the system had to reach the ceiling value a large number of times consecutively to indicate convergence. Second, the learning parameter, a was set to 1,000 in the BSA implementation to establish control actions quickly. In the present implementation, because the state-space partitioning is not fixed, learning needs to remain plastic to prevent premature establishment of control actions. Hence a was set to 0.8. The parameters used in the EUCART decoder were α=0.00001, β=0.9 and ρ=0.8. The FCSR option was used.

Table 4.1 shows the results of the first 10 runs of the EUCART-BSA hybrid system.

seed	1	2	3	4	5	6	7	8	9	10
trials	9	1550+	108	271	404	223	746+	287	455	24
nodes	24	533	188	429	472	334	573	355	454	62

Table 4.1 The results for the first 10 runs of the EUCART-BSA hybrid system. The number of trials to convergence and the number of nodes generated are shown Note that learning times are much longer than for the BSA original system. This is because a state-space partitioning has to be learned.

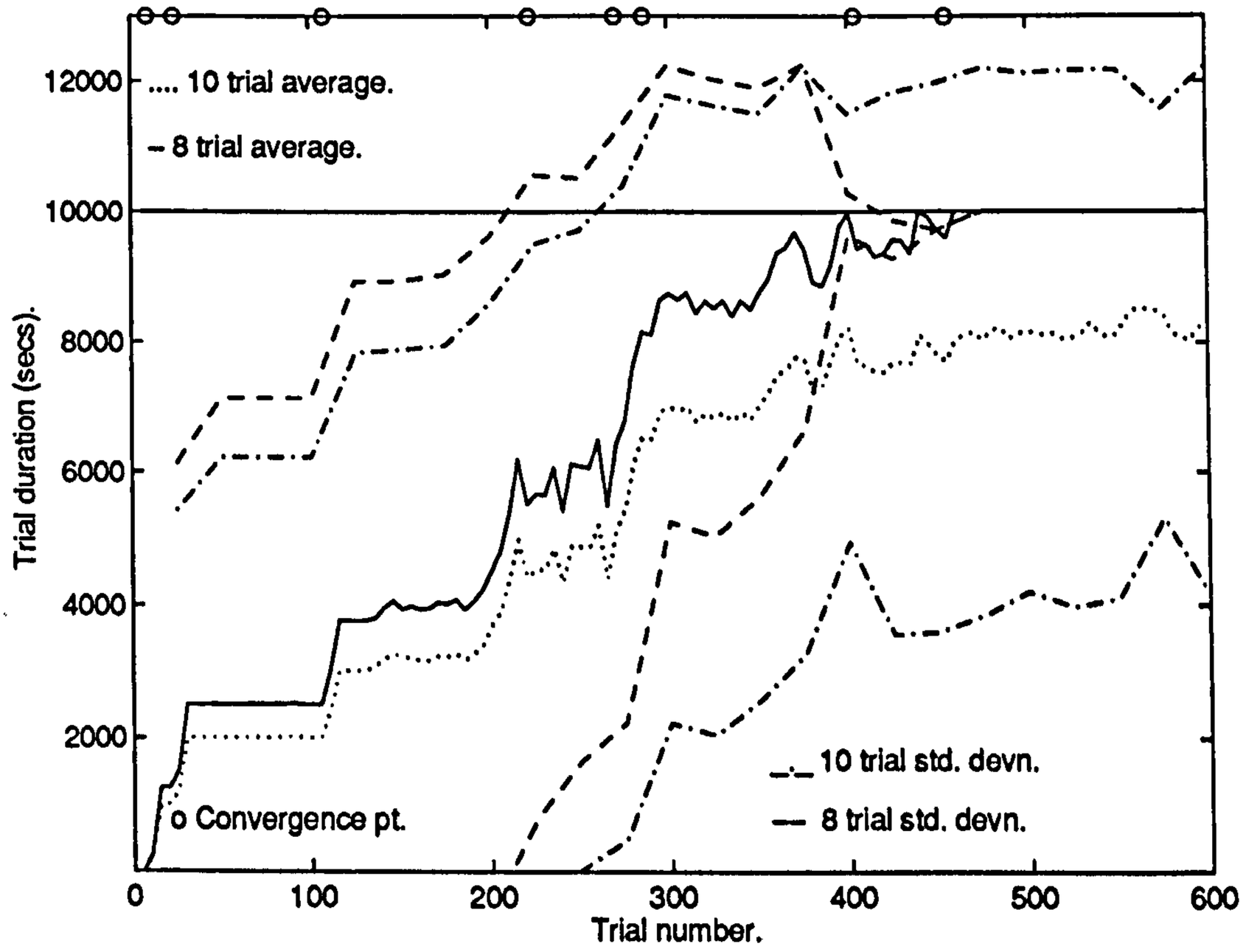


Figure 4.9. Simulation results showing the performance of the ASE / ACE system with the EUCART state-space decoder. The trials were averaged over ten or eight runs as described in the text.

Figure 4.9 shows the results of 10 runs and a subset of 8 runs. The Simulated trial duration is plotted against the trial number to show how control performance changes with increasing experience. The subset was required for clarity as 8 of the 10 runs converged to the ceiling value of 10,000 seconds (500,000 time steps) within the 600 trial limit; The remaining two runs converged at about 1500 trials and 1200 trials respectively. The solid curve shows the average of the 8 runs which converged during the trial limit. The dotted curve shows the average with the remaining two runs added to the ensembles for each trial. As with the BSA

study, a single point is plotted to indicate the average of each bin of 5 consecutive trial (ensemble) averages. The remaining curves show 1 standard deviation either side of the respective means, i.e. the dashed curve is associated with the solid curve and the chained curve is associated with the dotted curve. These are calculated at 25 trial intervals on the original ensemble values (not on the five trial bins). Although the sample size is small, standard deviation is used to indicate spread, since maximum and minimum values are dominated by the trial which converges first. The circles at the top of the graph indicate at which trial the members of the 8 run subset converged.

In the original BSA implementation the simulation results show that convergence towards a solution of the cart-pole problem occurs mostly within 100 trials. The present implementation requires more trials than this on the whole, but does eventually solve the problem. Here, learning is incremental, and is required to be more plastic; consequently, learning is slower to allow for adjustments in the state-space representation. With rapid learning of control actions, changes in state-space representation for a particular node centre and its immediate vicinity would not be followed by concomitant changes in the control actions to the required extent. Thus, the control action would not be representative of the modified state node and its current sphere of influence; it would represent, instead, the established control based upon a premature partitioning of state-space.

The utility of EUCART lies in the generality of the resulting approach when coupled with reinforcement learning. No *a priori* partitioning of state-space is required unlike "boxes" (Mitchie and Chambers, 1968) or the original BSA implementation (Barto *et al.* 1983). In principle, the new approach could be applied to other dynamical systems with little modification without the requirement for an alternative fixed state-space partition specific to the new system. This indicates the possibility of general purpose autonomous neurocontrollers.

Note the wide variation of average trial duration for each ensemble of 8 or 10 trials. The stochastic nature of the control output for a new node results in widely varying state-space trajectories while the control mapping is being established during each run. Within a single ensemble of trials, one run may have established a control mapping very quickly within a limited region of state-space while another may still have low trial durations as a result of initial control outputs pushing the state-space trajectory further away from a desired region and causing the creation of many naïve nodes requiring training.

Figure 4.10 shows the average increase in the number of EUCART nodes for both the full set of 10 runs and for the 8 trial subset. Both the 8 run averages and the 8 run maxima reflect convergence to a final set of desirable control actions. The 10 run averages and 10 run maxima indicate that adequate state-space coverage has not yet been achieved for the remaining two runs; adequate coverage in the present context means that a control mapping has been established which maintains control for a cart-pole system starting with a given set of initial conditions. Whether coverage is adequate given a different set of initial conditions is another matter.

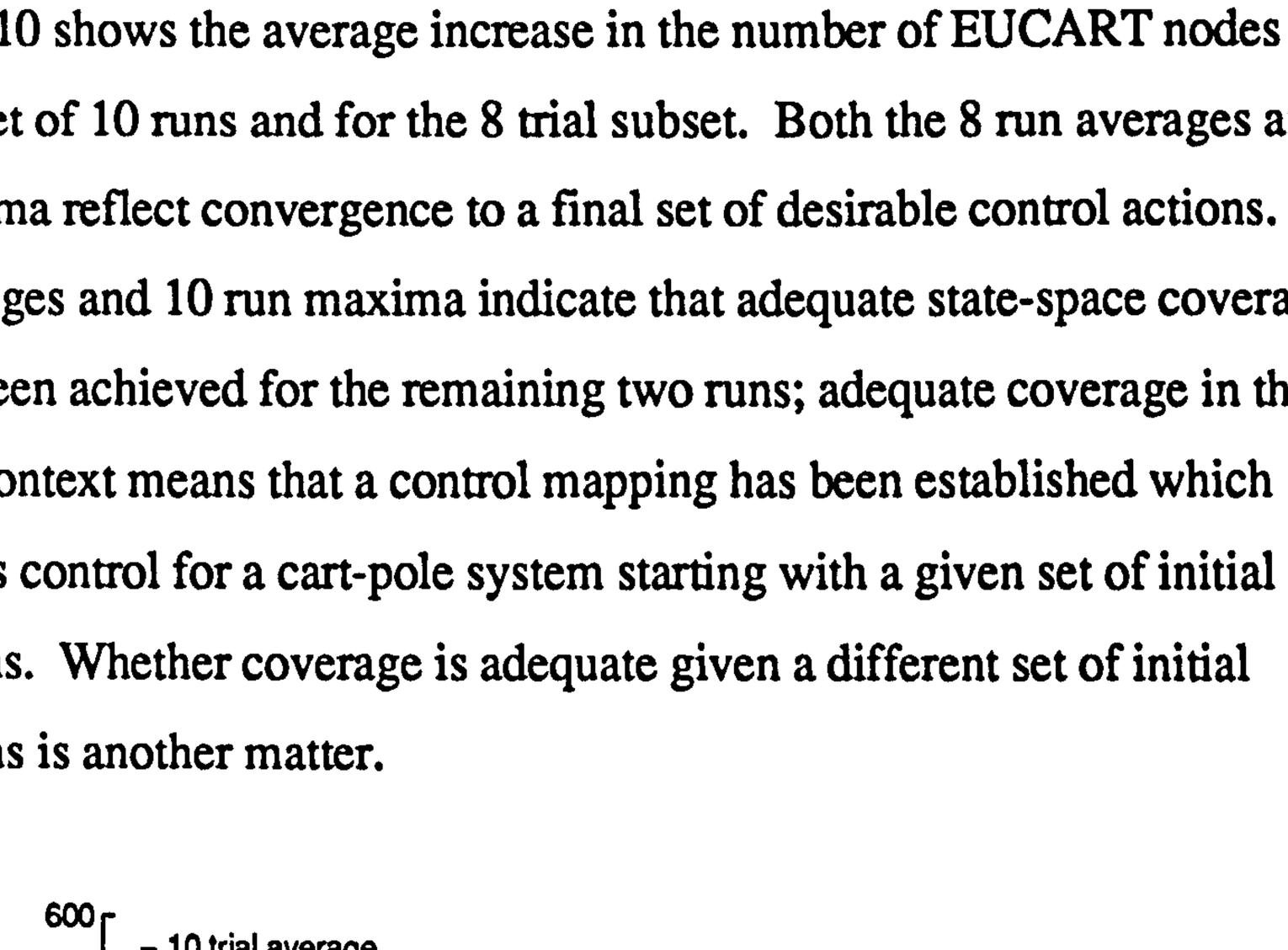


Figure 4.10 Simulation results showing the increase in the number of EUCART nodes representing individual state-space regions and their associated control actions.

As expected for the neurocontroller performance, the trend is towards greater trial durations as the trial number increases. However, the increases in trial duration are not monotonic. This is because the addition of a new EUCART node introduces an initial arbitrary control action. This sometimes pushes the state-space trajectory into previously unencountered regions of state-space or a region where the control actions are not properly established. The neurocontroller is then likely to fail if the well-established state-space regions are not re-entered quickly. Also, new nodes are sometimes added to cover "gaps" in state-space and their influence replaces some well-established state-space regions with naïve coverage because the regions are now associated with a new node (i.e. the new node centre is now nearer to states previously encompassed by other nodes). For winner-takes-all competition, the EUCART category composition theorem is not violated because the input vectors are associated with the new category nodes. However, information is lost from long-standing categories which are now "further away" because only a winning node is chosen and previous associations are discarded. A distributed representation of data would prevent this by taking into account all categories with which the input is associated.

Figure 4.11 illustrates the situation schematically. The dark border shows the decision boundary within which states belong to the new node. The degree of disruption caused by a new (and naïve) node depends upon the extent of overlap. The extent of overlap, in turn, is a function of the Euclidean distance between the category centres. If this distance, for a particular node with respect to its nearest neighbour, exceeds twice the maximum possible category radius, then no overlap will occur until a new node is added which violates the minimum distance condition. The dependence of overlap on inter-category distance is exploited in the modified EUCART decoder implementation discussed later.

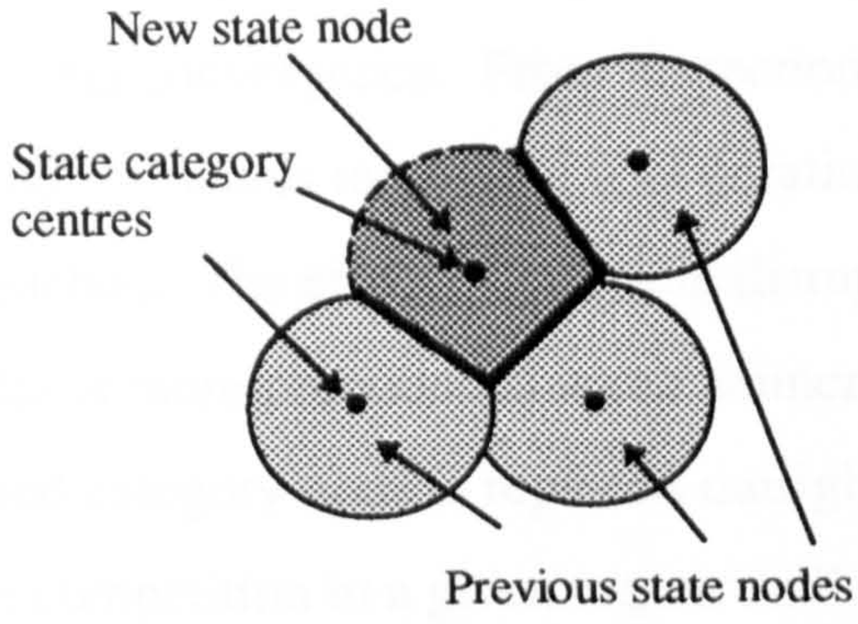


Figure 4.11. A schematic illustration of the problem of overlap associated with adding new nodes. The decision region for the new node now includes space previously covered by the original EUCART nodes. Disruption to established control actions within these regions is possible until the new node has learnt to represent a desirable control action.

As the results show, performance is eventually recovered when the new nodes learn to represent desirable control actions. The undesirable disruption is reminiscent of the *stability-plasticity dilemma* (Carpenter and Grossberg, 1987a) which states that adaptive systems must balance the requirement for stable learning of information against the requirement of plasticity and adaptation to novel phenomena.

With fixed non-overlapping box-based decoders, the neurocontroller is a pre-established look-up table which is filled during learning. Although there is no overlapping and hence no disruption of learning between state-space regions, *a priori* assumptions are made by an operator which restrict the autonomy of a learning system; such assumptions include a prespecified state-space granularity and a prespecified distribution of state-space categories. The disruption effect is a consequence of the autonomy of a EUCART-based neurocontroller; attempts to improve neurocontroller performance must include a reduction of this disruption effect without reducing the level of autonomy.

Figure 4.12 shows the results of a typical run. Again, the results are plotted as an average of bins of 5 consecutive trial values. The graph indicates some correlation between increases in node numbers and disruption of trial duration. This is readily apparent at around trial 400 with the small increase in the number

of EUCART nodes occurring simultaneously with a drop in the trial duration before recovery and final convergence. From inspection of Figure 4.12, it is apparent that the trend is towards increasing trial durations until the ceiling of 10,000 seconds is reached. The effect of transient disruptions caused by the addition of new nodes is more pronounced when winner-takes-all dynamics are used because a trained category node is replaced outright by a naïve node which, henceforth, wins the competition in a given region until, possibly, replaced by a new node. Over time, this node will be trained and will reflect the control mapping correctly within a given region of state-space.

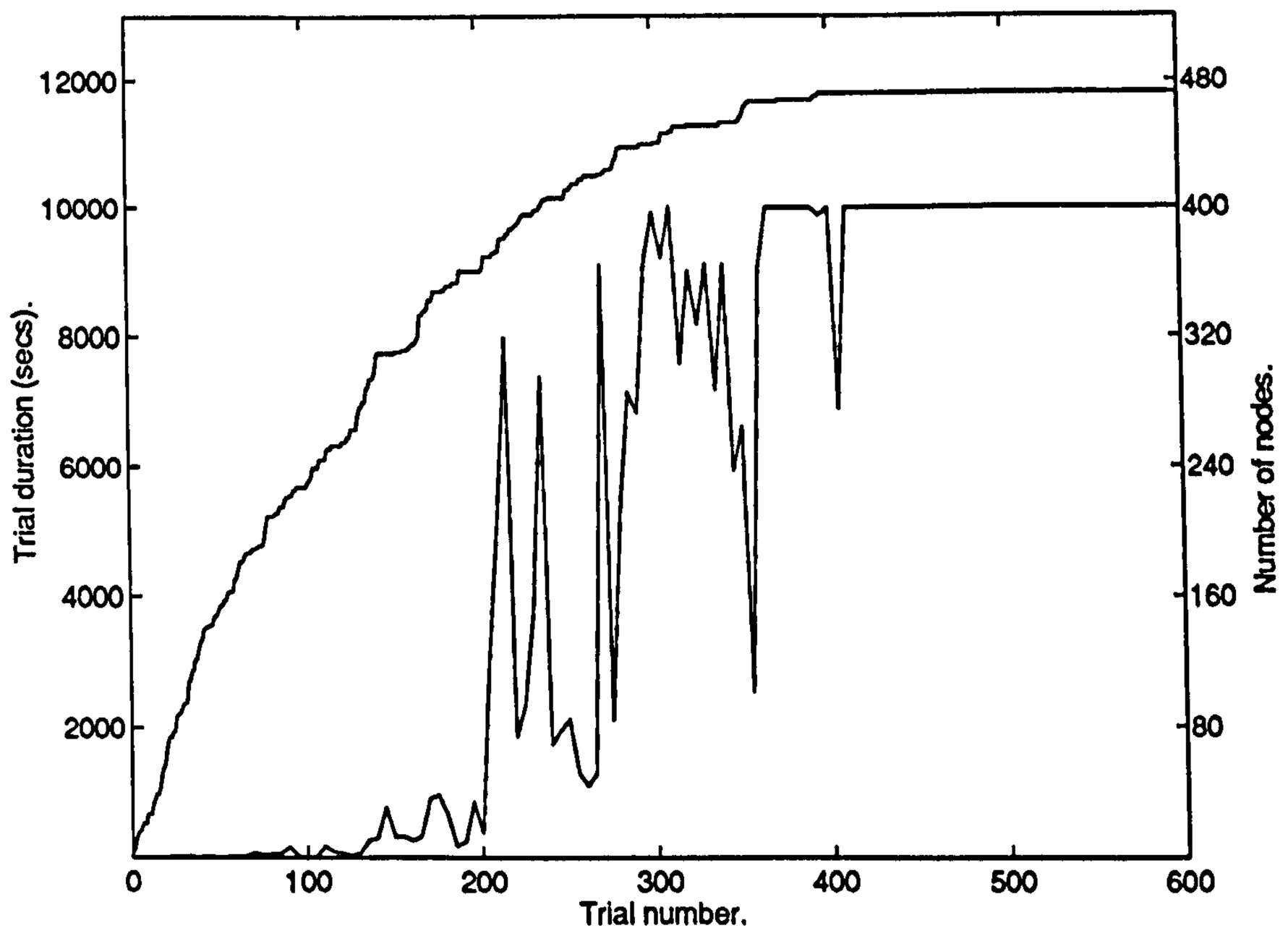


Figure 4.12. One typical run from the ensemble. Results are plotted as averages of five consecutive trials. Note the transient disruptions caused by the addition of new nodes.

4.3.3 EUCART, Incremental Clustering and Stability

The last point in section 4.3.2. raises the question of stability. The incremental clustering algorithm of EUCART gradually builds a *cover* over regions of state-space; whilst the cover is being built, transient disruptions will occur. When no "gaps" exist in a region of state-space, disruptive naïve nodes will no longer be

required and a tessellation of this region by the choice function hyperplanes, between neighbouring category centres, will have formed.

Moore (1989) proposes two types of stability for incremental clustering algorithms:

- *Stable* 1: no prototype vector can "cycle", or take on a value that it had at a previous time (provided it has changed in the meantime), and

- *Stable 2*: only a finite number of clusters is formed with infinite presentation of the data.

Moore modifies the condition of Stable 1 to include the case where a prototype vector may include a previous value but it must eventually stop moving. The condition of Stable 2 is also restated as:

"*in a bounded input space, condition (2) is equivalent to requiring that prototype vectors do not get arbitrarily close to each other.*"

EUCART is Stable 1, in the modified sense. For a given category, the category centre will stop moving when the EUCART category reaches its maximum extent; the category centre may pass through a previous value but will converge towards its final position in the fully extended category. Analogously to fuzzy ART, (Carpenter Grossberg and Rosen. 1991) $\|\mathbf{u}_j^E\|$ monotonically decreases and $\|\mathbf{v}_j^E\|$ monotonically increases until the category reaches its maximum diameter; at this point the category has stabilised. EUCART is also Stable 2 because the input space is bounded (because the dynamics of this particular problem are constrained to lie on a manifold) and thus requires a finite number of hyperspheres to contain it. Category hyperspheres may extend beyond the input space but, where they do, no input vectors will be found there by definition; this "fictitious" input space allows a complete cover of the Euclidean input space by hyperspheres of a fixed radius and thereby obviates the requirement of collections of hyperspheres near the input space boundary with radii tending to zero. The shortest run of the set of

runs converges after just 10 trials with only 24 nodes. This set of control actions is almost certainly limited because comparatively little of state-space has been explored. The controller would not be expected to be as robust and to possess as good disturbance rejection properties as those controllers with many more nodes which indicates a wider experience of state-space. From the point of view of robustness, the random perturbations caused by the introduction of naïve nodes have a beneficial effect on the long term experience of the neurocontroller by extending its experience into new state-space regions.

In many control applications these random perturbations by the naïve nodes may not be desirable or practical when working within a real environment (it may be dangerous) but including a model of the environment alongside a neurocontroller may allow "what if" probing by the neurocontroller to improve the rate of convergence towards a viable control solution. A better solution perhaps, would be to set failure limits for the reinforcement learning neurocontroller which lay within regions of performance recovery by an operator, or other control method, so that learning from failure did not necessarily entail disastrous consequences within a real operating environment. Failure would then represent undesirable system states to be avoided by a neurocontroller and which would lead to an operator warning to allow manual recovery of performance.

The naïvety of neurocontrollers with comparatively few nodes is considered in Chapter 5 which illustrates the adaptiveness of the EUCART approach; there, it is shown that when new regions of state-space are encountered, a EUCART-based neurocontroller is able to adapt without catastrophic forgetting (Sharkey and Sharkey, 1994). The next section will present a modified EUCART-based neurocontroller which attempts to reduce the disruption by naive nodes during incremental clustering.

4.3.4 Simulation 2: Nearest Neighbour Priming

As discussed previously, it was found that näive nodes, added to fill "gaps" in state-space coverage, often disrupted currently established control information. Although recovery and convergence eventually occurs, it would be desirable to minimise disruption during learning. Where disruption is likely to be the most severe, it is because the region of influence of a naïve node infiltrates established nodes in the neighbourhood of the new node. Thus, some state vectors which were previously encompassed by the original nodes are now nearer to the new node centre and thus elicit the control action determined by the new node parameters; the parameters have not yet had time to tend towards desirable values because the node has been newly created.

To reduce disruption when a new node is added, information from surrounding nodes must be taken into account. Instead of beginning with a zero initial control action weight, the weight values of n nearest neighbours can be combined to give an initial weight value. In the present modified implementation, a scalar weighted average of the form

$$z_0^{new} = \frac{1}{n}\sum_{i=1}^{n}\eta_i z_i$$

is assigned as the ASE weight for the new node, where η_i is the scalar *contribution weighting* of the i^{th} neighbouring state-space category and z_i is the i^{th} ASE weight. The contribution weighting takes the following two factors into account,

- *category centre distance*; the further away the neighbouring state-space category is from the new category centre, the smaller the contribution to the initial ASE weight should be, and

- *category node age*; the "older" the neighbouring category in terms of learning experience, the more established the control action is and hence is less likely

to be disruptive; the contribution to the initial ASE weight should be reduced for recent (naïve) categories.

Ideally, new node priming is determined by close, well established categories with desirable control actions. The form of the contribution weighting for the i^{th} nearest neighbour is, $\eta_i = \left(T_i(\mathbf{c}_{new})\bar{x}_i\right)^p$ where $\mathbf{c}_{new}$ is the newly created category centre, $T_i(.)$ is the EUCART choice function for the i^{th} category node, $\bar{x}_i$ is the ACE trace for the i^{th} node and $p \geq 1$ is used for contrast enhancement (Nabet and Pinter, 1991). Note that $T_i(.) \leq 1$ and $\bar{x}_i \leq 1$ imply that $\eta_i \leq 1$. Contrast enhancement is used to weight more heavily those nodes which are nearest to the new node or are "older".

The parameters used in the runs of the modified EUCART system were the same as those used in the runs of the unmodified version. The number of nearest neighbours, used to determine the initial ASE weights of new nodes, is n=5 and the power p=5 is used to contrast enhance the contribution weighting, η_i.

The K nearest neighbours technique coupled with a variant of ART was used by Zhang and Grant (1992) in conjunction with the boxes learning algorithm (Michie and Chambers, 1968a). For a new input vector, if the input does not exceed a membership threshold, the K nearest neighbours to the input are selected and updated according to the degree of membership of the input with respect to the category nodes. Category centres are updated in proportion to input vector membership using a modified competitive learning scheme and represent the centre-of-gravity for a cluster of input patterns in state-space encompassed by the category node. The K nearest neighbour method is used in this context to update *existing* nodes; if no nodes fulfil the membership criterion, then a new node is created. Here, K nearest neighbours are used to prime the new node to minimise disruption during the learning process.

seed	1	2	3	4	5	6	7	8	9	10
trials	11	326	173	216	242	442	555	113	121	52
nodes	40	430	246	349	377	398	396	211	275	149

Table 4.2 The first 10 runs of EUCART-RL with priming showing trial durations and final number of nodes.

Figure 4.13 shows the results of ten runs using the parameters of the previous set of runs. This time, all ten runs converged within 600 trials.

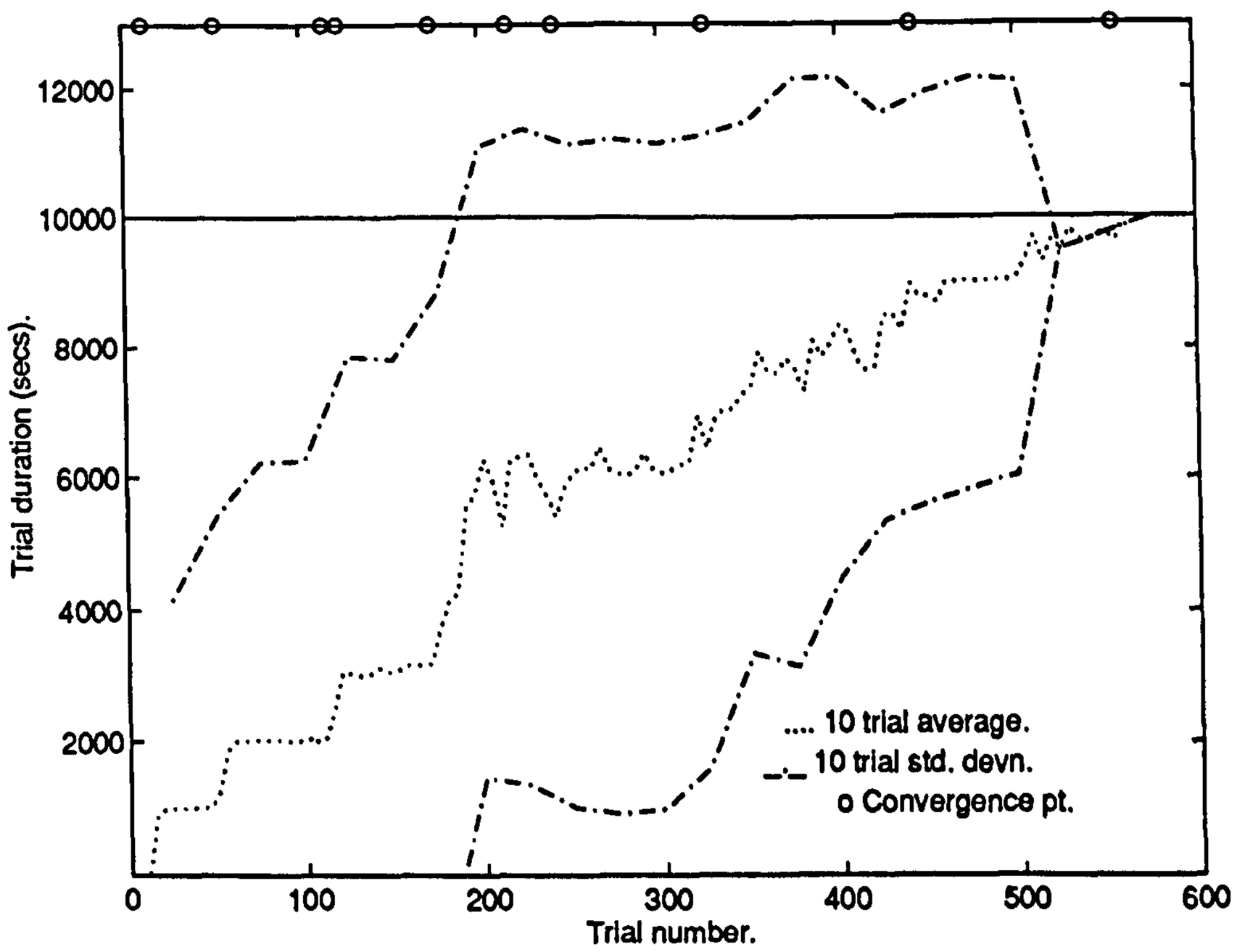


Figure 4.13. Simulation results showing the performance of the ASE / ACE system with the EUCART state-space decoder using nearest neighbour priming. The trials were averaged over ten runs before plotting in bins of five trials; all runs terminated within 600 trials.

After about 250 trials, the average number of nodes for the modified version of EUCART (Figure 4.14) is similar to that of the eight run average of the original version. This indicates that the increased average for the ten runs using the original version of EUCART is caused by the two runs which did not converge within the 600 trial limit. The time required-to-convergence and the number of nodes are linked by the fact that an increase in the number of naïve nodes requires an increase in learning time to modify the new parameters.

Priming is only effective in reducing disruption in regions of well-established nodes represent a desirable control mapping. Without priming, the effect of a new node is to issue a random control action which may cause the state-space trajectory to enter new or weakly established regions of state-space. Often, the result is that well established regions cannot be re-entered and failure occurs subsequently. Where state-space areas are not well established, priming has little or no effect because of the contrast enhancement of:

i) *distance effects*, where neighbouring nodes are relatively far apart in sparsely represented regions,

ii) *experience effects*, where weakly established nodes contribute little information to the new node.

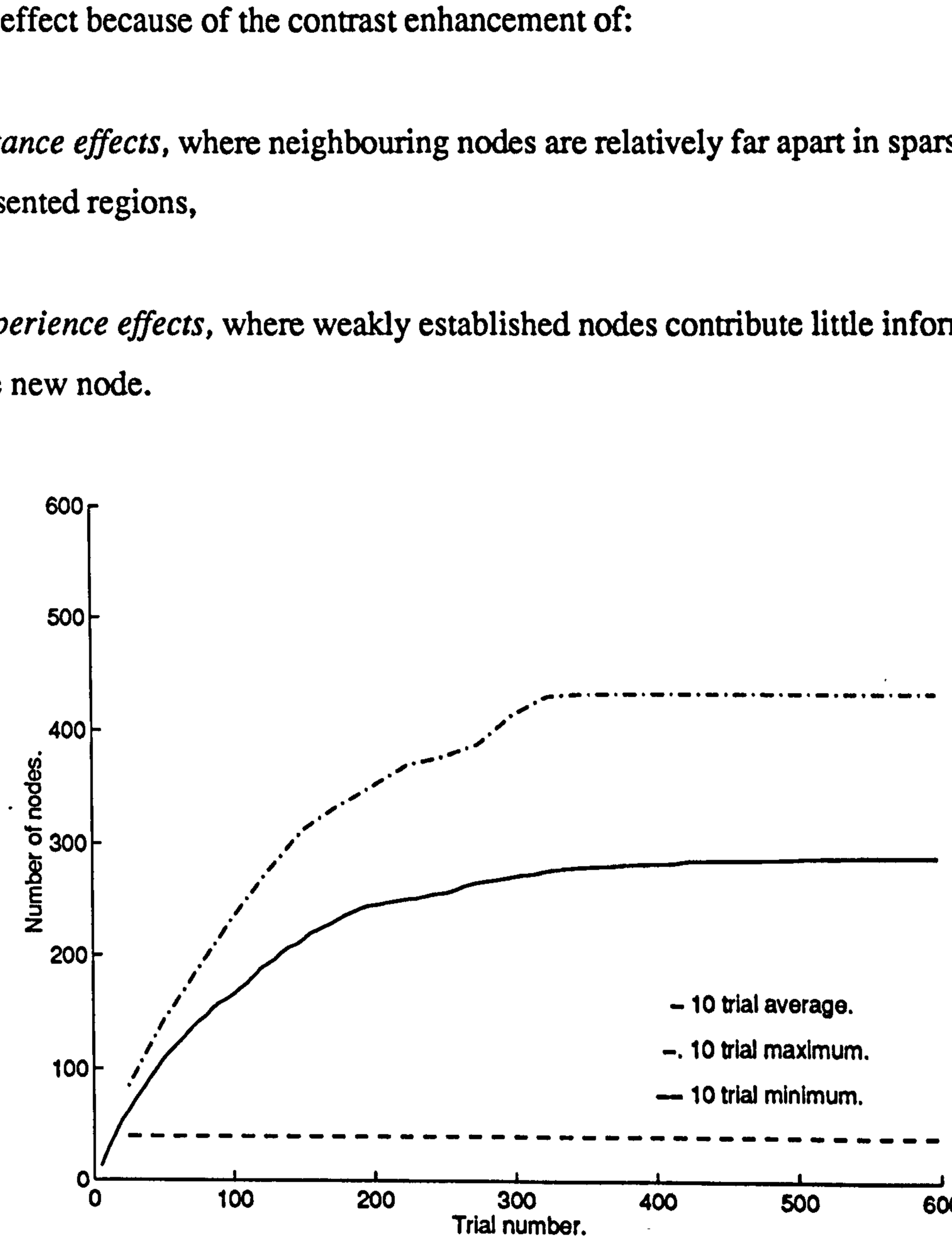


Figure 4.14 Simulation results showing the increase in the number of EUCART nodes when nearest neighbour priming is used.

Figure 4.15 shows a run using nearest neighbour priming which uses the same parameters as those used to produce the results of Figure 4.11.; the same random number seed was used to illustrate the difference in disruption effects. Comparing Figures 4.12 and 4.15 shows that the increase in the number of nodes is approximately the same until about trial 150 where the run using nearest neighbour priming begins to produce slightly fewer nodes and converges at around trial 250. The two peaks of Figure 4.15 that exceed 8000 seconds indicate that, although disruption occurs, the control mapping is becoming more effective. Without nearest neighbour priming in Figure 4.12, further disruption occurs for nearly 200 trials following the two peaks similar to those of Figure 4.15.

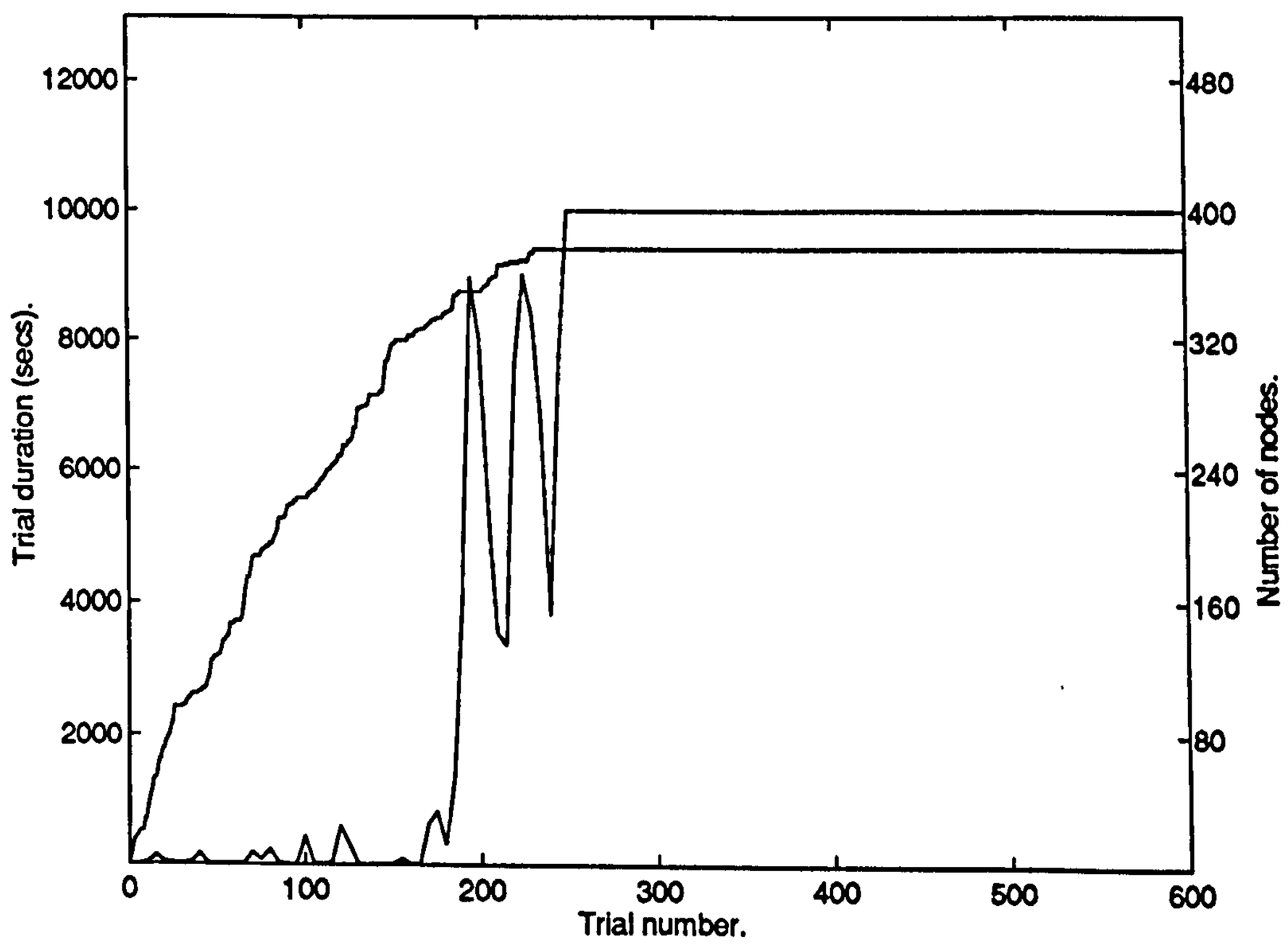


Figure 4.15. One typical run from the ensemble using nearest neighbour priming; the parameters are the same as those used in the run of Figure 4.12.

Figures 4.16 and 4.17 illustrate the effect of nearest neighbour priming upon one of the two runs which did not converge originally within the 600 trial limit. Figure 4.16 shows the original performance without priming and Figure 4.17 illustrates performance with priming.

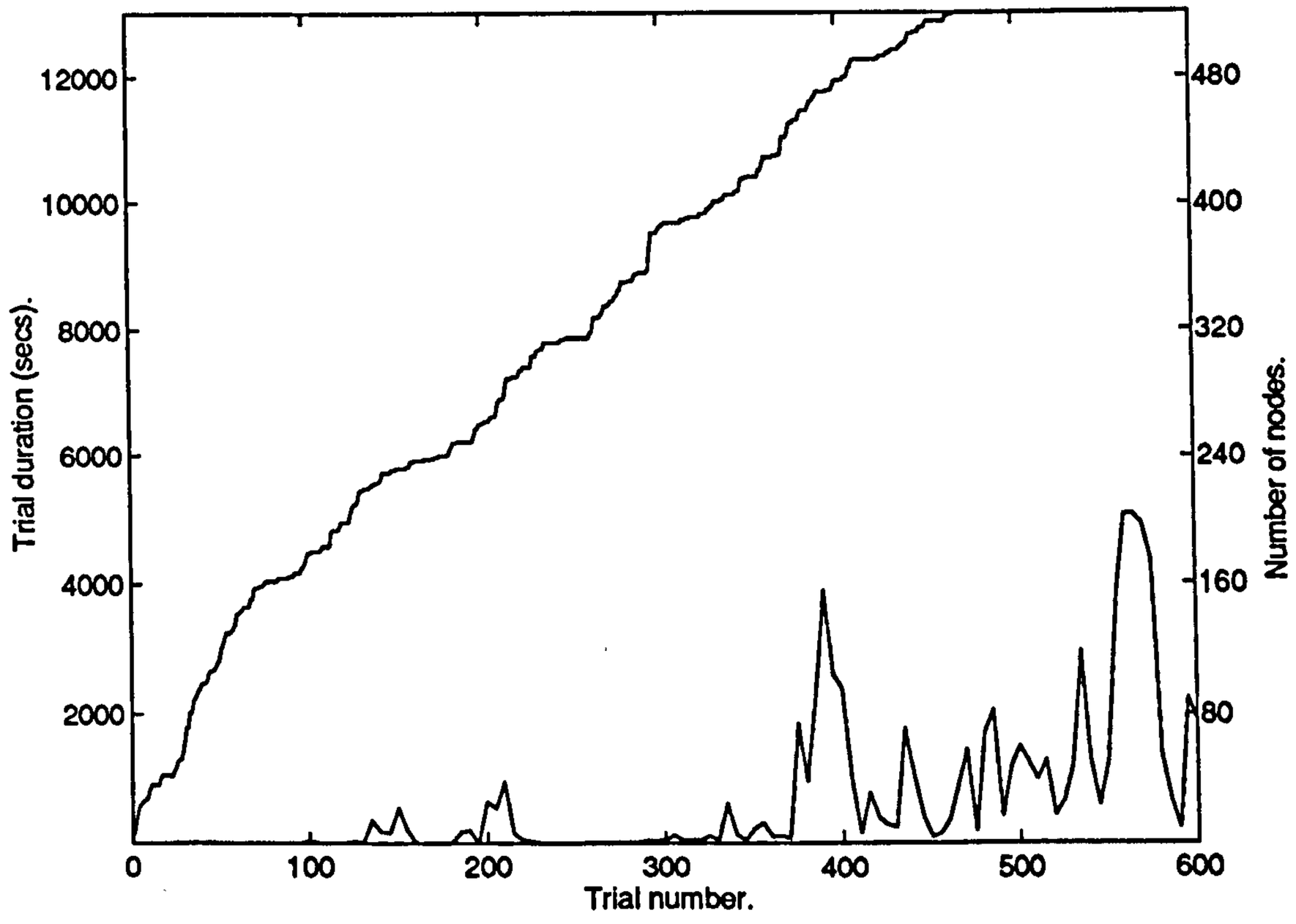


Figure 4.16. An example of a run without nearest neighbour priming which did not converge within 600 trials. Convergence occurred eventually after about 1200 trials. Note the rapid increase in nodes and the large variation in trial durations.

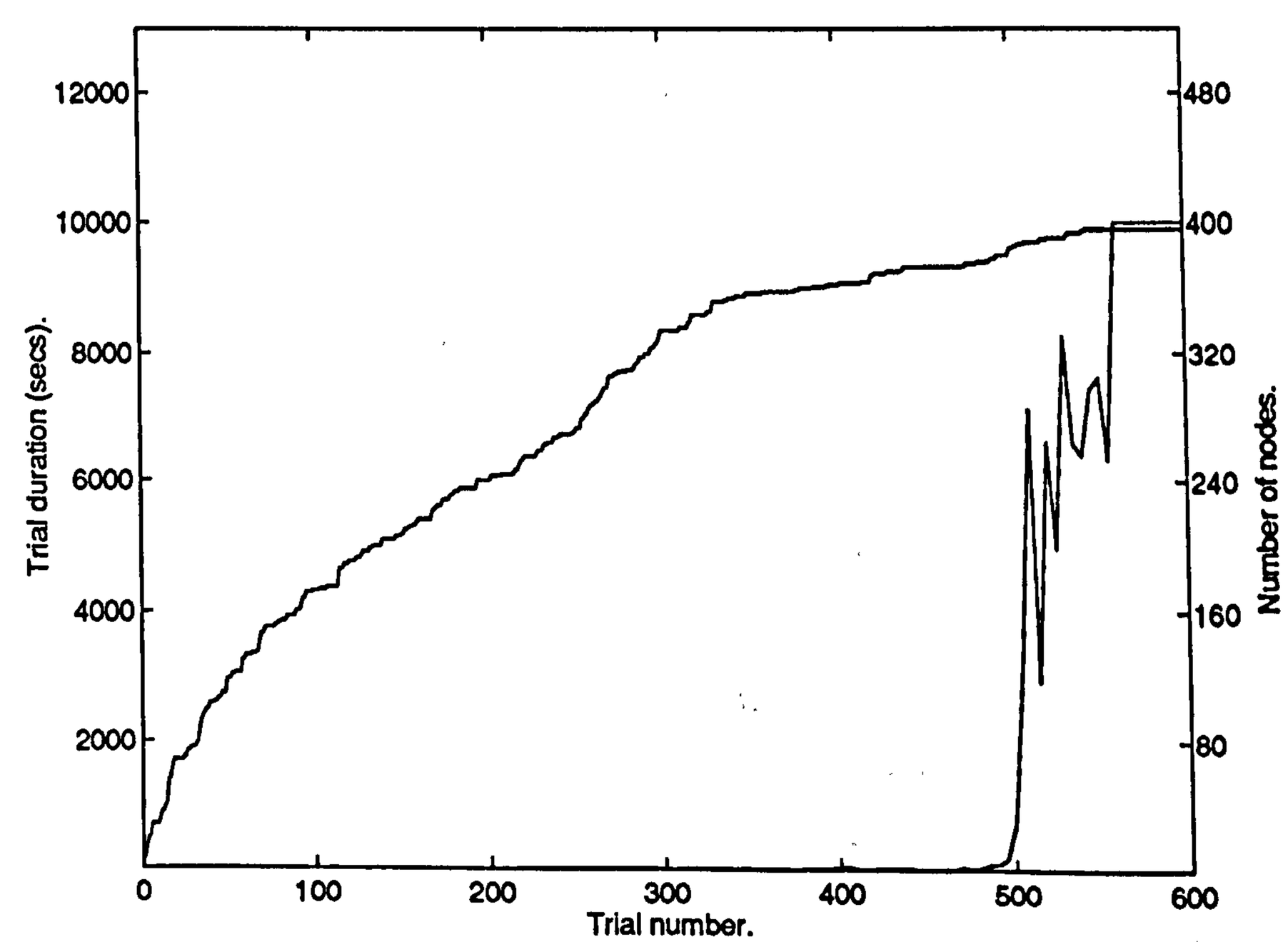


Figure 4.17. A run, using nearest neighbour priming, with the same parameters as those used in the run of Figure 3.26. Note the sizeable reduction in the number of nodes as compared with Figure 4.16.

4.3.5 Learning Rate

A set of 10 runs was carried out using the conditions of simulation 2 except that this time, the EUCART learning rate β=1. This is equivalent to fast learning (FCSR) in fuzzy ART. The results are shown in Table 4.3. It was decided to use the fast learning mode for EUCART for subsequent experiments because it made sense in the light of the comments on category membership which assumes that categories will always extend to contain any member.

seed	1	2	3	4	5	6	7	8	9	10
trials	39	108	431	521	368	177	408	113	591	87
nodes	106	231	434	436	409	308	391	222	502	170

Table 4.3 Fast learning in EUCART with the learning rate set to unity showing the number of trials to convergence and the final number of nodes for the first 10 random number seeds.

Comparing the results of Table 4.3 with those of Table 4.2 shows that setting β=1 has an effect, as expected. The simulation provides a baseline with which to compare future modifications using fully extended categories. Setting β=1 ensures that new inputs are always included in the category.

4.3.6 Simulation 4: Neurocontroller Adaptability with Different Initial Conditions

A question to ask is "what happens when different initial conditions are used in the cart-pole system simulation?". In other words, "how adaptive is a trained EUCART-based neurocontroller?". Any candidate neurocontroller must be plastic and must not suffer catastrophic forgetting when new information is encountered. Table 4.4 shows the results of a EUCART-based neurocontroller, using nearest neighbour priming, operated under different initial conditions following training; only the angle was changed in the simulations to illustrate the situation. With only 40 nodes, this controller has not explored much of state-space and has to reduce its naïvety through exploration of unknown regions.

New initial condition	No. of new trials required	No. of new nodes required
+1°	0	0
+3°	0	0
+6°	171	275
+11°	371	319

Table 4.4. Results from a naïve trained neurocontroller, consisting of 40 nodes, and the effect of changing the initial angle. After retraining, the angle was reset; no disruption of previous learning was observed.

The neurocontroller was trained with an initial state vector of zero. After training, the initial pole angle was changed before restarting the simulation. To check that learning under the new initial condition did not disrupt previous experience, the initial condition was then reset to the original value after successful training with the new condition. Resetting the initial condition did not disrupt the established control mapping in any of the cases i.e. the original mapping had not been "forgotten".

Initial conditions of +1° and +3° were dealt with easily by the 40 node neurocontroller and required no new nodes or further trials. For +6°, a further 275 nodes and 171 trials were required. Setting the angle to 11°, after resetting the neurocontroller, and training from zero initial conditions, resulted in a further 319 nodes (over and above the original 40 nodes) trained over 371 trials. The angle of 11° is near to the failure limit of 12°. These results indicate that a naive EUCART-based neurocontroller is able to adapt to the new conditions without disrupting previous learning. To illustrate the naïvety of the 40 node neurocontroller, a simulation was carried out using a trained 377 node neurocontroller with a new initial condition of +6° for the angle; for this neurocontroller, only a single further trial was required to train a single new node. The more extensive experience of the 377 node neurocontroller, compared with the 40 node neurocontroller of Table 4.4, is reflected in the reduced requirement for extra learning.

4.3.7 Simulation 5: Changing Plant Conditions and Robustness

some work has been carried out on the robustness of the current hybrid system which includes changing the operating characteristics of the simulation model (Marriott and Harrison, 1996). Preliminary results indicate (see Table 10) that the EUCART-based RL system is able to adapt to changes in simulation model characteristics (e.g. cart friction). The starting system at the beginning of each run is the naïve 40 node neurocontroller used in simulation 4. The first column of Table 3.10 indicates increase of the cart friction coefficient used in the original BSA study e.g. 650x signifies 650 times the original. Columns two and three indicate the number of extra trials required and extra nodes generated respectively before adaptation to the changed conditions. Learning following altered cart-pole conditions is usually accompanied by a "burst" of new node creation which represents a stochastic search of previously unseen state-space regions.

Frict. Inc.	Extra Trials	Extra Nodes
2X	0	0
64X	0	0
160X	0	0
240X	0	0
250X	946	484
320X	0	0
380X	0	0
384X	274	355
400X	0	0
450X	302	313
500X	162	239
600X	176	285
650X	265	360

Table 4.5 Changes in a EUCART-BSA neurocontroller required to recover control when cart friction is increased.

Both simulations 4 and 5 show that the EUCART-BSA hybrid is able to recover from changes in operating conditions which illustrates the adaptability of this autonomous neurocontroller. However, in both cases, this is at the cost of a considerable number of new learning trials and new nodes. Many of the new nodes will be spurious and can probably be removed without undue loss of control. Pruning of redundant nodes is covered in Chapter 5.

There are two anomalous results which have not yet been explained; they occur at 250X and 384X. The same random number sequence is used for all simulations but the number of extra nodes generated by the EUCART-BSA system indicates that new regions of state-space have been entered. The large number of extra trials required by the hybrid for the factor of 250 indicates that the neurocontroller had difficulty in maintaining control. It may be the case that for this particular non-linear problem, slight perturbations in any of the parameters can have disproportionate effects upon the outcome. Such anomalous behaviour may possibly be prevented by increasing the experience of the EUCART-BSA neurocontroller so that the state-space trajectory remains in a region of experience where a successful control strategy is more likely to be found. For a naïve neurocontroller, the region of successful control is likely to be small and control actions outside of this region are random thus exacerbating the control problem.

The fact that new learning is required is exacerbated by the initial naïvety of the neurocontroller. A neurocontroller which learns quickly has little chance of exploring state-space (Sammut and Crib, 1990). This is evident when using the 40 node neurocontroller which, like the others featured in this thesis, was started from the origin of state-space at the commencement of each trial. More experience of state-space in the initial stages of learning may reduce the requirement for future learning when conditions change. This is the exploration-exploitation trade-off once again.

4.4 Discussion

4.4.1 Problems

The EUCART-BSA hybrid is a potentially useful prototype neurocontroller. There is clearly much room for improvement of the current system but nevertheless it does provide an alternative approach to adaptive control. The achievement of increased neurocontroller autonomy (reduced designer intervention) is an ongoing process which can benefit from the combination of established neural network architectures in novel ways. The current drawbacks associated with the EUCART-BSA hybrid are:

- *transient disruption of established control* by the addition of new nodes;

- the *proliferation of uninformative nodes* caused by stochastic search of state-space during the early stages of establishing a control mapping;

- *naïve control* through lack of further experience of state-space over and above that required to balance the pole at the origin;

- the need for *arbitrary parameters* which have to be set by the user;

- *long learning times* when learning from failure, and

- *significant computational overheads* which increase linearly with the addition of new nodes.

4.4.2 Possible Solutions

Possible solutions and indicators of further work include:

- the use of *distributed control* to allow membership, of more than one state category enabling established nodes still to determine actions which would otherwise be determined by a naïve node;

- the use of "*relevance*" pruning to remove nodes created by state-space trajectories very rarely followed after control has been established—if operating conditions do change, new nodes can be created dynamically and will not be pruned if significant;

- the use of *self-tuning parameters* to adapt node size and position during learning. This is possibly the most difficult solution and will require "meta control" at a hierarchical level above that of the ACE element to ensure intelligent tuning based upon overall performance,

- *selective update* of significant trace values (Hu and Fellman, 1995) as discussed in this sub-section, and

- the training of the hybrid by starting the cart-pole system from *random state-space points* to allow the controller to experience more of state-space

All traces in the ASE and ACE sub-systems are updated each time. At any time instant, many of the trace values are insignificant and updating them is wasteful and computationally inefficient. Hu and Fellman (1995) propose a *state history queue* (SHQ) or set of registers which store a finite number of box addresses. Every time a state-space box is entered, its address is stored in the SHQ. States move through the queue which approximates the exponential decay of the traces. If a state which entered the queue is not accessed again during the time-length of the queue, it is discarded. As time progresses, box addresses are removed from the queue. Only state boxes currently in the queue are updated which removes

the need to access all box addresses each time the weights are updated. The SHQ method could be applied to the EUCART-BSA hybrid and the FUZBOX architecture of Chapter 5.

Hu and Fellman also put forward the idea of dynamic allocation of control memory, that is, control memory is only allocated for traversed state-space regions; this is used in both the EUCART-BSA hybrid and FUZBOX. The efficient use of storage and consequent reduction of computational overheads (more 'boxes' entails more computation) was less of a problem with the original BSA implementation; the set of boxes were optimised manually (Barto, Sutton and Anderson, 1983).

A larger coverage of state-space allows the adequate representation of possible system dynamics, but physical memory is only allocated to used states (Hu and Fellman, 1995). EUCART does this but a problem arises. There are a limited number of key regions of state-space which require coverage but many other regions are traversed, especially during exploration. Many of the state nodes created are seldom used, if at all, once a control strategy is established. The systematic removal of these spurious nodes through pruning would reduce computational overheads.

4.4.3 Short Conclusion

It has been shown that the EUCART state-space decoder, in conjunction with the ASE / ACE subsystems, is able to learn a control mapping for a non-linear control problem. The resulting neurocontroller is autonomous and does not require *a priori* information other than a choice of operating parameters. The incremental clustering algorithm of EUCART successfully partitions state-space and allows on-line adaptation to new regions which may not be accounted for by an *a priori* partitioning. The EUCART decoder simulations also extend the BSA implementation by considering the effect of new initial conditions on a trained neurocontroller that has converged to the simulation ceiling using the "all-zero"

initial state. Indeed, using a EUCART decoder has extended the generality of the BSA implementation of reinforcement learning by indicating the possibility of developing "general purpose" neurocontrollers; such controllers may not be as precise as those designed for specific tasks using high precision analysis and design techniques, but would be more readily applicable, "off-the-shelf", and ready to adapt through experience. This approach entails a movement away from highly accurate static mappings towards a more adaptive approach exemplified by the principle of increasing precision vs. decreasing intelligence (Saridis, 1989).

Isolating the state-space decoder task from the control action learning task and treating it as a "black box" allows the development of variant reinforcement learning networks which still retain the original ASE / ACE specification. The main operational criterion for a candidate state-space decoder used in this way is that it assigns a unique representation to distinct regions of state-space; the regions may overlap in places but the state-space representation and parameter updating methods must account for this. For example, winner-takes-all dynamics can be used to choose a winning neurocontroller node or parameters for several nodes can be updated in proportion to their respective activation levels (membership functions). The latter approach is consistent with fuzzy rule-bases where multiple rules may be activated. The EUCART-BSA approach uses the winner-takes-all method for choosing prediction and control information for consistency with the original ASE / ACE implementation which uses a fixed non-overlapping state-space partitioning. Although EUCART categories overlap, only one category is selected at any one time so potential conflict is avoided.

The EUCART self-organising state-space decoder discussed in this thesis has removed the need for such *a priori* restrictions but in doing so has introduced the problem of disrupted learning during incremental partitioning of state-space. This disruption is inevitable as the introduction of new nodes causes overlapping which changes the state-space tessellation and thus the established control mapping. Although the EUCART decoder system eventually stabilises, it is desirable to reduce transient effects during learning. The nearest neighbour modifications go some way towards reducing disturbances caused by the addition of new nodes but

a more distributed representation of state-space and the associated control mapping is desired while retaining the attractive properties of the ASE / ACE reinforcement learning system. The original BSA implementation does not preclude this. Indeed, the seminal paper of Barto *et al*, (1983) mentions this possibility.

The power law, for the nearest neighbour weighting using the fifth power, was chosen on the basis of empirical observation. Other forms of contrast enhancement law may be more suitable. The introduction of such parameters highlights one of the problems of self-organising systems; the danger is that by using self-organisation, other *a priori* assumptions are substituted for those assumptions that are to be removed. The requirement of numerous parameters can possibly reduce the utility of self-organisation over *a priori* structuring of information. On the point of *a priori* inclusion of information, Procyk and Mamdani (1979) state that

"it is impossible to design a controller which need not assume anything about its environment. One can only strive to lessen its dependency and sensitivity to it."

The minimisation of built-in assumptions about the environment must be a guiding principle in the development of neurocontrollers but with the proviso that, wherever possible and convenient, known facts can be included in the neurocontroller structure if performance will be improved by doing so. Having to learn known facts that could otherwise be built-into a neurocontroller to improve performance cannot always be justified by claims of autonomy.

Nearest neighbour methods can be used to compute both the control output and the predicted failure values for a given input vector by category membership value (Zhang and Grant, 1992) in conjunction with new node priming. This will probably reduce the disruption caused by the addition of new nodes and augment the limited applicability of new node priming by updating all nodes triggered by a state-space trajectory entering overlapping state-space regions. A method for distributed processing of predictions and control outputs within the ASE and

ACE processing units is required if internal representations of the state-space and control mapping are to be smoothed out.

Although nodes represent individual state-space regions and their associated control actions, the neurocontroller is not at all transparent to an operator. The nodes, in effect, represent "micro-rules" of the form 'if the state-space vector is in the region surrounding centre x then output y'. These numerical rules are not very meaningful and, in many cases, clusters of micro-rules could be replaced effectively by a more general rule. Pruning and generalisation of groups of micro-rules is possible but the associated technicalities may be obviated by using a more efficient state-space representation to begin with; for example, using nodes to represent fuzzy rules. Fuzzy systems are much better suited to knowledge extraction (e.g. Berenji and Khedkar, 1992; Jang, 1992,1993; Jang and Sun, 1995) than networks using micro-rules but introduce other considerations such as the choice between a rule base with a fixed number of rules or a self-organising rule-base; the task of rule extraction (Wang and Mendel, 1992) and the task of tuning the fuzzy membership functions.

The distribution of ASE / ACE dynamics is compatible with the fuzzy approach as it is possible that multiple rules are activated and contribute to the control or predictive outputs. Similarly, distribution of state-space decoding across multiple input lines may reduce the effect of state-space node overlap when EUCART is used.

In this thesis, it has been shown that the decoder section of the original BSA implementation provides a basis for the development of variant reinforcement learning architectures. The EUCART decoder is self-organising and is compatible with the original ASE /ACE formulation. Other types of state-space decoder that are similarly compatible are possible. The very fact that the principles of self-organisation and reinforcement learning can co-exist is an exciting prospect for artificial neural systems development and points a way forward to the development of autonomous neural systems that require much less outside intervention than at present.

Chapter 5 Extending the Hybrid

5.1 Meta-Control

Most neural network architectures are still relatively primitive when compared to even the simplest living systems. This fact cannot be attributed totally to the lack of available computing power because brute-force information processing would not solve the many complex problems which require more "intelligent" or heuristic methods. For example, human language and vision processing tasks would result in a combinatorial explosion if every possible combination of circumstances were coded for *a priori*.

A criticism of many current neural networks is that they are relatively *inflexible*. It is true that they learn, but this is only in a limited way. The majority of networks have a fixed structure and can only adjust themselves within a narrow band of possibilities. They also usually consist of a single structure, although modular structures are being developed (e.g. Jacobs and Jordan, 1993). Criticism of the supervised learning method has already been made.

The subject of animal learning has been covered briefly in section 3.1 onwards. Animal learning is purposeful and goal-orientated for the most part. Behaviour is internally generated and intermediate steps to a goal are developed by active exploration of an environment. Depending upon the level of evolution, animals become "aware" of obstacles in the way of reaching goals and avoid them by exploring alternative strategies There is much to be learned from studies of animal behaviour.

Neurocontrol is one area which requires more adaptive and autonomous systems endowed with a degree of intelligence. The term "*meta-control*" is introduced here to cover the concept of "controlling the controller"—with particular

reference to neurocontrollers. Although difficult to define, certain operational characteristics may be listed to help clarify the concept, viz.

- active, intentional or goal-driven behaviour as opposed to passive reaction to applied data;

- self-generated intermediate behavioural sequences,

- Higher-order (meta) evaluation of progress;

- 'intelligent' adjustment of control strategies;

- hierarchical and distributed systems composed of sub-modules;

- possible tracking of non-stationarity;

- forward planning and "what if...?" analysis.

- experiential modification of neurocontroller structure—c.f. *neural Darwinism* (Edelman, 1989) and genetic algorithms (e.g. Goldberg, 1989).

This list is not exhaustive but, it is hoped, conveys the idea of a genre of intelligent adaptive neurocontrollers which are capable of a greater degree of interaction with an environment in some ways similar to that of humans and animals. Any candidate intelligent neurocontroller will exhibit some of these properties to some degree.

5.2 Pruning

5.2.1 Introduction

One of the items of the list of section 5.1 is modification of the neurocontroller *structure*. The incremental addition of nodes has already been mentioned in the preceding chapters. Removal of nodes by pruning is also a viable approach to structural modification and has been investigated (e.g. Le Cun, Deenker and Solla, 1990; Reed, 1993, Fritzke, 1994). Removal of nodes from feedforward networks such as the MLP is more difficult than from ART-based networks because, in the former case, nodes contribute to a distributed representation and the amount of contribution to the mapping has to be computed globally (e.g Reed, 1993).

For the EUCART-BSA hybrid, many of the nodes that are created are relatively unimportant as they appear when the state-space trajectory moves between critical regions. One way to approach pruning is to remove nodes periodically if they appear to be of little relevance to control. Relevance may be assessed by calculating a measure the *relative trace strength* (RTS) given by

$$RTS_i = \frac{\bar{x}_i}{\sum_{k=1}^{n} \bar{x}_k}$$

The RTS has been introduced here to give a measure of *relative eligibility* which shows how much any particular node has been active. If the RTS for a node falls below a given threshold then the node may be removed because its relative importance has dropped in comparison with other nodes.

5.2.2 A Simulation

A simulation was carried out using the same conditions as for simulation 2 of section 4.3.1 with the random number seed set to 2. This time, pruning was carried out periodically with a period of 30 trials. The pruning threshold was extremely small with a value of 0.000001. This was to ensure that only very weakly active nodes were removed. The results are shown in Figure 5.1.

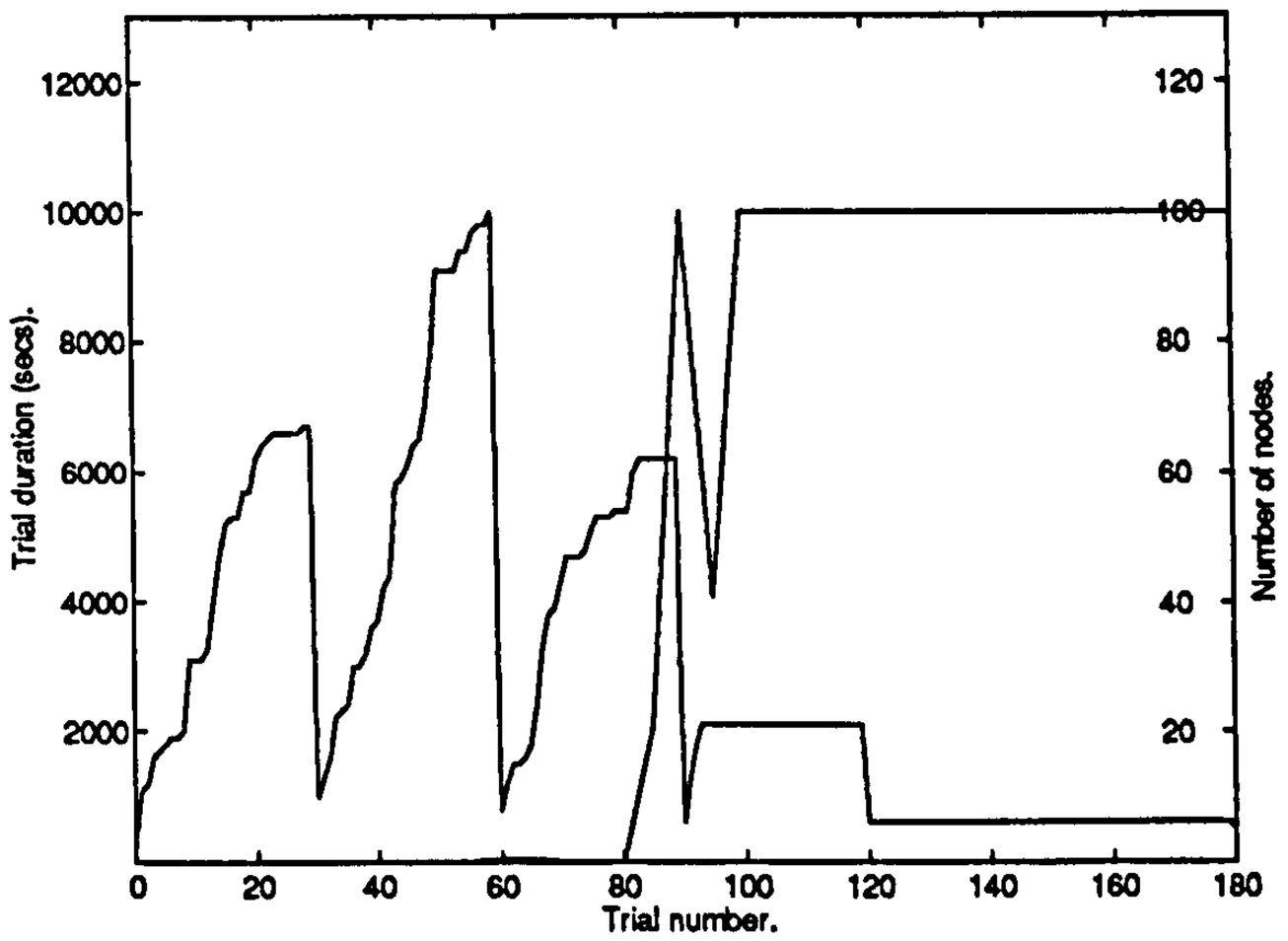


Figure 5.1 Simulation results showing the performance of the EUCART-BSA hybrid system with pruning. The pruning is periodic with a period of 30 trials. The resulting neurocontroller has only 6 nodes.

The run of Figure 5.1 converged at trial 84 with 62 nodes but pruning caused disruption until convergence occurred at trial 120 giving a 6 node neurocontroller. Figures 5.2 to 5.5 show the performance of the pruned system

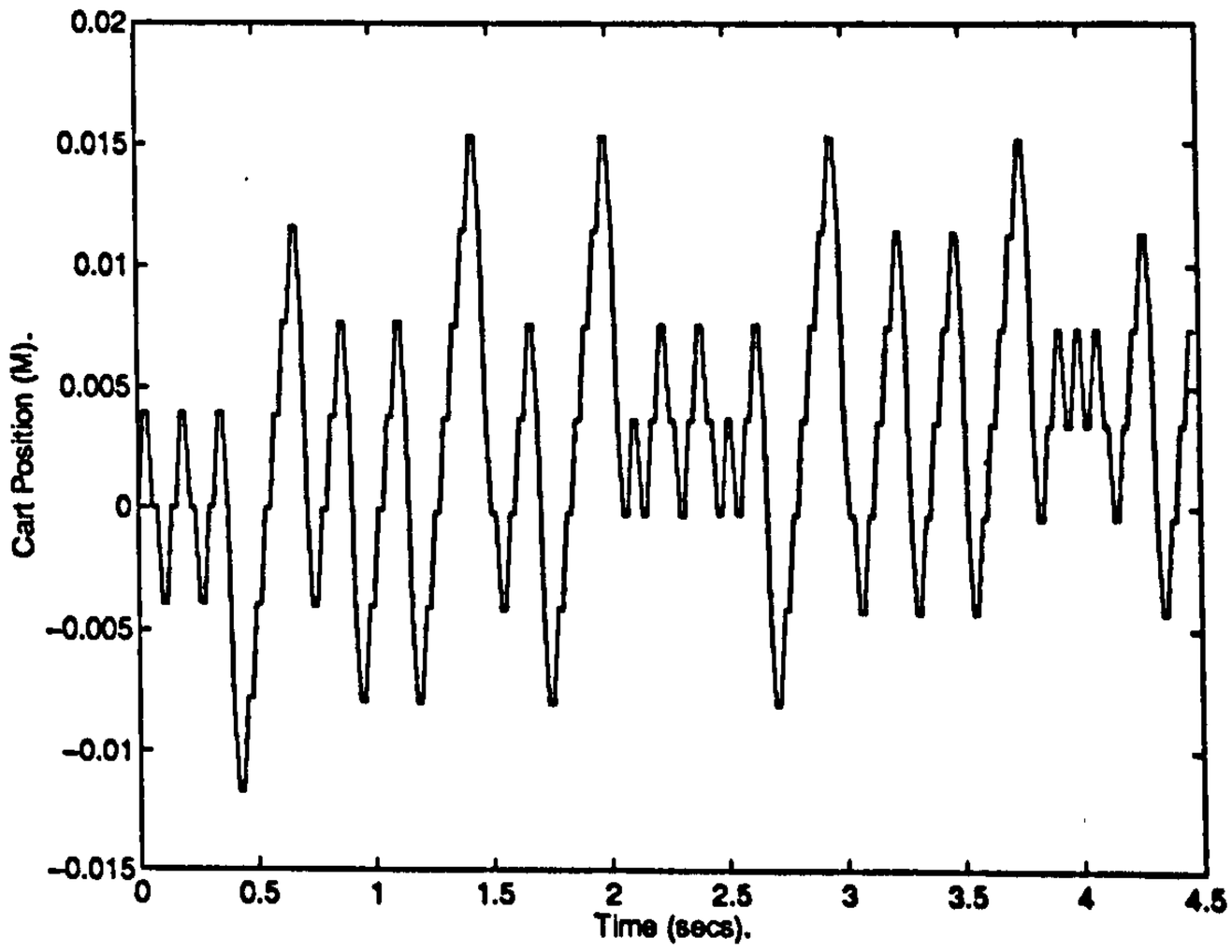


Figure 5.2. The cart position for the EUCART-BSA pruning simulation. Note the oscillation of the cart around the track origin; the cart is confined within about 0.015m of the origin or 0.63%.

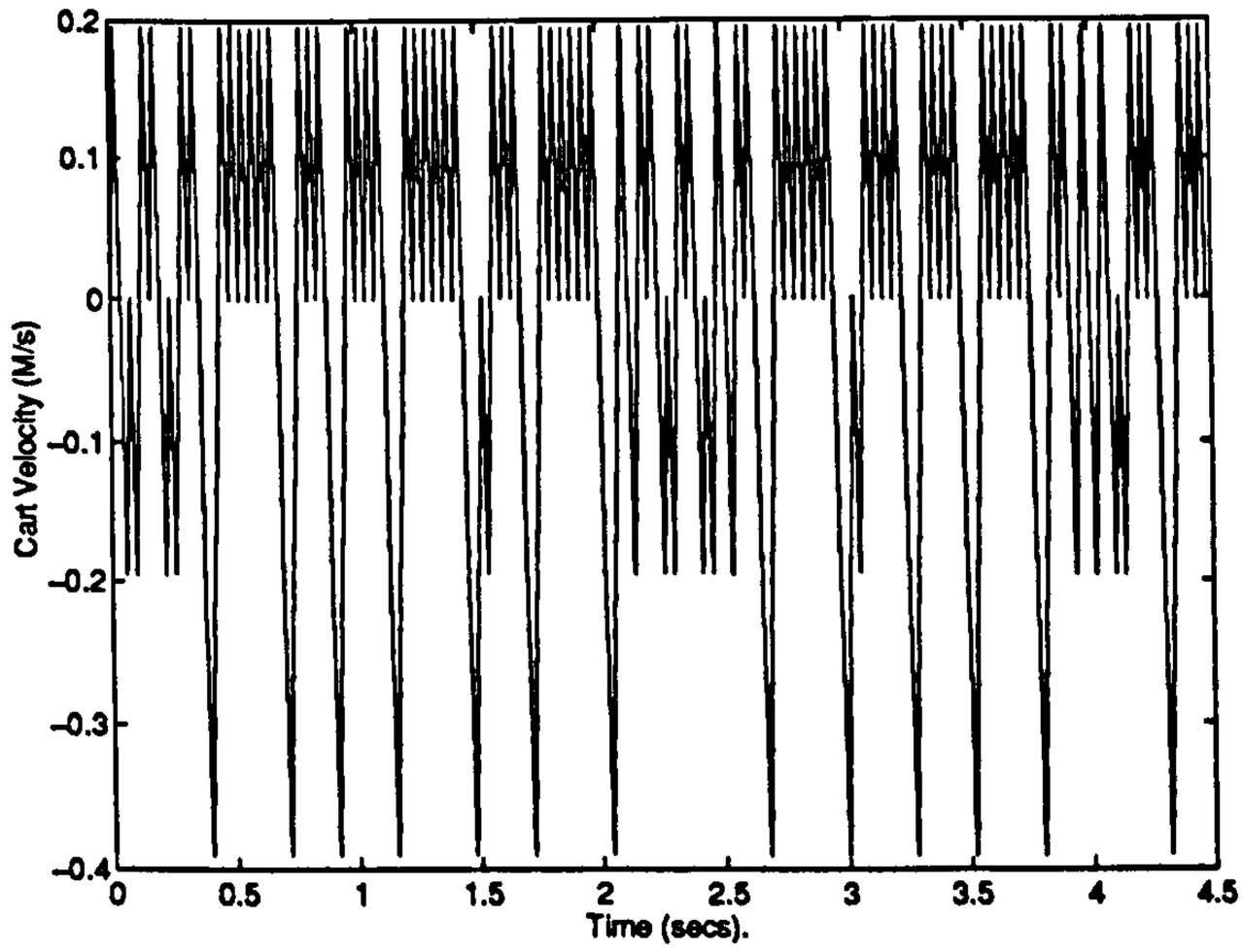


Figure 5.3. The cart velocity for the EUCART-BSA pruning simulation. Note the oscillations required to maintain a good cart position.

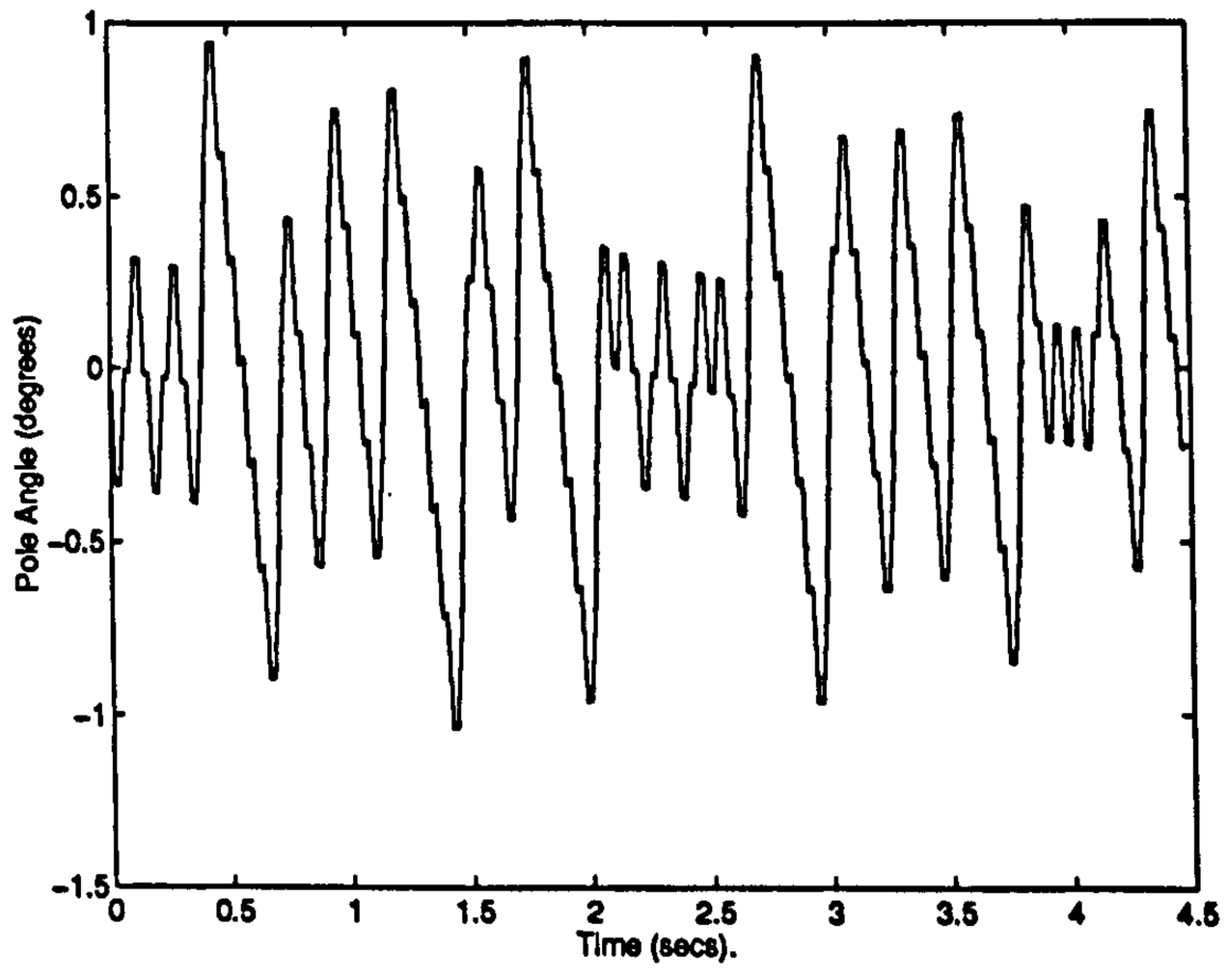


Figure 5.4. The pole angle for the EUCART-BSA pruning simulation Note that the pole remains within about 1 degree either side of vertical.

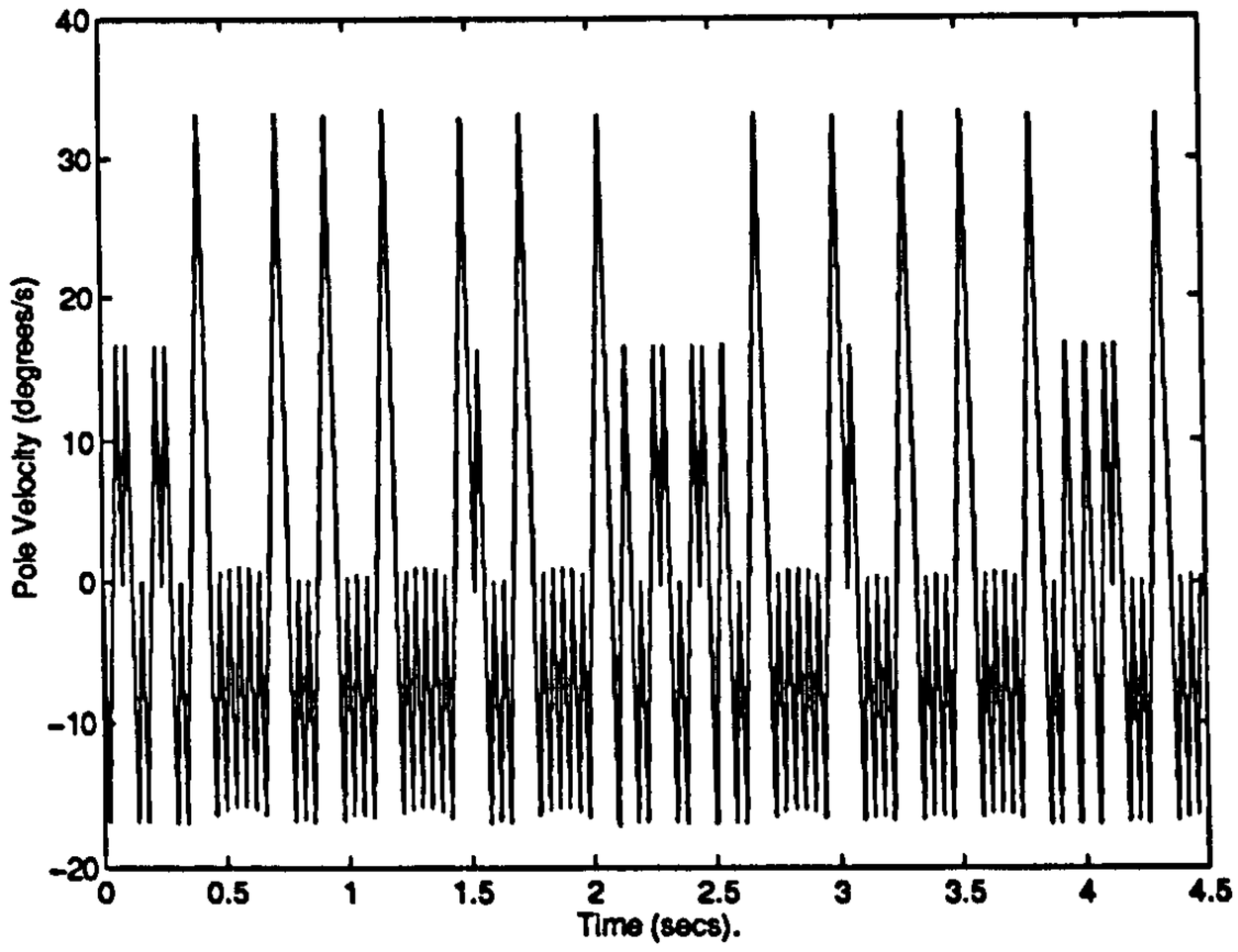


Figure 5.5. The pole angular velocity for the EUCART-BSA pruning simulation Note that the pole velocity is predominantly negative in opposition to the predominantly positive cart velocity.

Cart-pole control is oscillatory as expected when using bang-bang control. Smooth control requires a graded output so that a minimal corrective force may be applied as required. It is interesting to note that both the cart displacement and pole angle remain within a small range of the full scale allowed. Only the range end-points are specified for failure and oscillations up to the failure limits would be allowable although this would not constitute good control. Large oscillations just within the failure limits have been observed in the BSA system (Johnson and Smartt, 1993). It is likely that there will be similar cases observed for the EUCART-BSA hybrid if the control behaviour of many runs is examined.

5.2.3 Issues

The preceding simulation shows that a simple pruning scheme can be successful and that many of the nodes are created spuriously. However, there are two points which must be addressed if pruning is to be a viable extension of the EUCART-RL hybrid. First, pruning by usage—reflected by the RTS—is only useful for a stationary environment. The nodes required to maintain control are used regularly and, consequently, are not pruned. If environmental conditions change which necessitate a new "set-point" or control strategy, then the controller will adapt as

required but the old control strategy will be lost because the RTS of the nodes involved in the previous strategy will decline until they drop below the pruning threshold. If the environment returns to the previous conditions, then the original control strategy will have to be re-learnt. This is undesirable and does not allow the neurocontroller to build up a general control strategy. Clearly, the 6 node controller of the simulation will not be robust in a non-stationary environment. Balance will be maintained about the origin until the operating conditions change and a new control region established. A new method of pruning other than by absolute usage is required. A contextual usage measure may be possible in which a successful localised control strategy is not erased when operating conditions shift, e.g. if for a particular node the RTS exceeds a given relevance threshold in any context then it is deemed important and 'made immune' from removal.

Second, the simple pruning used in the simulation is carried out at set periods using a set threshold. The pruning period and pruning threshold are arbitrary parameters which have to be set by the user. Although it has been demonstrated that the use of pruning is feasible, the introduction of yet more arbitrary parameters is unsatisfactory. The pruning operation has to be made adaptive in some way so that pruning only occurs at relevant intervals and to a relevant degree. How these levels of relevance are decided is another matter and provides directions for further research.

These two points require investigation if improvements are to be made to the hybrid. A further point, though of lesser significance, is that when nodes are removed, useful information may be lost. The GCS system of Fritzke (1991, 1993, 1994) redistributes some of the information. A modified version of this method may be useful here.

The active removal of nodes may be augmented or replaced with techniques which "lump" nodes together to reduce redundant coverage of state-space (Michie and Chambers 1968b). The technique of lumping is closely related to its counterpart of "splitting" to give finer resolution; once a split occurs then redundant sections may be removed by pruning.

5.3. Distributing the EUCART-BSA Hybrid

5.3.1 Introduction

It is a reasonable assumption that distributing information across the decoder by allowing states to be members of more than one "box" will improve performance. This assumption is in accord with the relative smoothness of the cart-pole dynamics (Barto, Sutton and Anderson, 1983). Distribution across boxes is a form of generalisation in that an informed control decision can be made on the basis of information from neighbouring boxes even though a given box has never been entered previously.

The original BSA approach does not preclude distribution; winner-takes-all dynamics as a matter of design choice. A distributed reinforcement learning system is pre-empted by the form of the dynamical equations and the challenge to distribute is clearly stated in Barto, Sutton and Anderson, (1983).

The basis of a distributed approach rests upon finding a weighting system to combine sets of control outputs or predictions. A normalised category membership function is required to indicate the relative activity or contribution of a given box or node.

Before attempting to develop a distributed version of the EUCART-BSA hybrid, it is sensible to explore the feasibility of distributed ASE / ACE dynamics decoupled from the EUCART decoder. The decomposition of the design process makes it simpler and allows distribution to be investigated under less complex conditions. A distributed decoder with a fixed overlap is much easier to deal with than the dynamical EUCART decoder with variable membership functions.

An ideal candidate, which fulfils the fixed overlap condition and is equivalent to the original boxes system at the winner-takes-all limit, is a *fuzzy boxes system*. Such a system introduced in this thesis, forms the subject of section 5.5.

5.3.2 Why Fuzzy? The Distributed White-Box

Fuzzy systems (Zadeh, 1965) are ideal for distribution of information across more than one storage location in a mapping or representation. A fuzzy input-output controller can approximate to any degree of accuracy, a continuous system (Buckley and Hayashi, 1993). Fuzzy logic and fuzzy systems will be covered in section 5.4 onwards. The purpose of this section is to motivate the use of a fuzzy distributed decoder.

The two primary reasons for choosing a fuzzy system have already been mentioned. The overlapping coverage of the input space by fuzzy sets (see section 5.3.3.) and the property of 'boxes in the limit' make fuzzy systems a natural choice for a distributed decoder. Another attractive property of fuzzy systems is the ease of extraction of information in the form of *rules*.

It is difficult to extract knowledge from neural networks such as the MLP by considering the weighted connections between nodes. Knowledge-based systems (KBS), on the other hand, are in the form of rules easily interpretable by a human being. The distributed representation of a neural net may not easily be mapped to a set of rules. For a non-linear network a change in the antecedent of a rule may have a disproportional effect on the consequent thereby making it difficult to trace the effects of weights and activity levels for each rule.

Using a black-box approach of presenting and testing combinations of inputs leads to a combinatorial explosion of tests for all but a trivial number of input variables. Furthermore, rule extraction is an inefficient two-stage process where a network has to be trained prior to rule extraction. A more efficient method would be to train rules from the outset. Using a fuzzy decoder attached to a distributed ASE / ACE system would allow rules to be generated directly to give a self-tuning rule-base.

5.4 Fuzzy Logic

In the subsections that follow, the main concepts of fuzzy logic relevant to the development of ideas covered in this thesis are presented. This introduction to fuzzy logic is not meant to be exhaustive; it allows the thesis to be self-contained without depending too heavily upon auxiliary material. The concepts of fuzzy logic are introduced in anticipation of the discussion of a novel neurocontroller architecture.

Fuzzy logic allows the use of qualitative knowledge—often vague and imprecise—in the form of rules. Quantitative precision is also retained through the use of input-output mappings specified by the set of rules (e.g. Wang and Mendel, 1992; Buckley and Hayashi, 1993).

5.4.1 Crisp Sets and Fuzzy Sets

Traditional logic (e.g. Hamilton, 1988; Mendelson, 1987) deals with *crisp sets*, that is, sets with *membership functions* which map to *binary sets* such as $\{0,1\}$ or $\{True, False\}$. An element of a crisp set will be mapped to one of the binary values in the target set indicating membership or non-membership of the element. This can be stated more formally:

given a crisp set, X, Such that $X \subseteq U$, where U is the *universe of discourse* and the set X is defined by $X = \{x | P(x) \text{ is true}\}$ where P(x) is a truth function of x. The membership function $\overline{\mu}_X(x)$ can be defined as

$$\overline{\mu}_X(x) = \begin{cases} 1 & \text{if } x \in X \quad (P(x) \text{ is true}) \\ 0 & \text{if } x \notin X \quad (P(x) \text{ is false}) \end{cases}$$

for example, denoting the set of even numbers less than ten by $E = \{2,4,6,8\}$ and the set membership function by $\mu_E(e)$, the following facts can be stated,

$2 \in E \Rightarrow \overline{\mu}_E(2) = 1,\ 7 \notin E \Rightarrow \overline{\mu}_E(7) = 0.$

This polarisation of set membership represents the extreme case. In reality set membership (concept instantiation) is often not clear cut. Grades of set membership can exist on a continuum between zero (non membership) and one (full membership) e.g., for the concept pair young-old, when is someone young? From traditional logic a boundary point has to be defined. Does this make sense? Does a single second either side of the boundary point make a difference?

By defining a continuous membership function $\mu_X(x)$ the continuum of membership can be dealt with. More formally, the fuzzy membership can be stated as

$$\mu_X : X \rightarrow [0,1], \quad \mu_X(x) \mapsto m \in [0,1]$$

Fuzzy logic deals with *fuzzy sets*, that is, with sets having continuous membership function (Zadeh, 1965; Pedrycz, 1993; Kruse, Gebhardt, and Palm 1994; Kruse, Gebhardt and Klawonn, 1994). Viewed in this manner, traditional logic is a subset of fuzzy logic with the binary set values comprising the extremes of the fuzzy membership continuum. Conversion between fuzzy and crisp set memberships is achieved by specifying a cut-off boundary in the membership continuum and assigning elements to crisp sets depending upon whether or not they are above or below the cut-off point.

The power of fuzzy logic lies in its ability to deal with imprecise linguistic information represented by concepts such as "hot", "warm", "cold", "small" or "medium". The set of all numerical values (e.g. range of temperatures) involved in a given application of fuzzy logic is known as the universe of discourse, again denoted by U. This universe of discourse is coded by a group of *linguistic variables* e.g. temperature, pressure, and angle. The linguistic variables are composed of a number of *terms* representing imprecise quantifications of the linguistic variable. For example the linguistic variable of angle can be quantified by the set of terms,

$$X_\theta = \{LN, SN, NZ, SP, LP\}$$

where LN, SN, NZ, SP and LP represent the linguistic terms large negative, small negative, near zero, small positive and large positive respectively. The set of

linguistic terms for each linguistic variable, taken together, are said to *cover* the universe of discourse.

There are many forms of membership function. An example is shown in Figure 5.6.

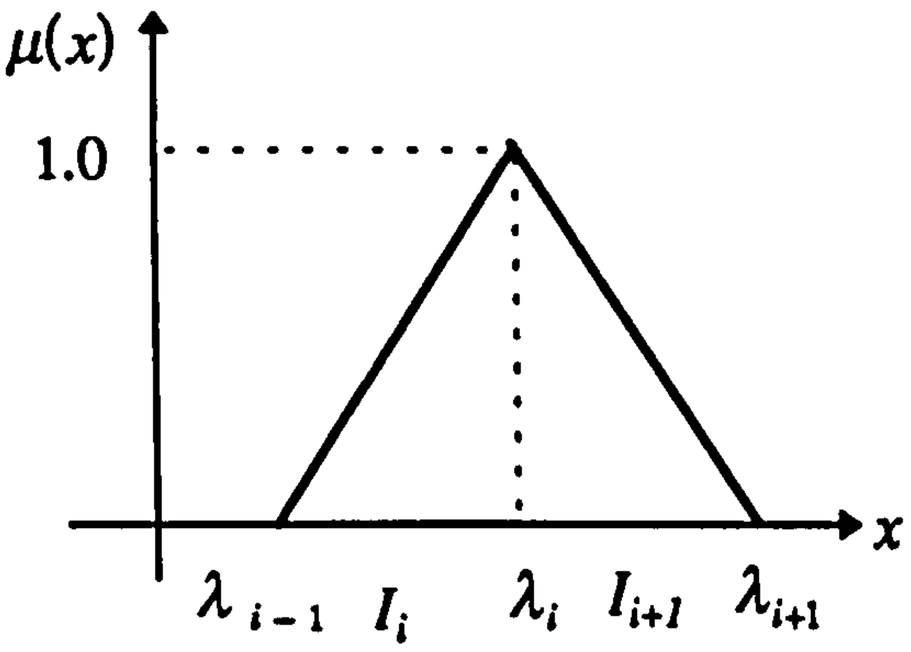


Figure 5.6 A triangular membership function often used to fuzzify a given variable.

The *triangular* membership function is commonly used in applications owing to its simplicity. Each dimension of the input space is divided into several intervals, I_i. A set of points, $\{\lambda_i\}$ along the input dimension, called *knots*, determines the size and location of the intervals, and thus, the width of fuzzy sets along the input dimension.

For some input lying in an interval, i.e. $x \in I_i = [\lambda_{i-1}, \lambda_i]$, membership of the fuzzy sets A_{i-1} and A_i is defined by

$$\mu_{A_{i-1}}(x) = \frac{\lambda_i - x}{\lambda_i - \lambda_{i-1}} \qquad (5.1)$$

and

$$\mu_{A_i}(x) = \frac{x - \lambda_{i-1}}{\lambda_i - \lambda_{i-1}} \qquad (5.2)$$

respectively.

Figure 5.7 shows a set of 5 triangular basis functions covering the input space. The first and last triangular functions are given a constant value of 1.0 beyond the end-points specified by knots. The triangular functions are *B-splines* of order 2 (Brown and Harris, 1994). The sets of basis functions covering each dimension,

taken together, form a *fuzzification* of the input space. B-splines form a parameterised class of sets of basis functions of many orders. Each individual set of a given order covers an input space. Sets of Gaussian basis functions may also be used.

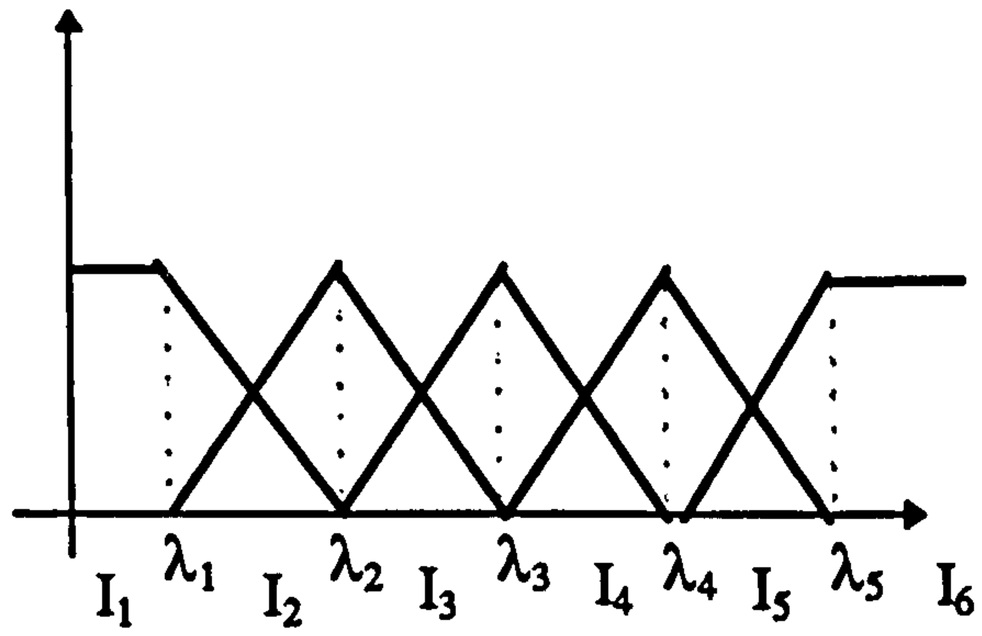


Figure 5.7. The fuzzification of a real variable using a set of triangular functions.

In the case illustrated in Figure 5.7, equations (5.1) and (5.2) hold for $i = 2,\ldots,5$.

For $x \in I_1 = [-\infty, \lambda_1]$, and for $x \in I_i = [\lambda_{i-1}, \lambda_i]$

The *support* of a fuzzy set, *A*, denoted here by *S(A)*, is defined as inputs,

$S(A) = \{x \in X : \mu_A(x) > 0\}$. If $S(A) \subset X$, a support is said to be *compact*.

If at each point (element) of a linguistic variable the membership functions applied to that point for each linguistic term sum to one, then the fuzzy sets (terms) are said to be *normalised*. For example, for the linguistic variable "angle", denoted by X_θ, the fuzzy membership functions conform to

$$\mu_{\theta,LN}(\theta) + \mu_{\theta,SN}(\theta) + \mu_{\Theta,NZ}(\theta) + \mu_{\theta,SP}(\theta) + \mu_{\theta,LP}(\theta) = 1$$

for all values of θ, where $\mu_{\Theta,LN}$ is the fuzzy membership function for the linguistic variable "angle" and so on. Normalised fuzzy sets are used subsequently. Note that there are a few general conditions on fuzzy membership functions such as their having to be defined at every point. Smooth membership functions such as Gaussian can also be used.

For the cart-pole problem, the universe of discourse, U consists of the set of all state vectors, $\mathbf{x} = (x, \dot{x}, \theta, \dot{\theta})$ with dimensions distance, speed, angle and angular velocity respectively. The four dimensions of the state vector form the four

linguistic variables of interest. Each of the four variables are covered by the five linguistic terms of LN, SN, NZ, SP and LP respectively triangular membership functions are chosen, as stated previously for the angle example.

5.4.2 A Fuzzy System (Mapping)

The main advantage of fuzzy systems is that they allow both the construction and representation of qualitative mappings and the possibility of "interrogation" of the internal structure; here, the "black-box" has been replaced by a "white-box" consisting of explicit rules understandable by a human being. Using a fuzzy rule base also allows the easier integration of *a priori* information into a learning system; known qualitative knowledge concerning a problem can be formulated in terms of "if-then" rules which are used to prime the rule base. A fuzzy system or mapping consists of a rule-base which codes the system knowledge and two algorithms which carry out the respective transformations between the input space and rule space and rule space and output space (Figure 5.8). Input data is fuzzified and used to select a set of relevant rules from the rule-base.

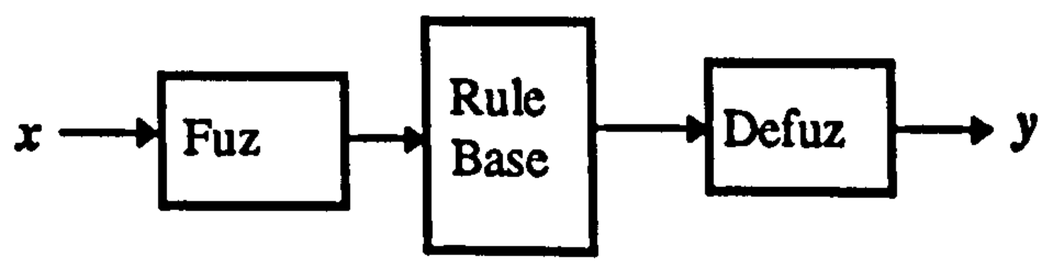


Figure 5.8 A schematic diagram of a fuzzy system. Inputs are fuzzified before being applied to the rule-base. The fuzzy outputs are combined and defuzzified to give the final output.

The selected rules are combined to give a fuzzy output which is then defuzzified to give a real output.

For a crisp input, x, components x_i lie within open or closed intervals, I_i where $i = 1,\ldots,m = \dim\{\mathbf{x}\}$. Similarly for y_j where $j = 1,\ldots,n = \dim\{\mathbf{y}\}$. Each

interval, I_i, is covered by M functions and each interval, I_j is covered by N functions.

So, for each of the m dimensions of x, there are M linguistic variables giving M^m possible rule antecedents ("if" parts). Similarly, for each of the n dimensions of y, there are N linguistic variables giving N^n possible consequents. This gives a total number of possible rules of $M^m \times N^n$. Where M and N are not fixed for all intervals, M_i and N_j are defined giving $\prod_{i=1}^{m} M_i$ possible antecedents and $\prod_{j=1}^{n} N_j$ possible consequents; this gives a total number of possible rules of $\left(\prod_{i=1}^{m} M_i\right)\left(\prod_{j=1}^{n} N_j\right)$. The curse of dimensionality should be apparent from this analysis; however, in practice, all of the rules may not be used.

As an example of a relatively simple system, consider the possible rule base required to control the cart-pole system. There are four state variables giving m=4. Using five linguistic variables for each input interval, M=5. For bang-bang control, where the output is not fuzzified, n=1 and N=1. In this case the total number of possible rules is given by $5^4 = 625$. This fuzzy rule-base will be covered in the discussion of a novel self organising fuzzy controller, FUZBOX discussed in section 5.5.

The large number of possible rules gives rules at the lowest level of rule generality, including all antecedent clauses. Lumping and splitting of rules is relevant to the discussion of fuzzy systems.

5.4.3 A Fuzzy Controller

Fuzzy systems are now used widely in control (e.g. Procyk and Mamdani, 1979, Pedrycz, 1993; Linkens and Abbod, 1993; Nie and Linkens, 1994). A novel self-organising fuzzy control system, FUZBOX, which uses reinforcement learning, is

introduced in the next section. The present section will introduce the subject of fuzzy control in anticipation of the discussion. A simple form of fuzzy controller is shown in Figure 5.9

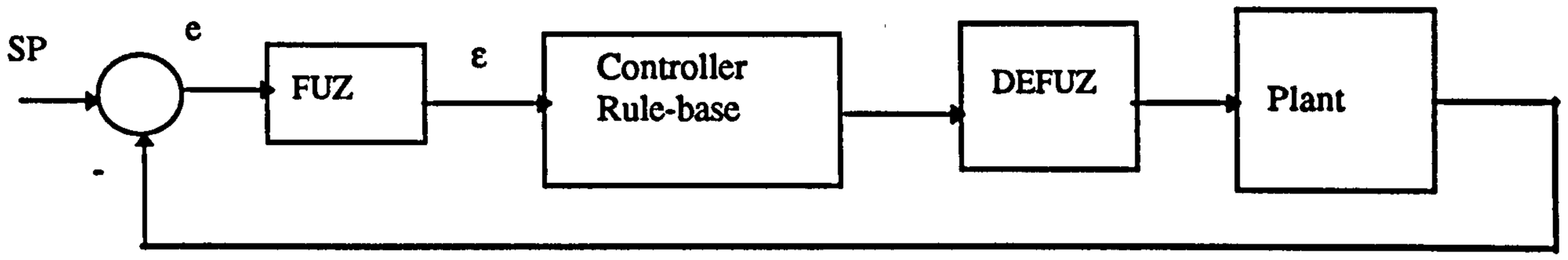


Figure 5.9. A schematic representation of a fuzzy logic controller. Fuzzification and defuzzification are denoted by FUZ and DEFUZ respectively.

The fuzzy controller of Figure 5.9 is a SISO system which uses a real error, e which is fuzzified to give a fuzzy error, ε. The controller is specified in the form of a *rule matrix* (Procyk and Mamdani, 1979) which gives the control output, u, governed by the error, ε, and the change in error, $\Delta\varepsilon$. For example, a small system with five linguistic variables given by LN, SN, NZ, SP, and LP may have a rule matrix given by

	LN	*SN*	*NZ*	*SP*	*LP*
LN					
SN			*NZ*	*SP*	
NZ		*SP*	*NZ*	*SN*	
SP			*SN*		
LP					

The rule matrix is an input-output mapping which is transparent to a user and yet still allows a quantitative mapping for control (Brown and Harris, 1994). Because the fuzzification process, rule matrix and defuzzification process are all known, a control law can be formulated directly in non-fuzzy terms, $u(e,\Delta e)$ if required.

Fuzzy logic control is closely related to neurocontrol and fuzzy neurocontrollers have been developed (e.g. Brown and Harris, 1994 Linkens and Nie, 1994). Fuzzy sets can be thought of as being analogous to basis functions which are combined to implement a mapping.

5.5 Distribution by Membership Function: FUZBOX

5.5.1 Distributed Reinforcement Learning Using Fuzzy Methods

Thus far, a hybrid approach to neurocontrol has been explored which combines the flexibility of self-organisation with the adaptability of reinforcement learning and its suitability to information-poor environments. As discussed, the hybrid approach shows much promise and is worth developing further. However, problems still remain and improvements in performance are required if such a system is to be of any practical use. One area for improvement, the possibility of distributing the ASE and ACE elements was mentioned in section 5.2.1.

Fuzzy systems are a natural choice for developing a prototypical distributed system because of the graded membership functions. Furthermore, the use of a rule-base is a natural extension of the boxes concept where the boxes form a crude rule base in the original undistributed formulations (Michie and Chambers, 1968a; Barto, Sutton And Anderson, 1983).

By demonstrating the use of distribution in a modified ASE / ASE system, it is indicated that a distributed version of the EUCART+BSA hybrid is possible which will allow the combination of neurocontroller outputs for a state which lies within the boundaries of two or more state nodes.

A novel fuzzy neurocontroller architecture, given the name FUZBOX, is introduced in this section. This architecture demonstrates that distribution of both the ASE and ACE modules is, indeed, possible and that learning—in the case of FUZBOX—is accelerated compared with the EUCART-BSA hybrid.

5.5.2 Direct Fuzzy Control

A fuzzy knowledge-based neural network (FKBNN) is proposed by Alché-Buc, Andrès, and Nadal (1992) which solves the cart-pole problem. The FKBNN allows the extraction of decision rules from an artificial neural system. The neurocontroller implements rules of the form: *if X is* A_i *then Z is* B_j. The network is composed of three layers:

- a *condition* layer which has a fuzzy set associated with each node and each node computes the membership function;
- a *conclusion* layer whose nodes associate fuzzy sets with their consequences; the weights on the incoming links from the condition layer are stored in a matrix representing the strength of if-then relations between conditions and conclusions;
- a *combination* layer which combines the rules to give an output; the output node computes the centre-of-gravity defuzzification.

The network uses supervised learning implemented by a modified form of the backpropagation algorithm which is divided into three separate steps:

i) the fuzzy sets in the first layer are corrected with all other parameters being frozen; this process gives a preliminary approximation to the rule conditions,
ii) when step i) is satisfactory, the first layer fuzzy sets are frozen and the network learns the strengths of the if-then relations; this is *rule identification*;
iii) finally, all parameters of the net are involved in optimisation, especially the centres of the consequent sets.

Learning is off-line and involves the minimisation of a specified error function which contains arbitrary parameters. The error function consists of three separate sections which are designed to impose *correctness*, *completeness* and *consistency* upon the network.

The FKBNN was applied to the simplified case of the cart pole system only, which dealt with the pendulum angle and angular velocity. Nine rules were used which gave successful control performance. The paper stated that a possible next step forward was to put a FKBNN into a control loop where feedback was provided by the pendulum itself. On-line *in situ* control similar to this proposal has been achieved by FUZBOX; the main differences between FUZBOX and FKBNN are that FUZBOX

- is autonomous,
- uses fixed fuzzy membership functions,
- does not use fuzzy sets at the output, which would entail defuzzification, and
- does not require the specification of desired intermediate control actions.

For convenience, the cart-pole control problem is resolved into two decoupled tasks:

- the *self-organisation task*, which involves the autonomous categorisation of input information to provide a basis for subsequent control actions, and

- the *control action learning task*, which involves the evaluation and correction of elicited control actions associated with individual states or distinct regions of state-space represented internally by the neurocontroller.

The hybrid approach is possible because the two tasks are decoupled and can be solved independently. For example, solution of the control action learning task only requires the assumption that individual regions of state-space are represented such that they provide a unique "key" to associated control actions which are stored separately from the internal representation of state-space. The assignment of credit or blame and the updating of individual control actions is not linked inextricably to the method of state-space partitioning provided that there is a one-

to-one correspondence between the evaluated region of state-space and its associated control action.

Decoupling the tasks also allows the possibility of other related neurocontroller methods. Indeed, as stated here, the first task is too restrictive and can be replaced by the more general

- *decoder task*, which involves the decoding of state-space into a representation with which individual control actions can be associated.

Note that the term 'decoder' is used as opposed to the term 'encoder'; this usage is consistent with the BSA formulation of reinforcement learning, and reflects the analogy between the BSA state-space decoder and a computer memory address decoder which decodes input addresses to allow access to physical memory locations (Barto *et al*, 1983). The state-space decoder task is to treat the state as a decoder "address" pointing to an associated control action stored within the neurocontroller.

The statement of the decoder task says nothing about its nature; for example, it can have a fixed structure and act as an indexing system for a control action look up table (Michie and Chambers, 1968a) or it can be self-organising as is the EUCART decoder. Also, both discrete-valued and continuous-valued decoders are possible; the latter consisting of a smooth mapping between a continuous input space and a continuous output space.

The original BSA approach can be decoupled into the decoder task and the control action learning task. Here the decoder indexes a fixed state-space representation which is in the form of a look-up table; as learning proceeds, the look-up table is filled (Barto *et al*, 1983). It is precisely because the two tasks are decoupled that other types of state-space decoder are possible whilst retaining the original BSA implementation of reinforcement learning.

Fuzzy-logic-based neurocontrollers allow a natural decoupling of the two tasks by treating the decoder task as one of determining *rule antecedents* and the control action learning task as one of determining associated *rule consequents* (e.g. Berenji and Khedkar, 1992; Jang, 1992,1993; Jang and Sun, 1995). For example, the generalised approximate-reasoning-based intelligent control (GARIC) architecture of Berenji and Khedkar (1992) uses an action selection network (ASN) and an action evaluation network (AEN) which are analogous to the ASE and ACE respectively. Here, although the decoding task is independent of the control action learning task, the state-space decoding is not carried out by a separate decoder system but is subsumed within the operations of the ASN. States are decoded into the constituent *terms* of *linguistic variables* where each linguistic variable consists of a set of terms. One term is selected from each individual linguistic variable; the selected term represents the "value" of the linguistic variable, e.g. the term "near zero" might be selected from the set of terms comprising the linguistic variable "velocity". The resultant collection of selected terms, consisting of a single term from each set, comprises a rule antecedent. Note that individual states are not represented by individual rules in this case; a single state can fulfil the antecedent conditions of more than one rule and thereby trigger multiple rules. Rule antecedents can be viewed as characterising distinct regions of dynamical space which overlap in places where multiple rules are involved. Control actions are associated with input states through fuzzy encoding as rule consequents. GARIC uses reinforcement learning to tune the fuzzy rule base so that it is able to represent the desired control mapping. The fuzzy encoding of both the input terms and the output terms is tuned adaptively. A fixed number of rules is used to solve the cart-pole problem.

The adaptive-network-based fuzzy inference system (ANFIS) of Jang (1992,1993; Jang and Sun, 1995) also tunes the fuzzy encoding of the state-space input patterns to form the rule antecedent terms but treats the rule consequents as real functions which are linear-in-the-parameters. These functions determine the control output when ANFIS is used to control the cart-pole problem, again, using a fixed number of rules. Learning consist of two stages: a forward pass to update the consequent parameters by recursive least squares estimation, and a backward

pass to update the antecedent parameters by gradient descent down the error energy surface determined by the input fuzzification. Thus, the two tasks are decoupled and separate learning phases are used.

A fuzzy version of the ASE element was developed by Johnson and Smartt (1993). It was based upon the observation that the decoder-ASE combination formed a type of expert system (Johnson and Smartt, 1993). The decoder represented the set of possible antecedents and the ASE the set of consequents.

The *fuzzy associative search element* (FASE) used a continuous output obtained by defuzzification of the consequents. As with the BSA system, the cart-pole problem was formulated in terms of four state variables.

Two linguistic variables were used for both the antecedent components and the consequent. The combination of state and linguistic variables gives $2^4 = 16$ rules if a single linguistic variable is chosen for each rule. The rule base is fixed in size and an appropriate output is defined for each rule in the rule base. One node is defined for each rule. The decoder assigns a rule applicability value, x_i, to each of the 16 rules where $i = 1,\ldots,16$. The weighted consequents are then defuzzified to give a continuous output value.

Johnson and Smartt (1993) note that the pole angle oscillates considerably in the BSA implementation and just manages to stay within the failure limits. The FASE implementation allows much smoother control.

FUZBOX, unlike FASE, does not fix the number of rules from the outset and allows the correct output actions to be learned. The output is still bang-bang but appears much less oscillatory when compared with the original BSA implementation e.g the pole angle remained within a degree either side of vertical for a number of cases.

The FUZBOX system is similar to the fuzzified actor / critic system of Jianan, Mohammed and Shihuang (1995) but does not use a CMAC to store the rules. Here, it was found that similar results were achieved using FUZBOX without the need for CMAC-based pre-processing. This indicates that using a CMAC to store "box" information (Lin and Kim, 1992), whilst effective for improving performance for a discrete actor / critic system, is possibly redundant if a distributed actor / critic is used.

5.5.3 FUZBOX: Description

The FUZBOX system is also based upon treating state boxes as rules and using fuzzy membership to distribute learned information. Bang-bang control is retained using a special case of the Sugeno method (Sugeno and Nishida, 1985) where linguistic variables are not used at the output. Instead for each rule with an antecedent of the form

R^i: x_1 *is* A_i^1,...,x_n *is* A_i^n,

the single valued consequent is of the form

$y^i = p_i^0 + p_i^1 x_1 +,\ldots,+p_i^n x_n$.

For FUZBOX, $y^i = p_i^0$ where $p_i^0 \in \{-1,+1\}$.

The maximum possible number of rules for FUZBOX is 625 which is determined by the use of four state variables and five linguistic variables for each of the state intervals. Each of the 625 possible antecedents is assigned one output only. The new state-space partition uses the linguistic variables LN, SN, NZ, SP, and LP. A typical rule is given by,

If x *is SN and* $\dot{x}$ *is SN and* θ *is SP and* $\dot{\theta}$ *is SP then output is +1;*

The knot vectors determining the fuzzification are chosen to be

$\mathbf{k}_x = [-1.8,-,1.4,0.0,1.4,1.8]$ $\mathbf{k}_{\dot{x}} = [-3.0,-1.0,0.01.0,3.0,]$

$\mathbf{k}_\theta = [-10.0,-2.0,0.0,2.0,10.0]$,and $\mathbf{k}_{\dot{\theta}} = [-80.0,-20.0,0.0,20.0,80.0]$

The relevance of a rule for a given input is measured by the *rule antecedent strength* (RAS) which combines the membership values of each state variable belonging to the fuzzy set associated with each linguistic variable. The RAS for the i th rule is defined by $(\mu_i(\mathbf{x}) = \mu_{x,A_j}(x)\mu_{xA_k}(\dot{x})\mu_{\theta,A_l}(\theta)\mu_{\theta,A_m}(\dot{\theta}))$ where, $\mu_{x,A_j}(x)$ denotes the fuzzy membership of the cart displacement with respect to the fuzzy set associated with the linguistic variable A_j and so on.

Rules are added incrementally if the hash code of the rule with the highest possible RAS indicates a non-existent rule so that previously encountered state-space regions are represented by at least the most relevant rule as part of the rule base.

The rules triggered by the input are combined to give a weighted average

$$y^{out} = \sum_{i=1}^{r} w_i(\mathbf{x}) y^i$$

where $w_i(\mathbf{x})$ is the normalised membership function given by

$$w_i(\mathbf{x}) = \frac{\mu_i(\mathbf{x})}{\sum_{k=1}^{r} \mu_{ki}(\mathbf{x})}$$

and r is the number of rules combined to give the output.

This normalisation is required because although $0 \le \mu_i(\mathbf{x}) \le 1, \forall \mathbf{x}$ and $\sum_{i=1}^{N_R} \mu_i(\mathbf{x}) = 1$ where N_R is the maximum number of possible rules, rules are added incrementally and only rules that are available are used. The number of rules used (combined), *r*, may be for example: all the current rules; all the rules triggered by the input; a subset of available rules depending upon a threshold, or a pre-maximum set number. FUZBOX sets *r* to the number of rules triggered by the

input to indicate the number of rules involved in determining the control output which is given by $u = 10\,\mathrm{sgn}(y^{out})$

Denoting the rule base at time instant, t, by $\Re_t$, all rules R_i such that $R_i \in \Re_t$ are updated according to the dynamics described in sections 5.5.4 and 5.5.5.

The modified actor / critic elements of FUZBOX, labelled distributed ASE (DASE) and distributed ACE (DACE) respectively, operate as the original BSA implementation when a single rule is chosen using winner-takes-all based upon the RAS ($\mu_i(\mathbf{x})$).

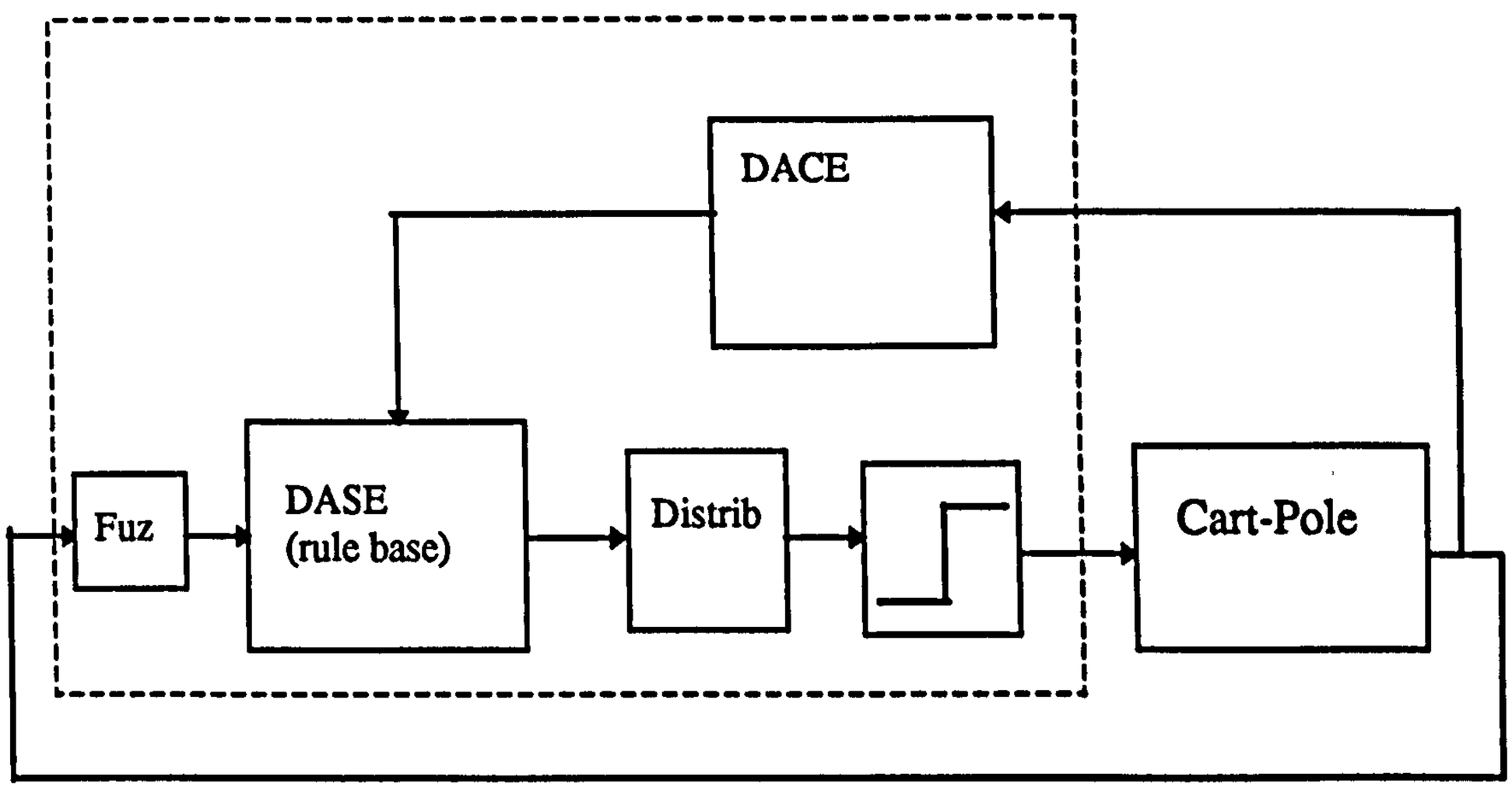


Figure 5.10 The FUZBOX neurocontroller. Fuz denotes the fuzzification process detailed in the text and distrib denotes the combination of rule information to give the weighted average output. This is then used to generate the actual control output.

5.5.4 Distributed ASE dynamics

DASE dynamics are similar to those of the original ASE system but with a normalised scalar parameter to weight the individual contributions. The final control output, u is given by

$$u(t) = 10\,\mathrm{sgn}\left(y^{out}(t) + \varepsilon(t)\right)$$

where $y^{out}(t)$ is the weighted output consisting of contributions from active rules given by $y^{out}(t) = \sum_{i=1}^{r} w(\mathbf{x}) y^i(t)$ and $\varepsilon(t) \sim N(0,1)$ is Gaussian noise derived from a zero mean source with unit variance. The factor of 10 scales up the output to ±10 Newtons.

The output for an individual rule is given by $y^i(t) = \mathrm{sgn}\left(z_i(t)\right)$ is the where $z_i(t)$ is the DASE weight for the i th rule.

The DASE weight evolution equation for the ith rule is given by,

$$z_i(t+1) = z_i(t) + a\hat{r}(t)e_i(t)$$

as for the ASE element where $\hat{r}(t)$ is the real-valued reinforcement at time, t, a is again the positive rate of change constant which determines the magnitude of change for the output weight time evolution and $e_i(t)$ is the eligibility trace with the evolution equation

$$e_i(t+1) = \delta e_i(t) + (1-\delta) w_i\left(\mathbf{x}(t)\right) y^i(t)$$

which now contains the weighting term.

5.5.5 Distributed ACE dynamics

The DACE is similar in structure to the ACE with the weighting term included to distribute activity across a set of rules.

The distributed prediction of expected reinforcement is given by

$$p(t) = \sum_{i=1}^{r} q_i(t) w_i\left(\mathbf{x}(t)\right) \qquad (5.3)$$

where $q_i(t)$ is the weight for the i^{th} rule and $w_i(\mathbf{x}(t))$ is the input weighting for that rule as before.

The weight evolution equation is given by

$$q_i(t+1) = q_i(t) + b\hat{r}(t)\bar{x}_i(t)$$

where $b > 0$, is a constant which determines the rate of change of learning in q_i, $\hat{r}(t)$ is the predicted reinforcement and $\bar{x}_i(t)$ is a trace of the activity of the input variable, x_i.

Unlike the eligibility trace, this trace does not take into account the control action chosen by the system for the region of state-space but now incorporates the weighting factor. It is given by:

$$\bar{x}_i(t+1) = \lambda\bar{x}_i(t) + (1-\lambda)w_i(\mathbf{x}(t))$$

Where λ, $0 \le \lambda < 1$ is a rate of change constant. The trace provides a record of the activity of the i th rule analogous to the activity of the i th input line, x_i, in the original ACE element and enables the determination of the contribution of that particular rule to the prediction. With the new, distributed, protocol of selecting multiple active rules, equation (5.3) gives the weighted prediction of failure for a combination of overlapping rules coding for neighbouring regions of state-space.

The predicted reinforcement is given by

$$\hat{r}(t) = r(t) + \gamma p(t) - p(t-1)$$

where r(t) is the external reinforcement, $r(t) \in \{0,-1\}$, and γ, $0 < \gamma \le 1$, is a discounting factor.

Sub-section 5.5.6 presents simulation results showing the application of FUZBOX to the cart-pole problem. The effects of using a distributed representation are demonstrated clearly. The FUZBOX results are compared with the previous reinforcement learning implementations discussed earlier.

5.5.6 FUZBOX: Some Results

This sub-section investigates the use of a distributed representation for the ASE/ACE elements in the novel reinforcement learning architecture, FUZBOX. As will be seen, the original objective of establishing the viability of distributed reinforcement learning has been fulfilled. The investigation of a distributed representation was carried out using a different structure from that of the EUCART-BSA hybrid so that issues affecting distribution were isolated from possible complications introduced by using self-organising clusters. Once the viability of distribution was established using a fixed representation—boxes with fuzzy boundaries—then generalisation to the self-organising version could be investigated.

Simulations, again following the method of Barto *et al* (1983) comprising 10 runs of 100 trials each, were carried out. As in the BSA implementation, the state vector was reset to $x = \dot{x} = \theta = \dot{\theta} = 0$ after each trial. The simulation conditions and parameters were similar to those of the BSA implementation except that, again, runs were not terminated when the trial of a particular run first reached the ceiling of 500,000 time steps As with the EUCART-BSA hybrid, learning was still occurring in some cases and the system had to reach the ceiling value a large number of times consecutively to indicate convergence. For the FUZBOX simulations, the learning parameter, a was set to 1,000 as in the BSA implementation to establish control actions quickly. Rules are added incrementally if the rule does not exist which would have the highest possible RAS.

Table 5.1 shows the results of the first ten runs using FUZBOX. Comparing Table 5.1 with Table 4.1 of section 4.3.2 and Table 4.2 of section 4.3.4 indicates that the number of trials required to converge to a solution of the control problem is generally lower for FUZBOX in comparison with the EUCART-BSA hybrid given the same cart-pole and noise conditions. This is confirmed by the average convergence time of 45.9 trials for FUZBOX (over 10 runs) compared with 407.7 trials for the EUCART-BSA hybrid (over 10 runs) and 83.8 for the original

BSA system. The results for a further ten runs are shown in Table 5.2. The average convergence time has increased to 61.2 for twenty runs.

seed	1	2	3	4	5	6	7	8	9	10
trials	31	31	42	59	54	70	54	37	29	52
rules	152	215	200	260	223	210	202	160	196	158

Table 5.1 The first ten runs of the FUZBOX simulation using similar conditions to those of sections 4.3.2. and 4.3.4. The same initial conditions and random number seeds are used to provide a direct comparison.

seed	11	12	13	14	15	16	17	18	19	20
trials	179	29	227	113	23	29	55	16	76	18
rules	333	183	273	236	175	158	251	163	270	157

Table 5.2 A further ten runs of the FUZBOX simulation using the same conditions as the simulation illustrated in Table 5.2

These results for FUZBOX indicate that distribution of information across several boxes decreases the learning time required to acquire a successful control strategy for the given initial conditions. Figures 5.11 and 5.12 illustrate the performance of FUZBOX for the first 10 runs featured in Table 5.1. All of these runs converged within 100 trials. The solid curve shows the average pole-balancing time over the 10 runs for each trial. Again, a single point is plotted to indicate the average of each bin of 5 consecutive trial (ensemble) averages. The dotted curves show 1 standard deviation either side of the mean. The circles at the top of the graph indicate at which trial the members of the 10 run set converged.

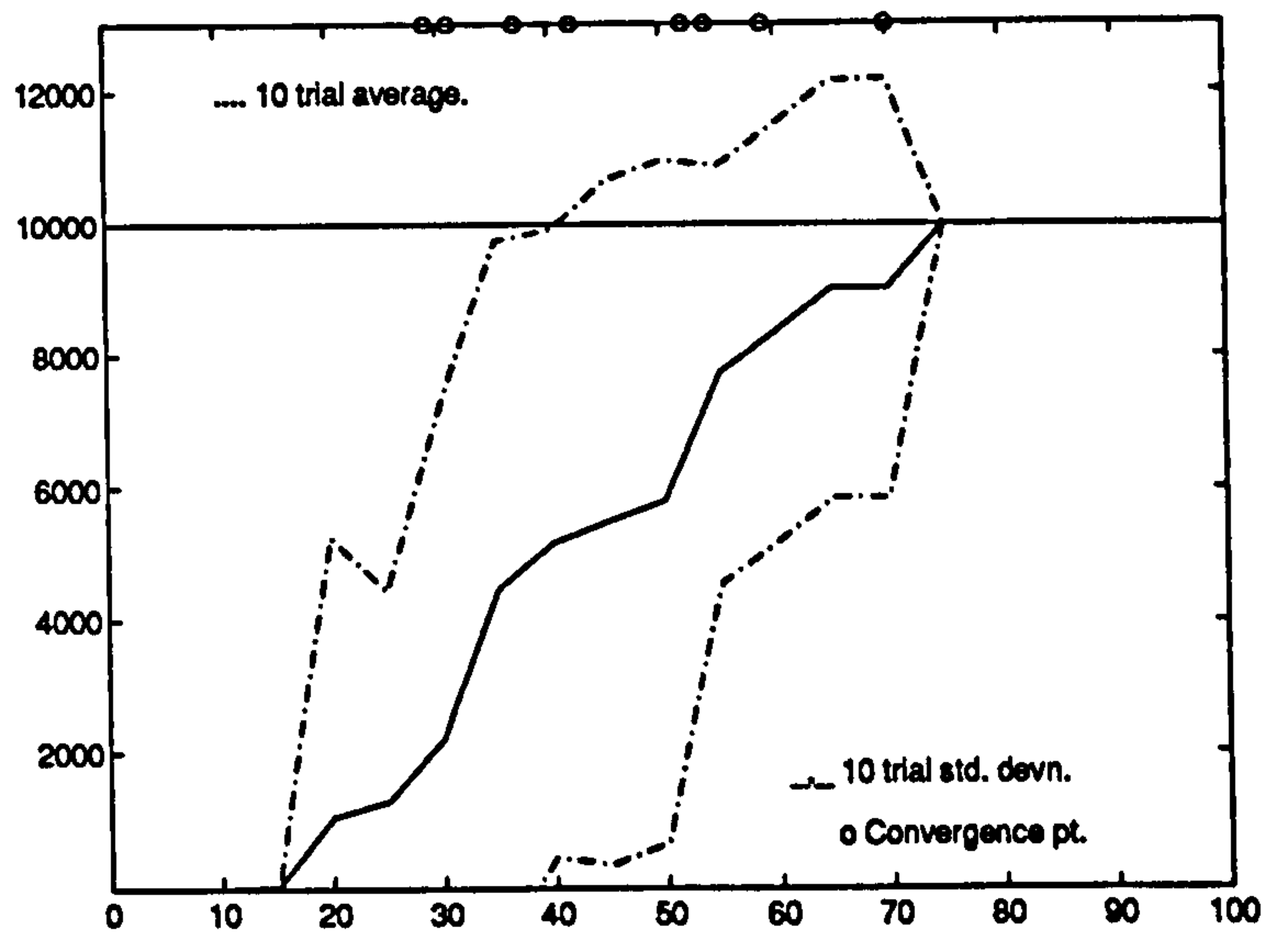


Figure 5.11 FUZBOX simulation results showing the first 10 runs. Note that there are two coincident convergences at 31 and 54 trials respectively.

Comparing this with Figure 4.9 of section 4.3.2 illustrates visually the more rapid convergence of FUZBOX as compared with the EUCART-BSA hybrid. One important thing to note is that the pole-balancing durations increase monotonically. Although the use of 'bins' of five consecutive trials smoothes the curve and removes some of the fluctuations, there are no drops in average performance compared to the EUCART-BSA hybrid

Figure 5.12 illustrates the increase in number of rules (boxes) as a function of trial number. The curve appears to approach an asymptotic value of approximately 200 rules. This means that approximately 425 rules remain unused for this set of initial conditions. Dynamic allocation of rules prevents the allocation and use of redundant memory, thus, reducing computational overheads. It is likely that more rules will be required for more demanding initial conditions and will be allocated accordingly.

Another advantage of dynamic allocation of rules is that it facilitates pruning of redundant rules. Rules may be removed from the rule-base and thereby from the storage requirements of the system.

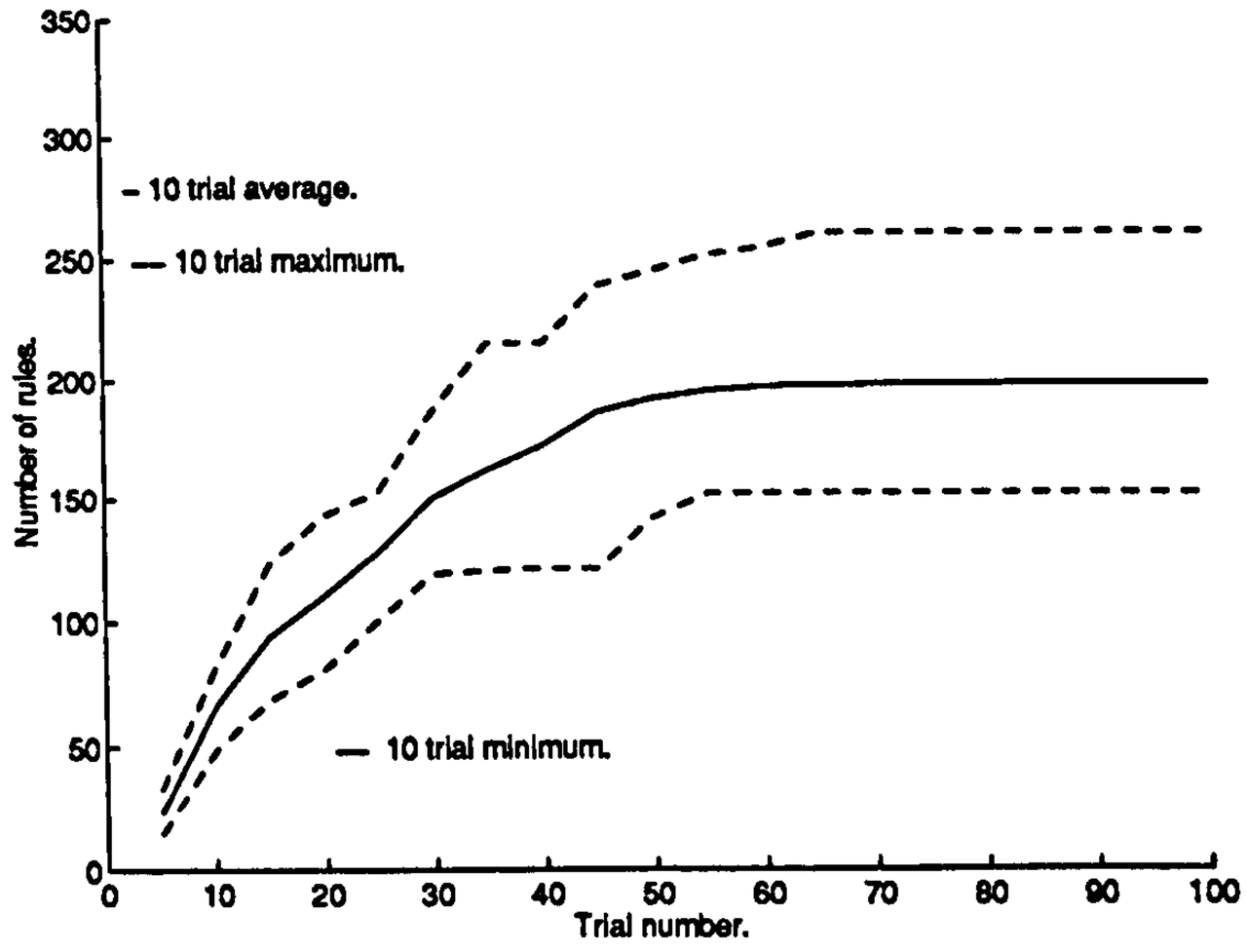


Figure 5.12 The average number of rules for ten runs using the FUZBOX neurocontroller. Note that about 200 rules are generated on average compared with the possible 625.

A single run of FUZBOX was carried out using the conditions given for the 20 run set except that the pole angle was initialised to 11 degrees from the vertical for training and testing. Figure 5.13 shows the phase plane plot for this run. Figure 5.14 shows the cart-position for the first 8.5 seconds and illustrates clearly the use of predominantly right directed forces to rectify the pole. This control policy pushes the cart to around 1.2m away from the origin after which corrective action attempts to push the cart back to the origin without losing control of the pole.

Figure 5.15 is commensurate with this and shows the transition between positive cart velocity and negative cart velocity as control emphasis switches from the pole to the cart. In other words, for the pole initial condition of 11 degrees, control forces have to be predominately right-directed giving the cart a positive velocity (and displacement). To compensate for this, the cart velocity is made negative with rapid switching to maintain the pole balance (Fig 5.15). The cart then moves towards the origin.

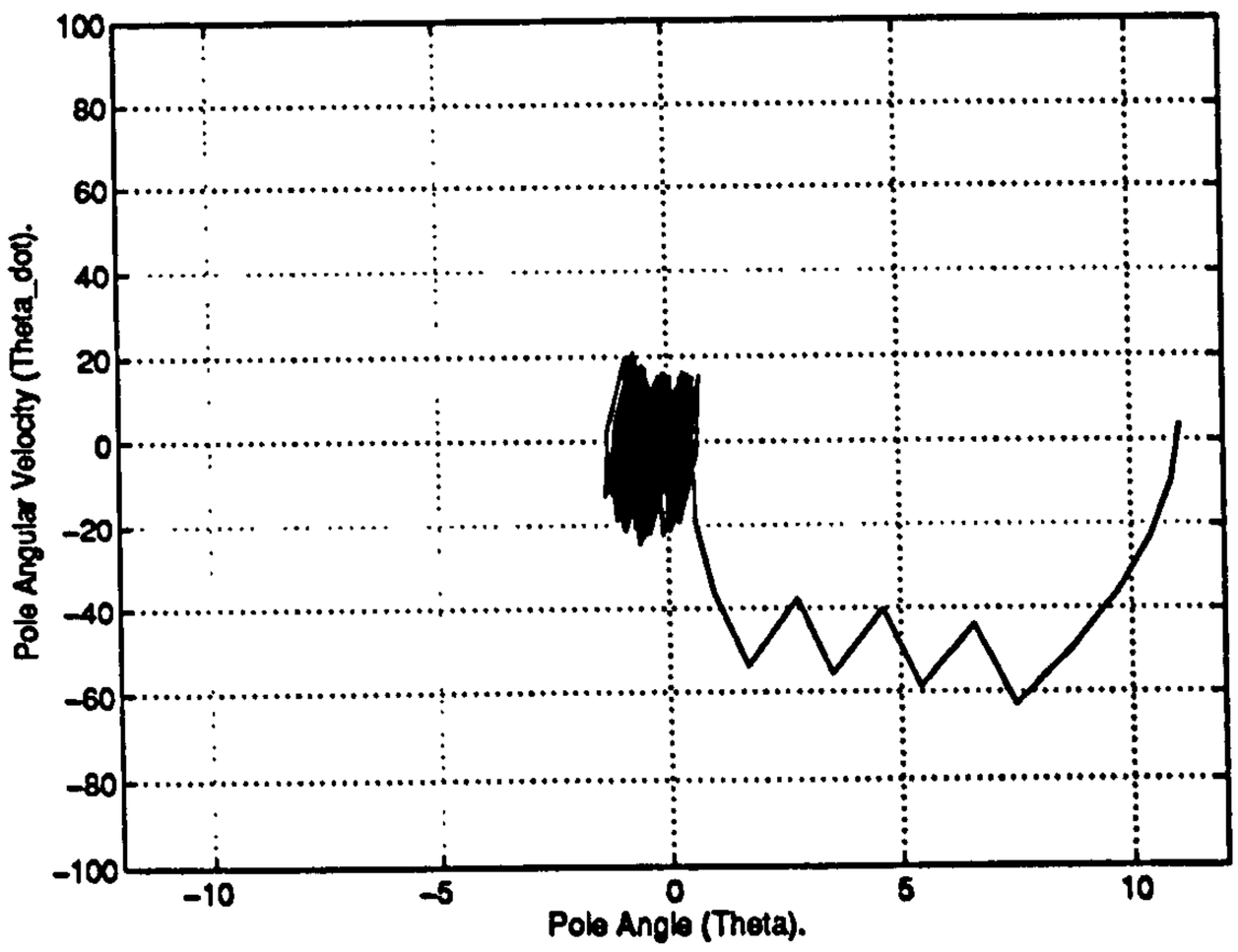


Figure 5.13 Phase plane plot for the 11 degree initial condition FUZBOX run. Note how the angle is brought into the stable region in the centre of the phase plane.

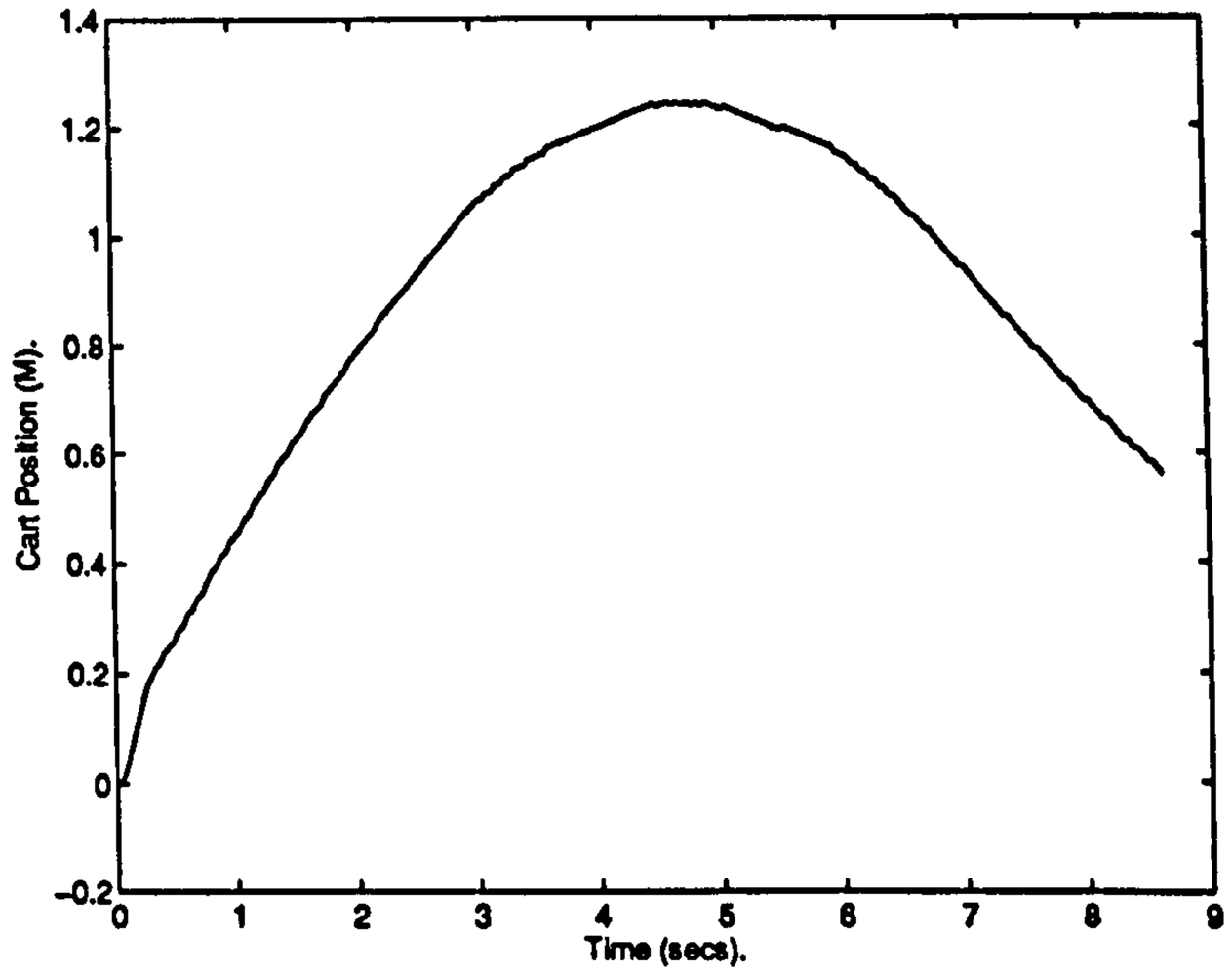


Figure 5.14 cart displacement plot for the 11 degree initial condition FUZBOX run. Note the significant move away from the origin as the pole angle is corrected. The large displacement is then corrected.

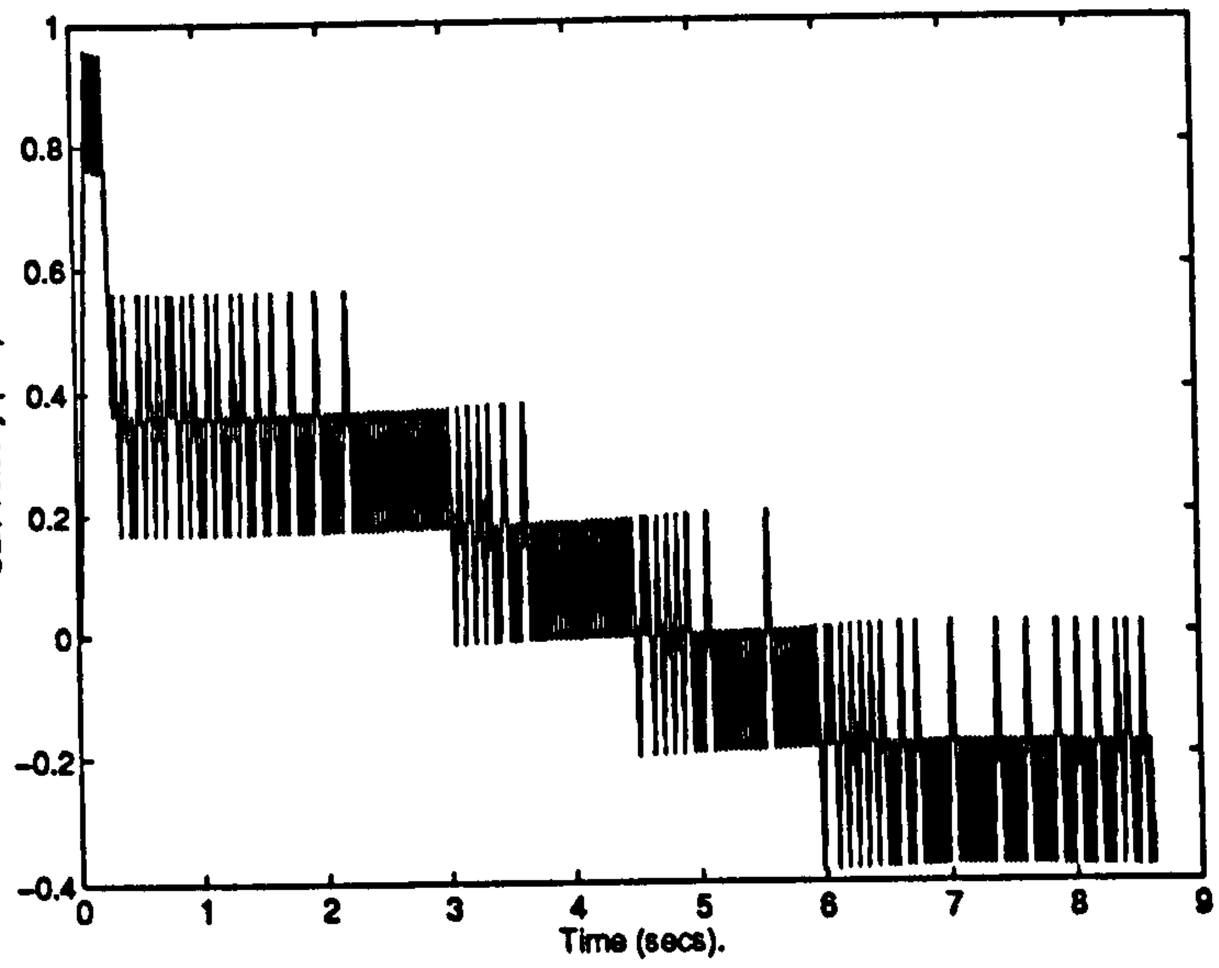


Figure 5.15 cart velocity plot for the 11 degree initial condition FUZBOX run.

The pole angle evolution is shown in Figure 5.16 where there is an initial rapid compensation forcing the pole towards zero followed by oscillation between zero degrees and -2 degrees for about six seconds.

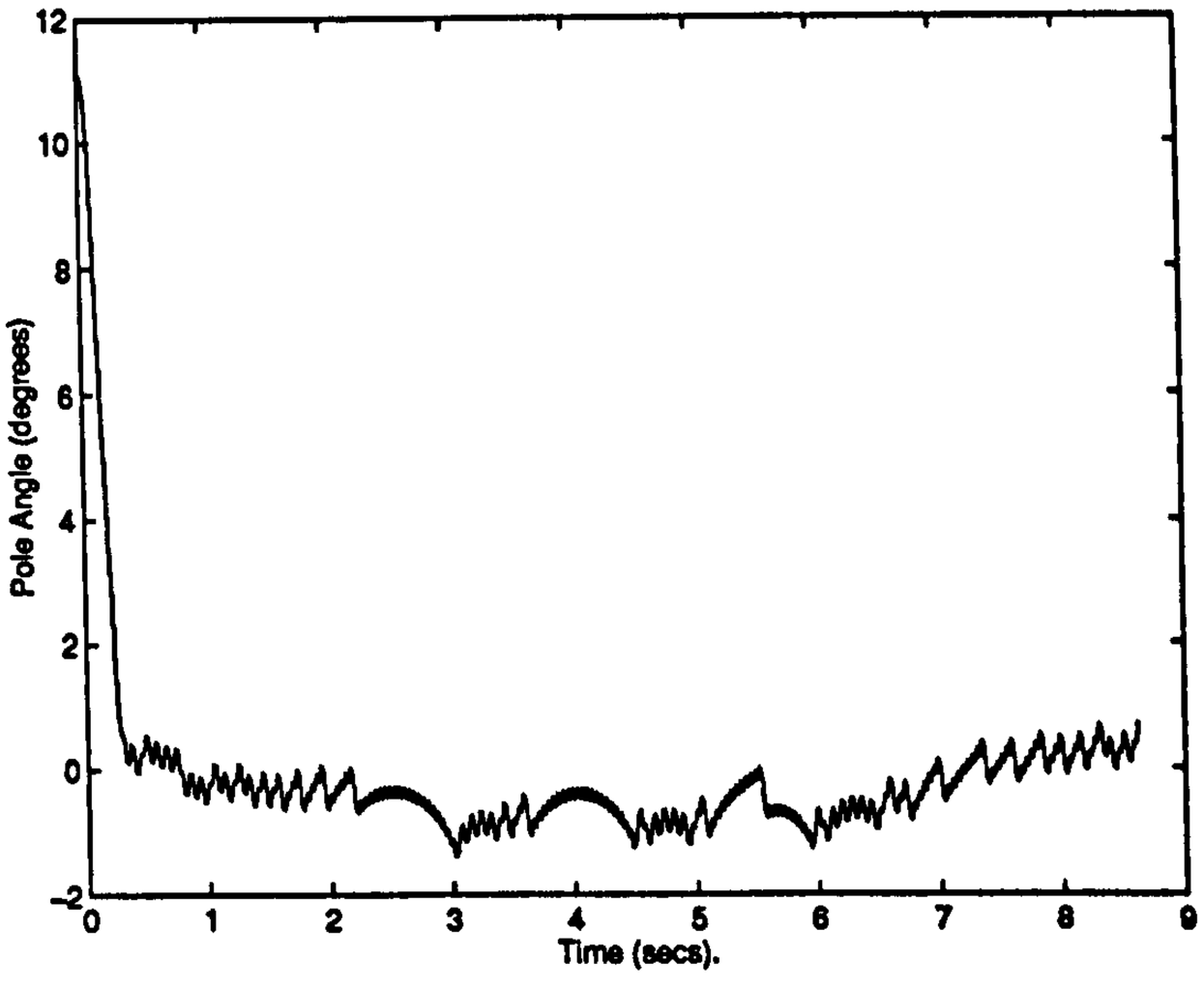


Figure 5.16 pole angle plot for the 11 degree FUZBOX run.

The pole velocity is shown in Figure 5.17. There is an initial negative pole velocity as expected followed by rapid oscillation of velocity around zero. The oscillatory behaviour around zero is predominantly positive as the neurocontroller compensates for the cart displacement.

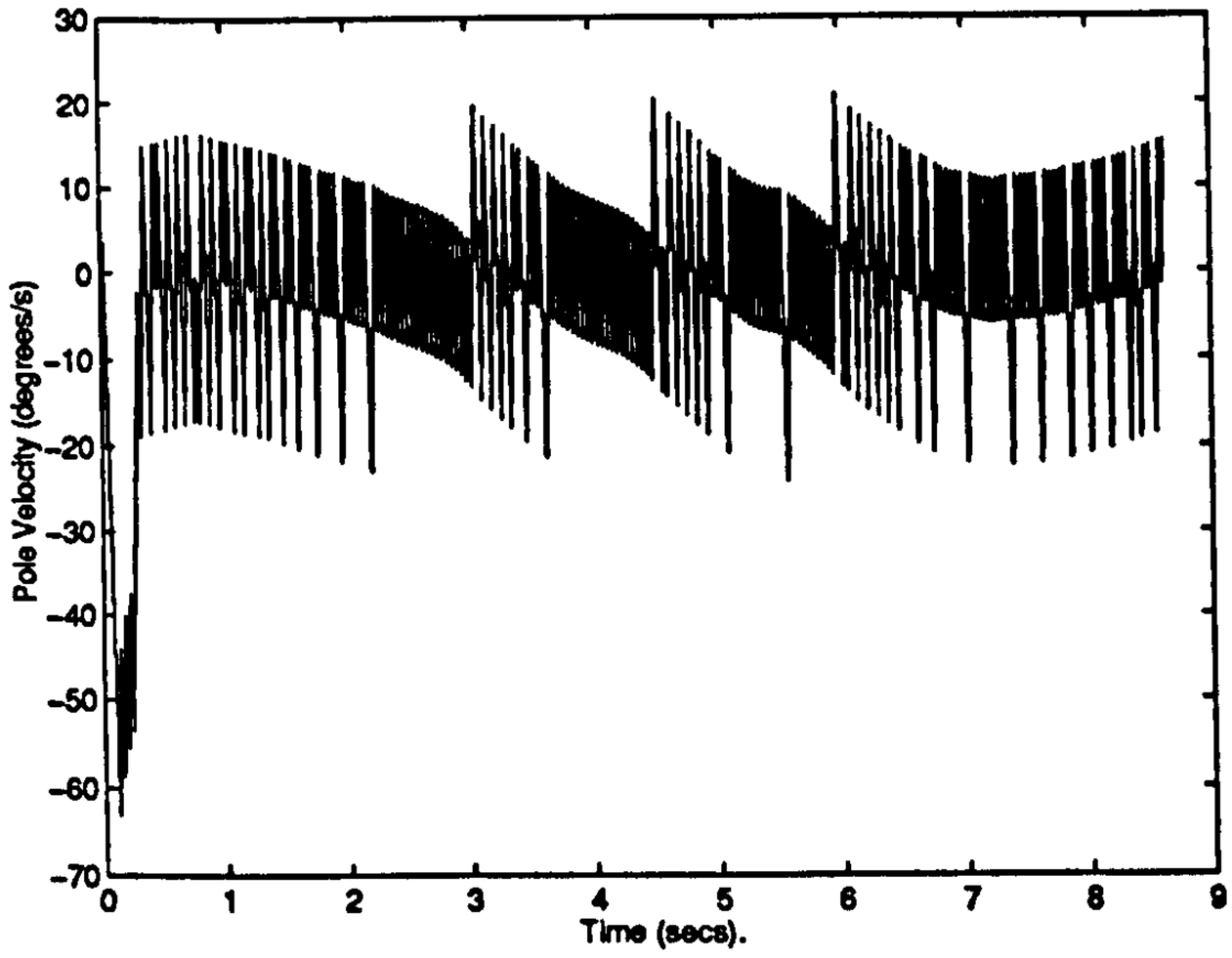


Figure 5.17 pole angular velocity plot for the 11 degree initial condition FUZBOX run.

5.5.7 An Example Rule-Base

A FUZBOX simulation was carried out using the following parameters: α=1000, β=0.5, δ=0.9, γ=0.95 and λ=0.8 which were identical to those used in the original BSA implementation. This simulation corresponds to the first entry of Table 5.1.

Following the simulation, the 14 most important rules—in terms of relative rule strength—were selected out of a total of 152 generated by FUZBOX. These 14 rules accounted for 89.6 % of the total rule strength of unity. Figure 5.18 shows the cumulative rule strength with respect to the rule rank. Table 5.3 shows the form of these rules together with the associated relative rule strengths (RRS). RRS1 is the relative rule strength of the 14 rules when they were chosen from a run of 32 trials, that is, one trial following convergence. RRS2 is the relative rule strength obtained after 5 trials of a run using the 14 rules *a priori* to prime the rule-base. Eleven new rules were generated for this run of five trials. The maximum RRS2 value of the newly generated rules was 0.02 or 2%. The total relative rule strength attributable to the 11 new rules was 5.3% which means that the total rule strength had increased slightly from 89.6% to 94.7% indicating that little information had been added to the *a priori* rule-base taken from the first run.

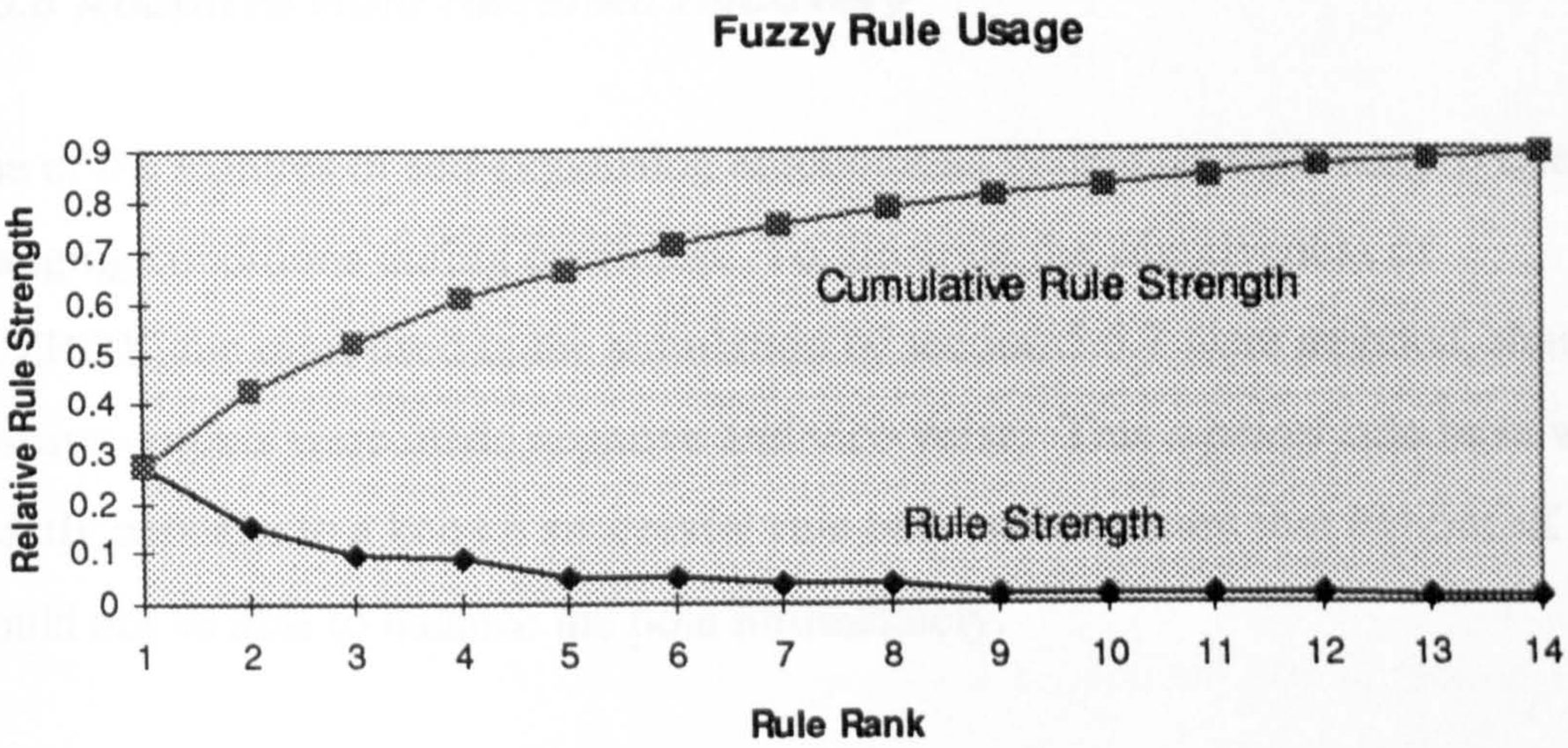


Figure 4.2.3.10 A plot showing the absolute and cumulative relative rule strength values for a ranked set of rules comprising a successful FUZBOX rule-base.

x	xdot	theta	thetado t	output	RRS(1)	RRS(2)
NZ	NZ	NZ	NZ	POSITIVE	0.274062	0.254235
SN	NZ	NZ	NZ	NEGATIVE	0.154640	0.253995
NZ	NZ	NZ	SN	NEGATIVE	0.093908	0.082556
NZ	NZ	NZ	SP	POSITIVE	0.090831	0.080343
SN	NZ	NZ	SP	POSITIVE	0.050219	0.080159
SN	NZ	NZ	SN	NEGATIVE	0.049607	0.082504
NZ	NZ	SN	NZ	NEGATIVE	0.039465	0.016461
NZ	NZ	SP	NZ	POSITIVE	0.035519	0.011798
NZ	SN	NZ	SP	POSITIVE	0.020934	0.019486
NZ	SP	NZ	SN	POSITIVE	0.019701	0.020025
NZ	NZ	SP	SP	POSITIVE	0.018708	0.007265
NZ	NZ	SN	SN	NEGATIVE	0.017498	0.009705
SN	NZ	SP	NZ	POSITIVE	0.015824	0.016116
SN	NZ	SN	NZ	NEGATIVE	0.014594	0.012024

Table 5.3. A rule-base consisting of fourteen rules generated using FUZBOX These rules were sufficient to solve the cart-pole problem when the cart-pole system is started from near the origin. RRS(1) is the original relative rule strength of the rules; this figure was used to select these 14 rules from the original 152. RRS(2) is the new relative rule strength obtained when using the rule as part of a 14 rule rule-base

5.5.8 Adaptive Rule Reversal Recovery

One of the features of self-organising autonomous systems is their adaptiveness to changing conditions during operation. To illustrate the adaptiveness of FUZBOX, the rules used in the simulation of section 5.5.7 were negated, that is, positive outputs were made negative and vice versa. This *a priori* rule base was exactly opposite to a known successful rule base which meant that FUZBOX would not be able to balance the pole immediately.

FUZBOX required a total of 98 trials before converging to a consistent balancing time of 10000 seconds. A total of 194 rules were generated, although, as illustrated in the previous simulation, only a small fraction of this total number might be required for balancing.

This simulation demonstrates clearly that the self-organising nature of FUZBOX allows on-line recovery from changes in operating conditions; indeed, even changes as drastic as complete reversal of sucessful rules. The loss of control during the recovery period may be unacceptable in practical terms but would it be reasonable to expect any system to maintain control immediately following an inversion of its rule base? More pertinent though is the ability of FUZBOX to re-establish control.

5.5.9 Pruning

The rule base of section 5.5.7 consisted of 14 rules selected from a total of 152. Pruning, in this case, was done by hand. It is not inconceivable that this may be carried out automatically. A simple method would be similar to that of the EUCART-BSA hybrid with pruning in that the 'weakest' nodes would be removed periodically if the relative rule strength drops below a given threshold. However, the same criticisms hold as for the EUCART-based system.

5.6 The EUCART+BSA Hybrid Revisited

5.6.1 Distribution of Information

The FUZBOX architecture shows clearly that distribution of the information throughout the ASE / ACE modules is possible. Distributing the EUCART-BSA hybrid requires a type of fuzzification of category membership. A method similar to that of Zhang and Grant (1992) can be applied to EUCART by defining a *relative node activation* (RNA) in terms of the category centre. Using the EUCART choice function defined by

$$T_j^E(\mathbf{I}) = 1 - \frac{\|\mathbf{c}_j - \mathbf{I}\|}{\sqrt{M}}$$

a relative node activation can be defined as

$$\mu_j(\mathbf{I}) = \frac{T_j^E(\mathbf{I})}{\sum_{k=1}^{N} T_v^E(\mathbf{I})}$$

to give a category membership function.

The EUCART choice function presents a natural fuzzification of the input space because it is constrained on the interval [0,1] as no input and exemplar vectors can be more than $\sqrt{M}$ apart. The choice function values then have to be normalised to give a normalised fuzzification for a weighted distribution of information. Here, the weighting of distributed information is directly proportional to the radial distance between a given input and all category centres; node "age" is not taken into account.

The EUCART category centres—defined by the category extent markers—may be replaced by centroids as detailed in section 4.2.14.

Preliminary experiments using the conditions of simulation 2 of section 4.3.4 were carried out using both the centre and centroid prototypes. A single run using random number seed 1 for both prototypes gave the results of Table 5.5

method	trials	nodes
EUCART centre	235	360
centroid	313	392

Table 5.5 The results of a single run of a distributed EUCART-BSA hybrid using two different methods of distribution of information. EUCART centre denotes the method of fuzzification (and hence distribution of information) with respect to the category node centre determined by the category extent markers. Centroid denotes fuzzification using the centroid of a given category node.

From the preliminary results of Table 5.5 it is clear that for this single run the distributed EUCART-BSA hybrid learns using either the original EUCART centre or the centroid. However, for this single case, the distributed hybrid learns more slowly than any of the previous methods explored in this thesis. This is counterintuitive and requires a statistical study of a set of runs to see if this method is slower on average. If such is the case, then further research is required to establish why.

5.6.2 Distribution of Information: A Conclusion

It has been established that the non-distributed version of the EUCART-BSA hybrid succeeds in learning a meaningful control mapping by self-organising a representation of the state-space. It has also been established that, for a fuzzy decoder with fixed overlap, distributed ASE and ACE modules are possible. Furthermore, learning is much more rapid with the distributed system when compared with a winner-takes-all system (boxes) with the same crisp boundaries as the intersections of the fuzzy sets used for the distributed version.

The logical next step is to distribute the EUCART-BSA hybrid system. Preliminary results show that this is not straightforward and learning may in fact be made more difficult by distributing the EUCART-BSA hybrid. One possible cause of the problems is that the category membership functions used to obtain

the distribution weightings are always changing because the hyperspherical category boundaries are dynamic. This must be investigated further.

In both the EUCART and fuzzy decoders, the spectre of the *curse of dimensionality* is raised. The number of possible nodes or rules rises dramatically with the increasing dimensionality of input space. For example, for a modest ten-dimensional system using five fuzzy sets for each dimension of the input space, there are 100,000 possible rules. Such systems rapidly become untenable as the number of input variables is increased. Clearly a more compact or efficient representation is required.

For the EUCART (fuzzy) system, nodes (rules) are added as required which helps to reduce storage and computational overheads. Furthermore, as discussed, relevance pruning may provide a viable option for reducing complexity and aiding efficient storage.

Another possible extension has already mentioned, that of "splitting" and "lumping" (Michie and Chambers, 1968b). The "lumping" together of sets of rules or nodes of low generality to give rules or nodes of higher generality will both increase generalisation and reduce storage requirements. In the case of FUZBOX, increased rule generality will aid comprehension of the rule-base by a user.

Both the EUCART-BSA hybrid and FUZBOX need to be extended to multi-valued or continuous outputs if their usefulness and generality are to be increased. For the multi-valued case, probabilities may be associated with the output values and modified according to success or failure. The trade-off for the added flexibility will undoubtedly be in terms of learning time which will increase with the number of additional outputs.

Chapter 6 General Discussion and Conclusions

6.1 General Discussion

Three novel architectures, namely PROBART, EUCART and FUZBOX, have been introduced in this thesis and have been applied to mapping and control problems with some success. All three novel architectures have addressed some of the shortcomings of alternative architectures.

Both the EUCART-BSA hybrid and FUZBOX demonstrate the possibility of autonomous neurocontrollers which can be "plugged in" and left to learn *in situ*. To avoid potentially catastrophic results, a human controller or other device could be used to maintain control beyond recoverable "failure" limits. Much work is still to be done concerning robustness, but the simulation results provide inspiration for further work.

One of the main points which has arisen from this work is the notion of compromise. Apparently, there is no universal neural network which is equally suitable for solving all types of problem. A number of competing constraints become apparent when considering neural architectures and problem domains. It might be said that there are no solutions, only insight into the nature of the automated learning task. The following list gives some of the areas of conflict:

- stability vs. plasticity
- look-up vs. generalisation
- discrete vs. distributed knowledge representation
- Off-line vs. on-line (causal) learning
- exploration vs. exploitation

One area of concern for machine learning is the efficient (optimal) use of information. Although learning an optimal strategy may be the desired goal, the learning of that strategy may be sub-optimal. The significance of sub-optimality depends on its degree.

A major criticism aimed at reinforcement learning is the invariably long period of learning taken to acquire a behavioural (control) strategy. The lack of intermediate supervisory signals (supervised learning) has the drawback of making learning relatively slow whilst behavioural action pairs are constructed along the lines of trial and error.

The lack of model-based processing leads to inefficient use of information. Stochastic search of the problem space replaces the strategy of repeatedly estimating and refining parameters. This may be fine for small problems but will become untenable rapidly with increasing problem complexity.

Failure-driven learning is not necessarily an optimal way of acquiring an optimal strategy. The lack of meta-processing and evaluation makes reinforcement learning a brute-force method. It would not be desirable for a mobile autonomous agent to bump into an obstacle a large number of times before changing its strategy. Inclusion of meta-processing algorithms to switch intelligently between strategies or *a priori* information may provide an alternative to purely failure-driven learning. Reinforcement learning could be used to fine-tune coarse strategies acquired more rapidly through other learning methods.

A more general criticism can be aimed not at the *length* of time taken to acquire a behavioural strategy using reinforcement learning, but that learning time is used at all to measure performance. The most frequently used measure of performance for connectionist and genetic algorithms is the *learning rate* (Sammut and Crib, 1990). The learning rate is a measure of how many trials are required before performance is adequate. Sammut and Crib (1990) criticise the use of the learning rate of a system as a measure of learning efficiency; they maintain that it is not necessarily the best measure of performance and that it can be misleading at times.

A system may learn quickly but may only be applicable to a small operating region; this raises the question of robustness. The pole balancing problem was put forward to illustrate the point.

Sammut and Crib looked at a variety of learning algorithms such as the BSA reinforcement learning system and found that specific learning—pertaining to a limited region of state space—was not transferable, and that rapid learning implied that learning was specific. The investigation revealed the trade-off between rapid learning and generalisation; this is reminiscent of Michie and Chambers' idea of *exploration vs. exploitation* (1968b). Rapid learning rates do not lead to general solutions.

A bang-bang control system, such as the ASE / ACE system, which uses a quantised representation of state space can be treated as a finite state automation. An adequate control strategy for the cart-pole problem is represented by a cycle between states. Figure 6.1 shows a cycle between states with binary outputs (bang-bang control)

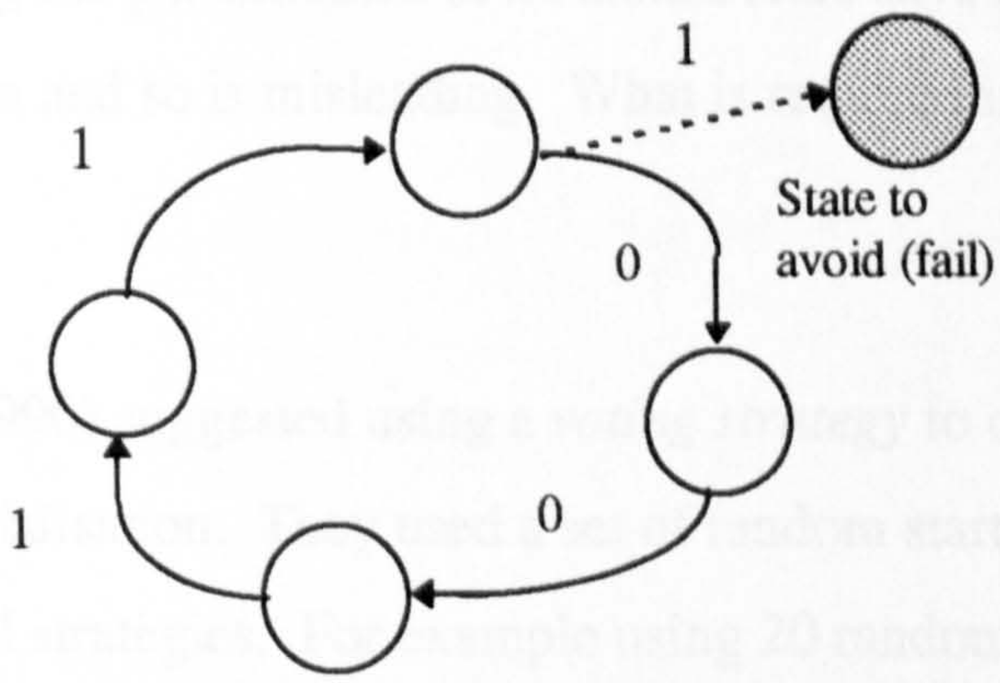


Figure 6.1 An absorbing cycle representing an adequate control strategy. Once such a cycle between states has been entered, control will be maintained unless disturbances force the state-space trajectory out of this operating region.

When a neurocontroller learns a control strategy quickly it means that a cycle has been discovered within a short time; it also implies that the rest of the finite state automaton graph has not been explored. There are many potential solutions to the cart-pole problem. Some of the cyclic solutions have state nodes in common owing to the variability within a given state node; this variability within a node

stems from the fact that a state node represents a *set* or *cluster* of states which may lead to different regions of state space, even for the same output (See Figure 6.2)

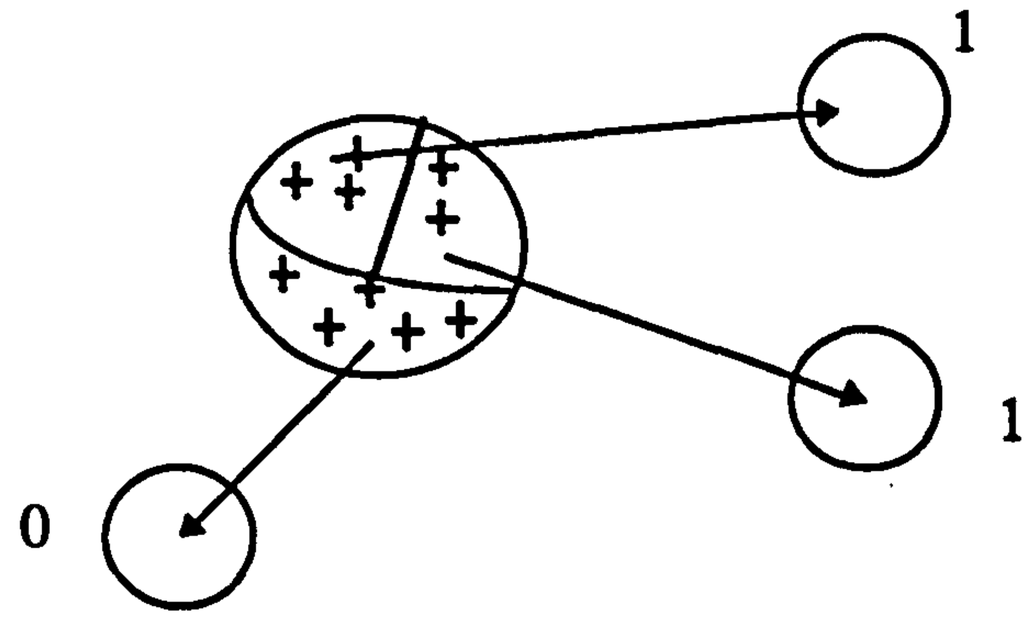


Figure 6.2 Variation within a region of state-space represented by a single node. The same output may lead to state-nodes with different outputs.

Starting a system with different initial conditions from those used for training very often does not balance the pendulum again. In effect, the system only learns to balance the pendulum from favourable conditions and some states are never experienced and, thus, never learned. Thus using learning rate as the sole criterion for assessing the performance of an architecture says nothing about the quality of the solution and so is misleading. What is required is a robust control strategy.

Sammut and Crib (1990) suggested using a *voting strategy* to counteract the problem of overspecialisation. They used a set of random starting points and 'froze' the successful strategies. For example using 20 random starting points for 832 trials of a single experiment resulted in 100 successful trials. Using a voting strategy to select the most successful strategies amongst the 100 successful trials gave a new set of boxes. The new set of boxes resulted in a score of 20/20 successful trials when tested on each of 20 new random starting points. This is compared with a near zero score for learning systems trained using a single long run with a single starting point.

The use of random starting points when training may increase learning time but this may be acceptable when taking into account the increase in robustness.

Random starting points will lead to a greater proliferation in EUCART nodes used *en passant* which will not contribute greatly to any given strategy. However, relevance pruning will remove these providing that the pruning strategy is *contextual* as opposed to purely usage-based. Contextual pruning would only permit the removal of nodes in a given context or strategy. Nodes which contributed to a successful control strategy for a given starting point would be rendered "immune" from pruning when either the starting point or plant operating conditions changed. This immunity would prevent pruning from favouring a single strategy. Previously useful nodes which are no longer used in the current strategy would have a falling relative usage count. Without immunity, these nodes would become unimportant and would be removed when the usage count fell below a given threshold if operating conditions warranted a change in strategy. Contextual pruning provides a direction for future work.

The cart-pole problem provides a useful dynamical system for the development and testing of putative neurocontrollers. However, difficulties arise in comparing the performances of systems developed by various authors. Geva and Sitte (1993) highlight two main problems:

- a lack of an agreed experimental protocol upon which to base benchmark tests, and
- a lack of an agreed reporting standard for results.

In addition to this, not all experimental details are reported which makes replication, and consequently comparison, difficult (Randall, Thorne, and Wild, 1994). Geva and Sitte (1993) also expressed concern that the cart-pole problem is trivial if zero initial conditions are used. Randal, Thorne and Wild (1994) suggest:

- a standard set of parameters for the simulation including a difficult starting condition (e.g. cart displacement +1m, cart velocity +1m/s, pole angle 1rad, pole angular velocity 0.1789 rad/s)

- a set of standard assessment criteria, e.g.

    -balancing time

    -centering / oscillation (RMS values)

For the case of the oscillation criterion, the findings of Randal, Thorne and Wild concur with those of Geva and Sitte (1993) in that for box-based systems, large oscillations (in centering of pole and cart) are caused by course partitioning of state-space. The suggested reason is that the state-space trajectories remain within the same state-space region (box) for more than one time-step. Consequently, the same control action is issued for a number of consecutive time-steps giving large cart and pole displacements. The situation is likely to be repeated when the trajectory crosses a box boundary. The oscillation problem indicates that variable granularity state-space partitioning may provide better performance. However, a number of practical problems need to be overcome including the difficulty in specifying a "granularity adjustment" algorithm or procedure. This is a meta-control problem.

One possibility may be to introduce match-tracking in EUCART to "split" nodes by including smaller nodes in critical or boundary regions. The increased storage overheads will possibly be mitigated by the judicious use of pruning to reduce coverage of less important regions. The viability of pruning has already been demonstrated although more work has to be done to make it more "intelligent".

A recent paper by Lin and Lin (1996) proposes a network, RFALCON, which ties together a number of ideas presented in this thesis. The *R*einforcement *Fuzzy* *A*daptive *L*earning *Co*ntroller *N*etwork associates input patterns with output patterns according to a reinforcement schedule. It is constructed from two multilayer feedforward networks (FALCONs) known as the *actor* the *critic* respectively. Associated with RFALCON is an ART-based algorithm which is used to cluster the input and output spaces. The RFALCON system operates on-line using a two-phase process:

- *structure learning*, which uses ART to self-organise the input and output spaces, and,
- *parameter learning*, which uses a form of backpropagation to tune the input and output fuzzy membership functions.

Nodes representing linguistic terms or fuzzy rules can be added as required. Results for the cart-pole problem show an average convergence time of 15 trials for 5 runs. An average of 10 fuzzy rules were created. A run was terminated at 50 000 time-steps (1000 seconds) if failure did not occur before this time.

Although a fuzzy-logic based neurocontroller (FUZBOX) has been introduced as a novel functional learning system in this thesis, the aim was to demonstrate the feasibility of a distributed representation of state-space. Decoupling the distributed representation feasibility problem from EUCART-BSA dynamics made it easier to deal with. The main aim to distribute the EUCART-BSA hybrid still remains. The idea of tuning the membership functions from both the work of Berengii and Khedar (1992) and Lin and Lin (1996) by using a modified gradient descent method may be useful in developing an automated variable granularity system.

Reinforcement learning has only been applied to a single problem within this thesis. This was done deliberately so that operational and architectural issues could be explored with respect to a well-known problem. It is envisaged that once architectural issues have been resolved, the resulting architectures would be applied to other problem domains especially in "real-world" control. Outstanding issues include:

- contextual pruning,
- robustness,
- more efficient learning (distribution of information?),
- multi-valued or continuous outputs,
- a principled approach including a well-founded theory,

- variable granularity of state-space coverage (including automatic tuning of granularity), and
- meta-control including intelligent strategy shifting

For the FUZBOX, architecture in particular, areas of future work include:

- automatic tuning of fuzzy membership functions, and
- "splitting and lumping" to give more general rules using a fuzzy analogue of Karnaugh mapping (e.g. Bannister and Whitehead, 1983):

The ball and beam system (Lin and Lin, 1996) would provide a possible alternative benchmark problem.

Most, if not all, future improvements would be in the direction of increased autonomy and more intelligent behaviour. This, of course, is characterised by the concept of meta-control. Tolle and Ursü (1992) give further insight into the notion of meta-control by dividing the concept of "machine intelligence" into two levels: *Microintelligence:* characterised by

- computing units
- local processing
- input /output mappings
- functional groups of processing elements: networks
- generalisation and recall, and

Macrointelligence: characterised by

- goal orientated use of microintelligence
- co-ordination of functional groups
- multiple levels of processing (e.g. Churchland and Sejnowski, 1992)
- subroutines
- tokens-atoms of the next level.

One possible direction of future development is that neurocontrol will move away from single reactive networks towards a form of distributed macrointelligence (meta-control) operating within multiply re-entrant networks (Edelman, 1989) capable of tracking and intelligently pre-empting change. Such multi-component networks would be capable of learning through experience and constantly adapting to novel conditions. They would not be limited to a small pre-determined behavioural repertoire or constrained to optimise a single strategy. Although intelligence is difficult to define or even to characterise, it certainly does not mean doggedly persisting with a learning strategy when even a little higher-level (meta-) processing would reveal that the strategy was hopeless or needed adjusting.

It may be objected that speculation proposing such ill-defined intelligent systems is far removed from current capabilities such as those detailed here. Indeed, it may be that nothing short of a paradigm shift is required to change the emphasis from more conservative approaches to learning to more organic and diffuse approaches emphasising emergent properties. There is evidence that this is happening (e.g Langton, 1989). The development of intelligent autonomous systems may seem unrealistic and yet some of the simplest living organisms exhibit adaptive behaviour.

Arguably, the greatest source of inspiration is the biological world. Millions of years of evolution and untold numbers of organisms have constituted a learning experiment of unprecedented levels. Examples of successful intelligent adaptation abound in a harsh testing ground for learning algorithms. A "reverse-engineering" or analytic approach applied to the living world may seem antithetical to a bottom-up synthetic approach to artificial learning but it is not. Indeed significant advances in machine learning theory may possibly result from the synergistic combination of these complementary approaches.

6.2 Conclusions

In this thesis, a number of drawbacks of current approaches and fundamental problems behind these drawbacks have been presented. These include:

- The lack of one-to-many mapping in ARTMAP and fuzzy ARTMAP;

- The tracking of noise by fuzzy ARTMAP, i.e. noise is treated as novelty and learned;

- The minimal generalisation capabilities of ARTMAP and fuzzy ARTMAP, i.e. if an input does not fall within an existing category then an estimated output cannot be generated;

- The requirement for a fixed *a priori* structuring of state-space in the BSA implementation of reinforcement learning;

- The use of winner-takes-all dynamics with discrete boxes;

- The allocation of information storage capacity *en masse* regardless of use;

- The long learning times required for reinforcement learning.

The above list is not meant to give the impression that the architectures featured here are without merit. In particular, ARTMAP, fuzzy ARTMAP and the ASE /ACE systems have proved to be successful approaches to many problems involved in learning. However, further improvements can be made. Novel architectures and proposed modifications to architectures include:

- A novel mapping architecture, PROBART, base upon adaptive resonance theory

- Establishment of one-to-many mappings and noise suppression in PROBART;

- A proposed method of distributing information in PROBART to increase generalisation to unseen inputs;

- A novel self-organising architecture, EUCART, used in the self-organising partitioning of state-space for control applications;

- An autonomous hybrid architecture, EUCART+BSA, which is based upon two current areas of research, ART and reinforcement learning;

- A fuzzy-logic and reinforcement-learning based autonomous system, FUZBOX, capable of generating rules from experience;

- Both implemented and proposed mechanisms for dynamic allocation of storage via incremental learning and pruning mechanisms.

One particular aspect of importance which has arisen from this work is that of controller robustness. Even from consideration of a single simulated control example, it is clear that both the original BSA neurocontroller and the novel architectures indicate the possibility of autonomous control systems requiring minimal supervision. However, the results from both the replication studies and the novel architecture studies suggest that performance is sensitive to changes in a number of parameters including initial conditions and the introduction of noise to drive the stochastic search. The variability of convergence times for the same initial conditions is evidence of sensitivity. The resulting neurocontrollers will be robust to differing degrees depending upon the final extent of neurocontroller experience following convergence.

Arguably the most fruitful next step will be to increase the experience of the neurocontroller by exposing it to different control conditions through direct manipulation of the simulation. This increase in experience will entail a

commensurate increase in storage requirements which will require modulation using intelligent pruning. Robustness is an issue which must be dealt with if realistic neurocontrollers are to be developed.

The contents of this thesis establish some new results and provide an indication of possible future directions. It is not so much that the architectures discussed are end-products of a problem-orientated design process but are rather by-products of an exploration of the issues involved in machine learning. This exploration is open-ended and numerous future modifications have been proposed. Adaptive Resonance Theory and reinforcement learning are two biologically and psychologically inspired theories which continue to grow in importance and shed light on some of the age-old problems in learning theory, both biological and artificial.

References

Ahalt, S. C., Krishnamurthy, A. K. Prakoon, C. and Melton, D.E, (1990), Competitive Learning Algorithms for Vector Quantisation, *Neural Networks*, **3**, 277-290.

Albus, J. S. (1975a), A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC), *Trans. ASME. Jnl. Dyn. Sys. Meas. and Control*. **63** (3), 220-227

Albus, J. S. (1975b), Data Storage in the Cerebellar Model Articulation Controller (CMAC), *Trans. ASME. Jnl. Dyn. Sys. Meas. and Control*. **63** (3), 228 -233

Albus, J. A Model of the Brain for Robot Control (1979), parts I, II and III, *BYTE magazine*, June, July and August issues 10-34,. 55-95, 66-80. BYTE Publications Inc.

de Alche-Buc, F. Andres, V. and Nadal, J-P (1992), Learning Fuzzy Control Rules with a Fuzzy Neural Network, in *Artificial Networks 2* (Eds.) Alexander, I and Taylor, J. Elsevier Science Publishers, 715-719

Anderson, C. W., (1989), Learning to Control an Inverted Pendulum using Neural Networks, *IEEE Control Systems Magazine*, April 31-36

Anderson, C. W. (1993), Q-Learning with Hidden-Unit Restarting *Advances in Neural Information Processing Systems 5*

Anderson, J. R. (1995), *Learning and Memory: an Integrated Approach*, John Wiley and Sons, NY.

Arbib, M., A. (1987), *Brains, Machines and Mathematics* (2nd Edn.) Springer-Verlag NY.

Åström, K. J.(1995) Adaptive Control: General Methodology In *The Handbook of Brain Theory and Neural Networks* Arbib, M. A. (ed.) MIT Cambridge MA.

Bailey, N. T. J. (1964), *The Elements of Stochastic Processes with Applications to the Natural Sciences*. Wiley

Banks, S. P. (1986), *Control Systems Engineering* Prentice-Hall international Englewood cliffs NJ

Bannister, B. R. and Whitehead, D. G. (1983), *Fundamentals of Modern Digital Systems* The Macmillan Press Ltd. London

Barker, L. M, (1994) *Learning and Behavior: a Psychological Perspective* Macmillan College Publishing Company, New York.

Barto, A. G. (1992), Reinforcement Learning and Adaptive critic Methods, in White, D. A. and Sofge, D. A. (Eds.) *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*.

Barto and Anandan, (1985), Pattern-Recognizing Stochastic Learning Automata *IEEE Transactions on Systems, Man and Cybernetics* **15** (3) 360-375

Barto, A. G. ,Bradtke, S. J. and Singh, S.P. Learning to Act Using Real-time dynamic Programming. *Artificial Intelligence* Special Volume **72** (1): 81-138, 1995.

Barto, A. G. and Sutton, R. S. (1982), Simulation of Anticipatory Responses in Classical Conditioning by a Neuron-Like Adaptive Element, *Behavioral Brain Research*, **4**, 221-235.

Barto, A. G. Sutton, R. S., and Anderson, C. W. (1983), Neuronlike Adaptive Elements that can Solve Difficult Learning Control Problems *IEEE Trans. Syst. Man. Cybern.* Vol. SMC-13, 834-846.

Bengio, Y., Simard, P. and Frasconi, P. (1994), Learning Long-Term Dependencies with Gradient Descent is Difficult *IEEE Trans. On Neural Networks*, **5** (2), 157-166

Berenji, H. R. and Khedkar, P. (1992), Learning and Tuning Fuzzy Logic Controllers Through Reinforcements. *IEEE Trans. On Neural Networks*, **3** (5), 724-740

Best, J. B. (1992), *Cognitive Psychology* (3rd Edn.) West Publishing company St Paul MN.

Bianchini, M, Gori, M and Maggini, M (1994), On the Problem of Local Minima in Recurrent Neural Networks *IEEE Trans. On Neural Networks*, **5** (2), 167-177

Bishop, C. (1995) *Neural Networks for Pattern Recognition* Clarendon Press Oxford

Broomhead, D., and Lowe, D. (1988), Multivariable Function Interpolation and Adaptive Networks. *Complex Systems*, **2**, 321-355

Brown, M. and Harris, C. (1994), *Neurofuzzy Adaptive Modelling and Control,* Prentice Hall Series in Systems and Control engineering, Prentice Hall, New York.

Buckley, J. J. and Hayashi, Y. (1993), *Proceedings of the World Congress on Neural Networks (INNS)*, Portland, Oregon **2**, 92-96

Budnick, F. S. (1988), *Applied Mathematics for Business, Economics and the Social Sciences* (3rd Edn.) McGraw Hill

Carlson, N. R. (1994), *Physiology of Behavior*, Allyn and Bacon Needham Heights MA.

Carpenter, G. A. and Grossberg, S. (1986), Absolutely Stable Learning of Recognition Codes by a Self-organizing Neural Network, In J. Denker (Ed.) *AIP conference Proceedings 151: Neural Networks for Computing* 77-85 New York AIP.

Carpenter, G. A., & Grossberg, S, (1987a), A Massively Parallel Architecture for a Self-organizing Neural Pattern Recognition Machine, *Computer Vision, Graphics, and Image Processing*, **37**, 54-115.

Carpenter, G. A., & Grossberg, S, (1987b), ART 2: Self-organisation of Stable Category Recognition Codes for Analog Input Patterns. *Applied Optics*, **26**, 4919-4930.

Carpenter, G. A., & Grossberg, S, (1989), ART 3: Hierarchical Search Using Chemical Transmitters in Self-Organizing Pattern Recognition Architectures, *Neural Networks*, **3**, 129-152.

Carpenter, G, and Grossberg, S. (1992), A Self-Organizing Neural Network for Supervised Learning, Recognition, and Prediction *IEEE Communications Magazine* **30** (9) 38-49

Carpenter, G and Grossberg, S. (1994), Fuzzy ARTMAP: A Synthesis of Neural Networks and Fuzzy Logic for Supervised Categorization and Nonstationary Prediction *in Fuzzy Sets, Neural Networks, and Soft Computing*, Yager, R. R. and Zadeh, L. A. (Eds.) VNR New York.

Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., & Rosen, D. B.,(1992), Fuzzy ARTMAP: A Neural Network Architecture for Incremental

Supervised Learning of Analog Multidimensional Maps, *IEEE Transactions on Neural Networks*, **3**, 698-712.

Carpenter, G. A., Grossberg, S. & Reynolds, J. H., (1991), ARTMAP: Supervised Real-time Learning and Classification of Nonstationary Data by a Self-organizing Neural Network. *Neural Networks*, **4**, 565-588.

Carpenter, G. A., Grossberg, S., & Rosen, D. B.,(1991), Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System. *Neural Networks*, **4**, 759-771.

Carpenter, G. A. and Tan, A-H. (1993), Rule Extraction, Fuzzy ARTMAP, and Medical Databases. In *Proceedings, World Congress on Neural Networks, Portland, OR*, Vol. I, 501-506 Lawrence Erlbaum Associates. hillsdale, NJ.

Churchland, P. S. and Sejnowski, T. J. (1992), *The Computational Brain* MIT Press Cambridge MA

Connell, M. E. and Utgoff, P. E. (1987) Learning to Control a Dynamic Physical System *Machine Learning and Knowledge Acquisition* 456-460

Cybenko, G. (1989), Approximation by Superpositions of a Sigmoidal Function. *Mathematical Control, Signals and Systems*, **2**, 303-314.

Daynan, P. and Hinton, G.E. (1993), Feudal Reinforcement Learning in Henson S.J., Cowan, J. D. and Giles, D. L. (Eds.) *Advances in Neural Information Processing 5*, Morgan Kaufmann, San Mateo, CA.

Edelman, G. M. (1989), *Neural Darwinism: The Theory of Neuronal Group Selection*, Oxford University Press, Oxford.

Elman, J. L. (1990), Finding Structure in Time *Cognitive Science* **14** 179-211

Fahlman, S. E. and Lebiere, C. (1990), The Cascade Correlation Learning Architecture in *Advances in Neural Information Processing Systems 2* Touretsky, D. (Ed) Morgan Kauffman 524-533

Freeman J. A. and Skapura, D. M. (1992), *Neural Networks: Algorithms, Applications and Programming Techniques* Addison-Wesley Inc.,. Reading Mass.

Friedland, B., (1987), Control System Design: An Introduction to State space Methods, McGraw-Hill Book Company, New York

Fritzke, B. (1991), Unsupervised Clustering with Growing Cell Structures *Proceedings of the IEEE International Joint conference on Neural networks* Vol. II 531-536

Fritzke, B. (1993), Kohonen Feature Maps and Growing Cell Structures- A Performance comparison in *Advances in Neural Information Processing Systems 5* Hanson, S. J., Cowan, J. D. and Giles, C. L. (Eds). Morgan Kauffman Publishers Inc. 123-130

Fritzke, B. (1994), Growing Cell Structures-A Self-Organizing Network for Unsupervised and Supervised Learning. *Neural Networks* Vol. 7 No. 9 1441-1460

Fu, L (1994), *Neural Networks in Computer Intelligence* McGraw-Hill Inc. New York.

Fujita, O. (1992), Optimization of the Hidden Unit Function in Feedforward Neural Networks *Neural Networks*, **5**, 755-764.

Funahashi, K-I. (1989), On the Approximate Realization of Continuous Mappings by Neural Networks. *Neural Networks*, **2**, 183-192.

Fung, C. F., Billings, S. A. and Luo, W. (1994) On-Line Supervised Adaptive Training Using Radial Basis Function Networks *Research Report No 554*, The University of Sheffield, U.K.

Gallant, S. (1993) *Neural network Learning and Expert Systems* MIT Press Cambridge MA

Gellatly, A (Ed.) (1986), *The Skilful Mind: An Introduction to Cognitive Psychology* Open University Press Milton Keynes

Geva, S. and Sitte, J. (1993), The Cart-pole Experiment as a Benchmark for Trainable Controllers, *IEEE Control Systems Magazine* **13** (5) 40-51

Girosi, F. and Poggio, T (1990), Networks and the Best Approximation Property. *Biological Cybernetics* **63**, 169-176.

Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning* Addison Wesley.

Grossberg, S. (1968) Some Nonlinear Networks Capable of Learning a Spatial Pattern of Arbitrary Complexity, *Applied Mathematics* **59**, 368-372

Grossberg, S (1976a); Adaptive Pattern Classification and Universal Recoding I: Parallel Development and Coding of Neural feature Detectors, *Biological Cybernetics*, **23** 121-134

Grossberg, S (1976b); On the Development of Feature Detectors in the Visual Cortex with Applications to Learning and Reaction-Diffusion Systems, *Biological Cybernetics*, **21** 145-159

Grossberg, S., (1980); How Does a Brain Build a Cognitive Code? *Psychological Review*, **1**, 1-51.

Grossberg, S (1987), Competitive Learning from Interactive Activation to Adaptive Resonance *Cognitive Science* **11**. 23-63.

Grossberg, S (1988), Nonlinear Neural Networks: Principles, Mechanisms, and Architectures, *Neural Networks* **1**, 17-61.

Harvey, R. L. (1994), *Neural Network Principles* Prentice-hall International Inc. London

Hamilton, A. G. (1988), *Logic For Mathematicians* Cambridge University Press

Haykin, S. (1994), *Neural Networks: a Comprehensive Foundation* Macmillan NY.

Hebb, D. O. (1949), *Organization of Behavior* ,Wiley, NY.

Hebb, D. O. (1972), *Textbook of Psychology* Saunders Philadelphia

Hecht-Nielsen, R. (1990), *Neurocomputing* Addison Wesley, Reading, MA.

Hergenhahn, B. R. (1992), *An Introduction to the History of Psychology* Wadsworth Publishing Company Belmont CA.

Hertz, J., Krogh, A. and Palmer, R. G. (1991), *Introduction to the Theory of Neural Computation* Addison Wesley Reading MA

Hocking, L. M., (1991), Optimal Control: *An Introduction to the Theory with Applications*, Oxford Applied Mathematics and Computing Science Series, Clarendon Press, Oxford.

Hornik, K. (1993), Some New Results on Neural Network Approximation. N*eural Networks*, **6**, 1069-1072.

Hornik, K., Stinchcombe, M. & White, H. (1989), Multilayer Feedforward Networks are Universal Approximators., *Neural Networks*, **2**, 359-366.

Howard, R.W (1987), *Concepts and Schemata: An Introduction* Cassell Education

Howell, M. N. W. (1994), *Applications of Complex Systems to Pattern Recognition, Optimisation and Control* Ph. D. Thesis University of Sheffield

Hu, Y. and Fellman, R. D. (1995), An Implementation Efficient Learning Algorithm for Adaptive Control Using Associative Content Addressable Memory *IEEE Trans. on Syst. Man and Cybern.*, **25**, (4), 704-709

Jacobs, R. A. and Jordan, M. I. (1993), Learning Piecewise Control Strategies in a Modular Neural Network Architecture *IEEE Trans. on Syst. Man and Cybern.*, **23**, (2), 337-345

James, W (1892), *Textbook of Psychology: Briefer Course*, Macmillan & Co. Ltd, London.

Jang, R. J-S. (1992), Self-Learning Fuzzy Controllers Based on Temporal Back Propagation, *IEEE Trans. on Neural Networks*, **3** (5).

Jang, R. J-S. (1993), ANFIS: Adaptive-Network-Based Fuzzy Inference System, *IEEE Trans. on Syst. Man and Cybern.*, **23**, (3), 665-685

Jang, R. J-S. and Sun, C-T. (1995), Neuro-fuzzy Modelling and Control. *Proc. IEEE*, **83**, (3)

Jianan, F. Hussein, E. M. and Shihuang, S. (1995), Self_Learning Fuzzy Control Based on Adaptive Critic with CMAC Technique *Proceedings of the IEEE International Conference on Neural Networks and Signal Processing*, Nanjing China Vol. 1, 564-567

Johnson, J. A., and Smartt, H. B. (1993), Fuzzy Logic and the Associative Search Element *Proc. World congress on Neural Networks INNS* Portland Oregon Vol. II 52-55

Jordan M. I. and Jacobs, R. A. (1990), Learning to Control an Unstable System with Forward Modelling in *Advances in Neural Information Processing Systems 2* Touretsky, D. (Ed) Morgan Kauffman 324-331

Kadirkamanathan, V. and Niranjan, M (1992), Technical Report CUED/F-INFENG/TR.111, Cambridge University, Cambridge, UK.

Khedar, P. S. and Berenji, H. R. (1993), Generating Fuzzy Rules with Linear Consequents from Data *Proceedings of the World congress on Neural Networks INNS* Portland Oregon Vol. II 18-21

Kim, Y. S. and Lee, J. G. (1990), Design of a Pole-Balancing Controller Using Neural Networks *Proceedings of the International Conference on Neural Networks* Vol 2. 619-612

Klopf, A. H. (1972), Brain Function and Adaptive Systems: a Heterostatic Theory. Air Force Cambridge Research Laboratories Research Report AFCRL-72-0164, Bedford MA

Klopf, A. H., (1986), A Drive-Reinforcement Model of Single Neuron function: An Alternative to the Hebbian Neuronal Model in J. Denker (Ed.) *AIP Conference Proceedings 151, Neural Networks for Computing*, 77-85 New York, AIP

Klopf, A. H., (1988), A Neuronal Model of Classical Conditioning, Psychobiology, **16**, (2), 85-125.

Kohonen, T. (1989), *Self-Organisation and Associative Memory*, (3rd Edn.) Springer-Verlag, Berlin.

Kohonen, T. (1995), *Self-Organizing Maps* Springer series in information Sciences Springer-Verlag Berlin

Kosko, B., (1992), Neural Networks and Fuzzy Systems: A dynamical Systems Approach to Machine Intelligence (263 - 298) Prentice-Hall International, Inc.

Kruse, R, Gebhardt, J. and Klawonn, F. (1994), *Foundations of Fuzzy Systems* John Wiley and Sons Chichester

Kruse, R. Gebhardt, J. and Pair, R. (Eds) (1994), *Fuzzy Systems in Computer Science* Vieweg

Lahdhiri, T., Carnal, C. L. and Alouani, A. T. (1994), Cart-Pendulum Balancing Problem using Fuzzy Logic Control *Proceedings of the IEEE Southeastcon 1994* 393-367

Langton, C.G. (1989) *Artificial Life* Addison Wesley, Redwood City CA

Le Cun, J. Denker, J. S. and Solla, S. A. (1990), Optimal Brain Damage in *Advances in Neural Information Processing Systems 2* Touretsky, D. (Ed) 598-605

Levin, A. U. and Narendra, K. S. (1993), Control of Nonlinear Dynamical Systems Using Neural Networks: Controllability and Stabilization *IEEE Transactions on Neural Networks*. **4**, 2, 192-207.

Levine, D. S. (1991), *Introduction to Neural and Cognitive Modelling* Lawrence Erlbaum Associates Hillsdale NJ

Lim, C. P. and Harrison, R. F. (1995) Probabilistic Fuzzy Artmap: an Autonomous Neural Network Architecture for Bayesian Probability Estimation. *Proc. IEE, 4th Int Conf. On ANNs, Cambridge* 148-153.

Lin, C-J and Lin, C-T (1996), Reinforcement Learning for an ART-Based Fuzzy Adaptive Learning Control Network *IEEE Transactions on Neural Networks* **7** (3) 709-731

Lin, C-S. and Kim, H.(1991), CMAC-based Adaptive Critic Self-Learning Control. *IEEE Transactions on Neural Networks*. **2**, 5, 530-533.

Linkens, D. and Abbod, M. F. (1993), Supervisory Intelligent Control Using A Fuzzy logic Hierarchy *Trans. Inst. MC* **15** (3) 112-132

Liu, G. P., Kadirkamanathan, V. and Billings, S. A. (1994), Stable Sequential Identification of Continuous Nonlinear Dynamical Systems by Growing RBF Neural Networks, *Research Report No 547,* The University of Sheffield, U.K.

Ma, Z., Harrison, R. F. and Kennedy, R. L. (1995) GR2 A Hybrid Knowledge-Based System using General Rules. *International Joint Conference on Artificial Intelligence* Montreal.

McClelland, D.E. and Rumelhart, J.L (1986), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol 1: Foundations* MIT Press Cambridge MA.

Marriott, S and Harrison, R. F. (1994), A modified Fuzzy ARTMAP Architecture for the Approximation of Noisy Mappings, *Research Report No 522,* The University of Sheffield, U.K.

Marriott, S and Harrison, R. F. (1995a), A modified Fuzzy ARTMAP Architecture for the Approximation of Noisy Mappings, *Neural Networks* 8 (4) 619-641

Marriott, S. and Harrison, R. F. (1995b), A Self-Organising State Space Decoder for Reinforcement Learning, *Research Report No 582,* The University of Sheffield, U.K.

Marriott, S. and Harrison, R. F. (1996), *Proceedings of Control'96 Exeter U.K* 1113-1117

Maricic (1991)

Mendel, J. M. (1995), Fuzzy Logic Systems for Engineering: A Tutorial, *Proceedings of the IEEE,* 83 (3) 345-377

Mendelson, E (1987). *Introduction to Mathematical Logic* Wadsworth and Brooks/Cole Monterey CA

Michie, D. and Chambers, R. A., (1968a), BOXES: an Experiment in Adaptive Control, in *Machine Intelligence 2*, E. Dale and Michie, D. Eds. Edinburgh: Oliver and Boyd.

Michie, D. and Chambers, R. A., (1968b),'Boxes' as a model of pattern-formation in *Towards a Theoretical Biology* (Ed.) Waddington, C. H. Oliver and Boyd Edinburgh

Minsky, M. L. (1967), *Computation: Finite and Infinite Machines* Prentice-Hall Series in Automatic Computation

Moody, J. and Darken, C. (1988), Learning with Localized Receptive Fields. In Touretzky, D. *et al* (Eds.), *Proceedings of the 1988 Connectionist Models Summer School,* (174-185) San Mateo, CA, Morgan Kaufmann Publishers.

Moody, J. and Darken, C. (1989), Fast Learning in Networks of Locally_tuned Processing Units. *Neural Computation* **1** (2) 281-294

Moore, B., (1989), ART 1 and Pattern Clustering. In Touretzky, D. *et al* (Eds.), *Proceedings of the 1988 Connectionist Models Summer School,* (174-185) San Mateo, CA, Morgan Kaufmann Publishers.

Morgan, J. S. Patterson, E. C. and Klopf, A. H. (1990), Drive-Reinforcement Learning: a Self-Supervised Model for Adaptive Control *Network* **1** 439-448

Myers, C. E. (1992), *Delay Learning in Artificial neural Networks*, Chapman and Hall, London.

Narendra, K. S. and Thathachar, M. A. L. (1974), Learning Automata-A Survey, *IEE Transactions on Systems, Man and Cybernetics,* 4, 323-334

Nie, J. and linkens, D. A. (1994), FCMAC a Fuzzified Cerebelar Model Articulation Controller with self-organizing Capacity *Automatica* 30 (4) 655-664

Nigrin, A. (1993), *Neural Networks for Pattern Recognition* MIT Press Cambridge MA

Norton, J. P. (1986), *An Introduction to Identification* Academic Press, London

Ogata, K. (1990) *Modern Control Engineering* Prentice-Hall International Inc. London

Pao, Y-H (1989) *Adaptive Pattern Recognition and Neural Networks* Addison-Wesley Publishing Company Inc., Reading Mass

Pavlov, I. P. (1928), *Lectures on Conditioned Reflexes, Vol. I,* (Trans.) Gantt, W. H., Lawrence & Wishart Ltd, London.

Pedrycz, W. (1993), *Fuzzy Control and Fuzzy Systems* Second, extended, edition Research Studies Press Taunton

Picton, P. (1994), *Introduction to Neural Networks* Macmillan Hants

Pinel, J. P. J. (1993), *Biopsychology* (2nd Edn.) Allyn and Bacon.

Platt, J. C. (1991), A Resource Allocating Network for Function Interpolation. *Neural Computation* 3 (2), 215-225.

Powell, M. J. D. (1987), Radial Basis Functions for Multivariable Interpolation: a Review, in *Algorithms for Approximation*, Mason, J. C. and Cox, M. G. (eds.) Clarendon Press Oxford 143-167

Prockyk, T. J. and Mamdani, E. H. (1979), A Linguistic Self-Organising Process Controller, *Automatica*, **15** 15-30

Randall, M., Thorne, C., and Wild, C. (1994), A Standard Comparison of Adaptive Controllers to Solve the Cart-Pole Problem *Proceedings of the Second Australian and New Zealand Conference on Intelligent Information Systems* 61-65

Reed, R. (1993), Pruning Algorithms-A Survey, *IEEE Transactions on Neural Networks*, **4**, 5, 740-747

Ritter, H., Martinetz, T., and Schulten, K. (1992), *Neural Computation and Self-Organizing Maps: an Introduction*, Addison-Wesley Publishing Company, Reading MA.

Roberts, J. (1993), *Making Sense of English in Psychology* Chambers Harrap Ltd. Edinburgh.

Rosenblatt, F. (1962), *Principles of Neurodynamics*. New York, Spartan.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986), Learning Internal Representation by Error Propagation. In Rumelhart, D. E. & McLelland (Eds.), *Parallel Distributed Processing*, I, 318-362 Cambridge, MA, MIT Press.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986) Learning Representations by Back-propagating Errors, *Nature* **323** 533-536

Rumelhart, D. E. and Zipser, D (1985) Competitive Learning In Rumelhart, D. E. & McLelland (Eds.), *Parallel Distributed Processing*, I, 152-193 Cambridge, MA, MIT Press.

de Sa V. R. and Ballard, D. H. (1993), a Note on Learning Vector Quantisation in Henson S.J., Cowan, J. D. and Giles, D. L. (Eds.) *Advances in Neural Information Processing 5*, Morgan Kaufmann, San Mateo, CA. 221-227

Sammut, C. and Cribb, J. (1990), Is Learning Rate a Good Performance Criterion for Learning? *Proceedings of the Seventh International Workshop On Machine Learning*. Morgan Kaufmann 170-178

Samuel, A. L. (1959) Some Studies in Machine Learning Using the Game of Checkers, *IBM Journal of Research and Development* 221-229

Santiago, R. A. and Werbos, P. J. (1994), New Progress Towards Truly Brain-Like Intelligent Control *Proceedings of the World congress on Neural Networks 1994* Vol 1. 26-33

Saridis, G. N. (1989), Analytic Formulation of the Principle of Increasing Precision with Decreasing Intelligence for Intelligent Machines. *Automatica* 25, 3, 461-467.

Scarborough, J. B. (1966), *Numerical Mathematical Analysis*. (Sixth Edn.) The Johns Hopkins Press. Baltimore.

Selfridge, O. (1959), Pandemonium: A paradigm for learning, in *Symposium on the mechanisation of though processes*, London, HMSO.

Sharkey, N. E. and Sharkey, A. J. C. (1994), Understanding Catastrophic Interference in Neural Nets. *Research Report CS-94-4*, University of Sheffield, U. K.

Simpson, P. K. (1990), *Artificial Neural Systems: Foundations, Paradigms, Applications and Implementations* Pergamon Press, NY

Specht, D (1990) Probabilistic Neural Networks *Neural Networks* **3** 109-118

Söderstrom, T. and Stoica, P (1989) *System Identification* Prentice Hall NY

Srinivasan, B. Prasad, U. R. and Rao, N. J. (1994), Back Propagation Throug Adjoints for the Identification of Nonlinear Dynamic Systems Using Recurrent Neural Networks *IEEE Trans. On Neural Networks*, **5** (2), 213-227

Sugeno, M. and Nishida, M. (1985), Fuzzy Control of Model Car, *Fuzzy Sets and Systems* **16** 103-113

Sutton, R. S. (1988), Learning to Predict by the Methods of Temporal differences, *Machine Learning*, **3**, 9-44.

Sutton, R. S. (Ed.) (1992), Reinforcement Learning: A Special Issue of *Machine Learning* on Reinforcement Learning Kluwer Academic Publishers.

Sutton, R. S., and Barto, A. G. (1981), Towards a Modern Theory of Adaptive Networks: Expectation and Prediction, *Psychological Review*, **88** (2), 135-170.

Sutton, R. S., and Barto, A. G. (1990), Time-Derivative Models of Pavlovian Reinforcement, in Gabriel, M. and Moore, J. *Learning and Computational Neuroscience: Foundations of Adaptive Networks*. MIT Press, Cambridge MA. 497-537.

Sutton, R. S., Barto, A. G. and Williams, R. J. (1992), Reinforcement Learning is Direct Adaptive Optimal control, *IEEE Control Systems Magazine*, April, 19-22.

Thorndyke, E. L. (1911), *Animal Intelligence*, Macmillan, NY

Timan, A. F. (1994), *Theory of Approximation of Functions of a Real Variable*, Dover Publication Incorporated, New York.

Tolat, V. V.and Widrow, B. (1988), An Adaptive "Broom Balancer" with Visual Inputs *Proceedings of the IEEE International Conference on Neural Networks* Vol. 2 641-647

Tolle, H., Ersü, E. (1992), *Neurocontrol: Learning Control Systems Inspired by Neuronal Architectures and Human Problem Solving Strategies*, Lecture Notes in Control and Information Sciences. Springer Verlag

Vapnic, V. N. (1995) *The Nature of Statistical Learning Theory* Springer N.Y.

Wang, Li-Xin, and Mendel, J. M. (1992), Generating Fuzzy Rules by Learning from Examples, *IEEE Trans. Sys., Man and Cybern.* 22 (6), 1414-1427.

Walker, S. (1975), *Learning and Reinforcement* Volume A3 of Essential Psychology Series Herriot, P. (Ed.) Methuen

Watkins, C.J.C.H (1989) Ph.D. Thesis University of Cambridge.

Watkins, C. J. C. H. and Daynan, P. (1992), Technical Note: Q-Learning *Machine Learning* 8 279-292

Watson, J. B. (1913), Psychology as the Behaviourist Sees it, *Psychological Review* 20, 158-177

Werbos, P. J. (1990) Backpropagation Through Time: What it is and How to Do it, *Proc. IEEE* **78** 1550-1560.

Wiberg, D. M. (1971) *State-Space and Linear Systems* McGraw-Hill

Widrow, B. (1987) The Original Adaptive Neural Net Broom-Balancer. *Proceedings of the International Symposium on Circuits and Systems* 351-357

Widrow, B. and Hoff, M. E. (1960). Adaptive Switching Circuits. In 1960 IRE WESCON Record, **4**, 96-104. NY IRE. Reprinted in Anderson and Rosenfield (1988)

Widrow, B, and Smith, F. W. (1963). Pattern recognizing Control Systems. *Computer Information Sciences* (COINS) *Symposium*

Widrow, B. and Lehr, M. A. (1990) 30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation *Proceedings of the IEEE* **78** (9) 1415-1442.

Wieland, A. P. (1991) Evolving Neural Network controllers for Unstable Systems *Proc. IEEE* Vol 2 667-673

Zadeh, L. A., (1965). Fuzzy Sets. *Information and Control,* **8**, 338-353.

Zhang, B. and Grant. E. (1992), Using Competitive Learning For State-Space Partitioning, *Proceedings if the IEEE international Symposium on Intelligent Control,* 391-395

Zeidenberg, M. (1990), *Neural Networks in Artificial Intelligence*, Ellis Horwood Series in Artificial Intelligence, Ellis Horwood, New York.

Appendix A The Monotonic Increasing Property of the fuzzy ART Choice Function (Marriott and Harrison, 1994)

This proof is included to illustrate an important property of ART module choice functions.

When $\mathbf{w}_j$ is a fuzzy subset of $\mathbf{I}$, the Fuzzy ART choice function is of the form

$$f(x)=\frac{ax}{b+x}.$$

Theorem: For a function $f:[0,1]\rightarrow[0,1]$, $f(x)=\frac{ax}{b+x}$, where a and b are positive constants, given some $x_1,x_2\in[0,1]$ $f(x_2)\geq f(x_1)\Rightarrow x_2\geq x_1$.

In the thesis, the above property is referred to as the "monotonic increasing property" or M.I.P.

Proof: For some $x_1,x_2\in[0,1]$ assume that $f(x_2)\geq f(x_1)$, i.e.

$$\frac{ax_2}{b+x_2}\geq\frac{ax_1}{b+x_1}.$$

Using the rules of inequalities

$$\frac{ax_2(b+x_1)}{(b+x_1)(b+x_2)}\geq\frac{ax_1(b+x_2)}{(b+x_1)(b+x_2)}$$

giving,

$$abx_2\geq abx_1$$

and

$$x_2\geq x_1.\qquad\blacksquare$$

Similarly, it can be proved that

$$x_2\geq x_1\Rightarrow f(x_2)\geq f(x_1).$$

Appendix B: ARTMAP: a Numerical Example.

The patterns to be associated in this simple example are:

$$\mathbf{I}_1^a, \quad 111110 \rightarrow 1010, \quad \mathbf{I}_1^b$$

$$\mathbf{I}_2^a, \quad 111100 \rightarrow 0101, \quad \mathbf{I}_2^b$$

$$\mathbf{I}_3^a, \quad 111000 \rightarrow 1010, \quad \mathbf{I}_1^b$$

The parameters used in this example are: $\alpha = 2.0$, $\delta = 0.01$, $M_a = N_a = 6$, $M_b = N_b = 4$. The baseline vigilance for ARTa, specified by $\overline{\rho}_a = 0.4$, illustrates match-tracking in ARTMAP. ART b vigilance, $\rho_b = 0.9$

Initialise weights:

$$w_{ij}^{(TDa)}(0) = 1.0,\ w_{ij}^{(TDb)}(0) = 1.0,\ w_{ji}^{(BUa)}(0) = \frac{1}{\alpha + M_a} - \delta = \frac{1}{2+6} - 0.01 = 0.115$$

and

$$w_{ji}^{(BUb)}(0) = \frac{1}{\alpha + M_b} - \delta = \frac{1}{2+4} - 0.01 = 0.157$$

Present inputs:, $\mathbf{I}^a(1) = \mathbf{I}_1^a = [111110]^t$, $\mathbf{I}^b(1) = \mathbf{I}_1^b = [1010]^t$

Consider the ARTb module. Propagate the input to F1b giving $\mathbf{x}^b(t) = \mathbf{I}^b(1) = \mathbf{I}_1^b$. Propagate to F2b via bottom-up connections:

$$net_j(1) = \sum_{i=1}^{M_b} w_{ji}^{(BUb)}(0) x_i^b(1)$$
$$= 0.157 \times 1 + 0.157 \times 0 + 0.157 \times 1 + 0.157 \times 0$$
$$= 0.314$$

It is the same for all F2a nodes as there are no committed nodes yet so the first node is chosen as the winner i.e. J=1.

Propagate back to F1b via top-down connections giving $\mathbf{x}^b(1) = \mathbf{I}^b(1) + \mathbf{w}_J^{(TDb)}(0)$ or componentwise, $x_i = I_i + w_{ij}^{(TD)}$. Apply the condition *If* $x_i \geq 1 + \bar{z}$ *then* $x_i = 1$ *else* $x_i^b = 0$ There is no gain as F1b and F2b are active and so,

$x_1^b(1) = 1+1 = 2$, $x_2^b(1) = 0+0.157 = 0.157$, $x_3^b(1) = 1+0.157 = 1.157$ and $x_4^b(1) = 0+0.157 = 0.157$, which implies that, $x_1^b(1) = 1, x_2^b(1) = 0, x_3^b(1) = 1$ and $x_4^b(1) = 0$. after thresholding.

Matching at ARTb F1 gives, for $\mathbf{x}^b(1) = [1010]^t$, $\frac{|\mathbf{x}^b(1)|}{|\mathbf{I}^b(1)|} = 1.0 \geq \rho_b = 0.9$. A similar sequence of events follows for the ARTa module where F2a node 1 is activated.

For the map field, $x_k^{ab}(1) = y_k^b(1) + y_J^a(1) w_{kJ}^{(ab)}(0)$, and thus, $x_1^{ab} = 1+1\times1 = 2$, $x_2^{ab} = 0+1\times1 = 1$, $x_3^{ab} = 0+1\times1 = 1$ and $x_4^{ab} = 0+1\times1 = 1$ giving $\mathbf{x}^{ab} = [1000]^t$ after thresholding.

Now, $\mathbf{x}^{ab}(1) = \mathbf{y}^b(1) \cap \mathbf{w}_J^{ab}(0) = \mathbf{y}^b(1)$ because, $\mathbf{w}_J^{ab}(0) = [1111]^t$ signifying that no association has been learned yet as the J th ARTa node is uncommitted.

At this stage the ARTa and ARTb modules and the map field have to be updated.

For ARTa: $w_{iJ}^{(TDa)}(1) = x_i^a(1)$ giving $\mathbf{w}_J^{(TDa)}(1) = \begin{bmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 0.0 \end{bmatrix}^t$ and

$$W^{(TDa)} = \begin{bmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{bmatrix}$$

Also, $w_{Ji}^{(BUa)}(1) = \dfrac{I_i(1)}{\alpha + |\mathbf{x}^a(1)|}$ giving $w_{11}^{(BUa)}(1) = \dfrac{1}{2+5} = \dfrac{1}{7} = 0.143$. similarly,

$w_{12}^{(BUa)}(1) = w_{13}^{(BUa)} = w_{14}^{(BUa)}(1) = (1)w_{15}^{(BUa)}(1) = 0.143,$

but $w_{16}^{(BUa)}(1) = \dfrac{0}{2+5} = \dfrac{0}{7} = 0.0$ giving

$\mathbf{w}_1^{(TDa)}(1) = [0.143 \quad 0.143 \quad 0.143 \quad 0.143 \quad 0.143 \quad 0.0]$ and

$$W^{(BUa)}(1) = \begin{bmatrix} 0.143 & 0.143 & 0.143 & 0.143 & 0.143 & 0.0 \\ 0.115 & 0.115 & 0.115 & 0.115 & 0.115 & 0.115 \\ 0.115 & 0.115 & 0.115 & 0.115 & 0.115 & 0.115 \\ 0.115 & 0.115 & 0.115 & 0.115 & 0.115 & 0.115 \\ 0.115 & 0.115 & 0.115 & 0.115 & 0.115 & 0.115 \\ 0.115 & 0.115 & 0.115 & 0.115 & 0.115 & 0.115 \end{bmatrix}$$

Similarly for ART b: $\mathbf{w}_1^{(TDb)}(1) = [1.0 \quad 0.0 \quad 1.0 \quad 0.0]^t$ and

$\mathbf{w}_1^{(BUb)}(1) = [0.250 \quad 0.0 \quad 0.250 \quad 0.0]^t$ giving the ARTb weight matrices

$$W^{(TDb)}(1) = \begin{bmatrix} 1.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & 1.0 \end{bmatrix} \text{ and } W^{(BUb)}(1) = \begin{bmatrix} 0.250 & 0.0 & 0.250 & 0.0 \\ 0.157 & 0.157 & 0.157 & 0.157 \\ 0.157 & 0.157 & 0.157 & 0.157 \\ 0.157 & 0.157 & 0.157 & 0.157 \end{bmatrix}$$

respectively.

The map field weights are given by $w_{kJ}^{(ab)}(t+1) = x_k^{ab}$ and, thus,

$$W^{ab}(1) = \begin{bmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{bmatrix}$$

For the next cycle, $\mathbf{I}^a(2) = \mathbf{I}_2^a = [111100]^t$ and $\mathbf{I}^b(2) = \mathbf{I}_2^b = [0101]^t$. Note that

$\mathbf{I}_2^a \subset \mathbf{I}_1^a$ and $\mathbf{I}_2^b = \left(\mathbf{I}_1^b\right)^c$

Consider the ARTb module. Propagate input to F1b giving $\mathbf{x}^b(2) = \mathbf{I}^b(2) = \mathbf{I}_2^b$.

Propagate to F2b (BU) giving

$$net_1(2) = \sum_{i=1}^{M_b} w_{1i}^{(BUb)}(1) x_i^b(2)$$
$$= 0.250 \times 0 + 0.0 \times 1 + 0.250 \times 0 + 0.0 \times 1$$
$$= 0.0$$

and

$net_2(2) = net_3(2) = net_4(2) = 0.314$. Choose J=2 as the winning node and propagate back to F1b (TD) giving $\mathbf{x}^b(2) = \mathbf{I}^b(2) + \mathbf{w}_J^{(TDb)}(1)$. Componentwise, $x_i = I_i + w_{ij}^{(TD)}$. Apply the condition *If* $x_i \geq 1 + \bar{z}$ *then* $x_i = 1$ *else* $x_i^b = 0$ again giving $x_1^b(2) = 0 + 1 = 1$, $x_2^b(2) = 1 + 1 = 2$ $x_3^b(2) = 0 + 1 = 1$ and $x_4^b(2) = 1 + 1 = 2$,which implies that $\mathbf{x}^b(2) = [0101]^t$ after thresholding.

Matching at F1b gives, f or $\mathbf{x}^b(2) = [0101]^t$, $\frac{|\mathbf{x}^b(2)|}{|\mathbf{I}^b(2)|} = 1.0 \geq \rho_b = 0.9$.

For ARTa: Propagate input to F1a giving $\mathbf{x}^a(2) = \mathbf{I}^a(2) = \mathbf{I}_2^a$. Propagate to F2a (BU) giving

$$net_1^a(2) = \sum_{i=1}^{M_a} w_{1i}^{(BUa)}(1) x_i^a(2)$$
$$= 0.143 \times 1 + 0.143 \times 1 + 0.143 \times 1 + 0.143 \times 1 + 0.143 \times 0 + 0.0 \times 1$$
$$= 0.572$$

and

$$net_2^a(2) = \sum_{i=1}^{M_a} w_{2i}^{(BUa)}(1) x_i^a(2)$$
$$= 0.115 \times 1 + 0.115 \times 1 + 0.115 \times 1 + 0.115 \times 1 + 0.115 \times 0 + 0.0 \times 1$$
$$= 0.46$$

similarly $net_3^a(2) = net_4^a(2) = net_5^a(2) = net_6^a(2) = 0.46$ andF2a node 1 wins the competition. This makes sense because $\mathbf{I}_2^a \subset \mathbf{I}_1^a$ and none of the other F2a nodes are committed at this stage.

Propagate back to F1a (TD) giving $\mathbf{x}^a(2) = \mathbf{I}^a(2) + \mathbf{w}_{J=1}^{(TDa)}(1)$ or Componentwise,

$x_i = I_i + w_{ij}^{(TD)}$. Applying the condition, *If* $x_i \geq 1 + \bar{z}$ *then* $x_i = 1$ *else* $x_i^b = 0$

gives

$x_1^a(2) = 1+1 = 2$, $x_2^a(2) = x_3^a(2) = x_4^a(2) = 2$, $x_6^a(2) = 0+1 = 1$, and

$x_6^a(2) = 0+0 = 0$ which implies that $\mathbf{x}^a(2) = [111100]^t$ after thresholding and

$$\mathbf{x}^a(2) = \mathbf{I}_2^a \cap \mathbf{w}_1^{(TDa)}(1)$$
$$= \mathbf{I}_2^a \cap \mathbf{I}_1^a$$
$$= \mathbf{I}_2^a$$

Matching at ARTa gives, for $\mathbf{x}^b(2) = \mathbf{I}_2^a$, $\frac{|\mathbf{x}^a(2)|}{|\mathbf{I}^a(2)|} = \frac{|\mathbf{I}_2^a|}{|\mathbf{I}_2^a|} 1.0 \geq \rho_b = 0.9$. The F2 activity vectors for ARTa and ARTb are now given by $\mathbf{y}^a(2) = [100000]^t$ and $\mathbf{y}^b(2) = [0100]^t$ respectively.

For the map field:

$\mathbf{w}_{J=1}^{ab}(1) = [1000^t]$, and $x_k^{ab}(2) = y_k^b(2) + y_{J=1}^a(1) w_{kJ=1}^{(ab)}(1)$,

thus, $x_1^{ab}(2) = 0+1\times1 = 1$ $x_2^{ab}(2) = 1+1\times0 = 1$, $x_3^{ab}(2) = 0+1\times0 = 0$ and

$x_4^{ab}(2) = 0+1\times0 = 0$ giving $\mathbf{x}^{ab}(2) = [0000]^t$ after thresholding,

i.e $\mathbf{x}^{ab}(2) = \mathbf{y}^b(2) \cap \mathbf{w}_J^{ab}(1) = \phi$. This is because the active ARTb category (category 2) is not predicted by ARTa. ARTb category 1 is predicted because the ART a input is a subset of the ARTa category 1 exemplar and ARTa category 1 is linked via the map field to ARTb category 1.

The map field match criterion $\frac{|\mathbf{x}^{ab}(2)|}{|\mathbf{y}^b(2)|} \geq \rho_{ab}$ fails because $|\mathbf{x}^{ab}(2)| = 0$ and the current ARTb category is not what is predicted. Match tracking attempts to rectify this incorrect prediction by raising the ARTa vigilance so that the currently active ARTa node is no longer chosen and a new ARTa category found with the

correct prediction or a new map field linkage is created using a newly committed ARTa node.

However, the problem lies within ARTa and cannot be solved with match tracking using the system described so far. Normally, a map field mismatch would trigger match tracking which would increase the ARTa vigilance above the ratio $\frac{|\mathbf{x}^a(t)|}{|\mathbf{I}^a(t)|} = \frac{|\mathbf{I}^a(t) \cap \mathbf{w}_j^{(TDa)}(t)|}{|\mathbf{I}^a(t)|}$ which, in this particular case is $\frac{|\mathbf{x}^a(2)|}{|\mathbf{I}^a(2)|} = \frac{|\mathbf{I}_2^a|}{|\mathbf{I}_2^a|} = 1$ and no future ARTa match will be greater than unity. This means that no other node may be recruited or created. ARTb has learned the new input but it cannot be associated with the current ARTa input.

For the third input $\mathbf{I}_3^a \subset \mathbf{I}_1^a$ and $\mathbf{I}_1^a$ belongs to ARTa category 1 linked which is linked to ARTb category 1 as required.

Appendix C: The ARTMAP Match-Tracking Theorem

Theorem: The *ARTMAP Match-Tracking Theorem* Any ARTa input which is not equal to any previously stored ARTa input will always trigger match tracking activity in ARTMAP if complement coding is used.

Proof:

Let $\mathbf{I}_j$ denote some input $\mathbf{I}^a(t)$ at time, t and $\mathbf{I}_i$ denote some previous input $\mathbf{I}^a(t-\tau)$ at time t-τ such that $\mathbf{I}_j \subset \mathbf{I}_i$. Using complement coding, the following inputs can be defined: $\mathbf{I}'_j = \left(\mathbf{I}_j, \mathbf{I}^c_j\right)$, and $\mathbf{I}'_i = \left(\mathbf{I}_i, \mathbf{I}^c_i\right)$.

Without loss of generality, assume $\mathbf{I}'_i$ is stored by some top-down ARTa weight, i.e. $\mathbf{w}_J^{(TDa)}(t) = \mathbf{I}'_i$.

The ARTa matching condition can be stated as $\dfrac{\left|\mathbf{I}^a(t) \cap \mathbf{w}_J^{(TDa)}\right|}{\left|\mathbf{I}^a(t)\right|} \geq \rho_a$.

Assume the input $\mathbf{I}'_j$ triggers ARTa node J such that $\mathbf{w}_J^{(TDa)}(t) = \mathbf{I}'_i$. The match condition is then $\dfrac{\left|\mathbf{I}^a(t) \cap \mathbf{w}_J^{(TDa)}(t)\right|}{\left|\mathbf{I}^a(t)\right|} = \dfrac{\left|\mathbf{I}'_j \cap \mathbf{w}_J^{(TDa)}(t)\right|}{\left|\mathbf{I}'_j\right|} = \dfrac{\left|\mathbf{I}'_j \cap \mathbf{I}'_i\right|}{\left|\mathbf{I}'_j\right|} \geq \rho_a$

It is required to prove that $\left|\mathbf{I}'_j \cap \mathbf{I}'_i\right| < \left|\mathbf{I}'_j\right|$ so that the matching ratio does not equal unity to allow the ARTa vigilance to be increased through match tracking.

By hypothesis, $\mathbf{I}_j \subset \mathbf{I}_i$, which implies that $\mathbf{I}^c_i \subset \mathbf{I}^c_j$. Now,

$$\mathbf{I}'_j \cap \mathbf{I}'_i = \left(\mathbf{I}_j \cup \mathbf{I}^c_j\right) \cap \left(\mathbf{I}_i \cup \mathbf{I}^c_i\right)$$
$$= \left(\mathbf{I}_j \cap \mathbf{I}_i\right) \cup \left(\mathbf{I}_j \cap \mathbf{I}^c_i\right) \cup \left(\mathbf{I}^c_j \cap \mathbf{I}_i\right) \cup \left(\mathbf{I}^c_j \cap \mathbf{I}^c_i\right)$$
$$= \left(\mathbf{I}_j \cap \mathbf{I}_i\right) \cup \phi \cup \phi \cup \left(\mathbf{I}^c_j \cap \mathbf{I}^c_i\right)$$
$$= \left(\mathbf{I}_j \cap \mathbf{I}_i\right) \cup \left(\mathbf{I}^c_j \cap \mathbf{I}^c_i\right)$$

By hypothesis, $\mathbf{I}_j \subset \mathbf{I}_i$ giving $\mathbf{I}_j \cap \mathbf{I}_i = \mathbf{I}_j$ and, by deduction, $\mathbf{I}^c_i \subset \mathbf{I}^c_j$, giving,

$$\mathbf{I}^c_j \cap \mathbf{I}^c_i = \mathbf{I}^c_i$$

Substituting these terms into $\mathbf{I}'_j \cap \mathbf{I}'_i = \left(\mathbf{I}_j \cap \mathbf{I}_i\right) \cup \left(\mathbf{I}^c_j \cap \mathbf{I}^c_i\right)$ gives

$$\mathbf{I}'_j \cap \mathbf{I}'_i = \mathbf{I}_j \cup \mathbf{I}^c_i$$

The condition $\left|\mathbf{I}'_j \cap \mathbf{I}'_i\right| < \left|\mathbf{I}'_j\right|$ can now be replaced by the equivalent condition $\left|\mathbf{I}_j \cup \mathbf{I}^c_i\right| < \left|\mathbf{I}'_j\right|$. Now, $\left|\mathbf{I}_j \cup \mathbf{I}^c_i\right| = \left|\mathbf{I}_j\right| + \left|\mathbf{I}^c_i\right|$, and $\left|\mathbf{I}'_j\right| = \left|\mathbf{I}_j\right| + \left|\mathbf{I}^c_j\right|$.

By hypothesis $\mathbf{I}_j \subset \mathbf{I}_i$ which implies that $\left|\mathbf{I}_j\right| < \left|\mathbf{I}_i\right|$. Similarly $\mathbf{I}^c_i \subset \mathbf{I}^c_j$ implies that $\left|\mathbf{I}^c_i\right| < \left|\mathbf{I}^c_j\right|$. The latter can also be proved

$$\left|\mathbf{I}_j\right| < \left|\mathbf{I}_i\right| \Rightarrow -\left|\mathbf{I}_j\right| > -\left|\mathbf{I}_i\right| \Rightarrow m - \left|\mathbf{I}_j\right| > m - \left|\mathbf{I}_i\right| \Rightarrow \left|\mathbf{I}^c_j\right| > \left|\mathbf{I}^c_i\right|$$

Now, starting from the fact that $\left|\mathbf{I}_j \cup \mathbf{I}^c_i\right| = \left|\mathbf{I}_j\right| + \left|\mathbf{I}^c_i\right|$ the equivalent match condition may be proved, thus, $\left|\mathbf{I}_j \cup \mathbf{I}^c_i\right| = \left|\mathbf{I}_j\right| + \left|\mathbf{I}^c_i\right| < \left|\mathbf{I}_j\right| + \left|\mathbf{I}^c_j\right| = \left|\mathbf{I}'_j\right|$ and the equivalent condition $\left|\mathbf{I}_j \cup \mathbf{I}^c_i\right| < \left|\mathbf{I}'_j\right|$ is proved as required. ■

Thus it is shown that $\left|\mathbf{I}'_j \cap \mathbf{I}'_i\right| < \left|\mathbf{I}'_j\right|$ and the match condition gives

$$\frac{\left|\mathbf{I}'_j \cap \mathbf{w}_j\right|}{\left|\mathbf{I}'_j\right|} = \frac{\left|\mathbf{I}'_j \cap \mathbf{I}'_i\right|}{\left|\mathbf{I}'_j\right|} < \frac{\left|\mathbf{I}'_j\right|}{\left|\mathbf{I}'_j\right|} = 1.0$$ which allows match tracking to increment the ARTa vigilance parameter.

Appendix D. Fuzzy ARTMAP Category Dynamics: A Single dimensional Example

D1 Introduction

This appendix illustrates the proliferation of categories by fuzzy ARTMAP on the real line when complement coding is not applied (Marriott and Harrison, 1994). This derivation differs from that given in Carpenter, Grossberg and Rosen (1991) by applying real analysis to adjacent categories to establish choice regions and category movement rather than the geometric interpretation. Carpenter, Grossberg and Rosen (1991) gives a geometric interpretation of the effect of complement coding in reducing the proliferation of categories.

Let w_{s-1} and w_s denote the exemplars for nodes s-1 and s respectively where $w_{s-1}, w_s \in [0,1] \subset \Re$. Without loss of generality, assume

$$0 \le w_{s-1} < w_s \le 1 \tag{D1}$$

and that for all inputs, I considered here

$$w_{s-1} \le I \le w_s \tag{D2}$$

for some $s-1, s \in N$. See Figure D1.

Figure D1 Two adjacent categories on the real line

Any input, I, can be parameterised in the range

$$I(\lambda) = w_{s-1} + \lambda(w_s - w_{s-1}) \tag{D3}$$

where $0 \le \lambda \le 1$. Henceforth, $I(\lambda)$ will be denoted by I.

In this case, the choice function of equation (2.13) gives

$$T_{s-1}(I) = \frac{w_{s-1}}{\alpha + w_{s-1}} \tag{D4}$$

and,

$$T_s(I) = \frac{w_{s-1} + \lambda(w_s - w_{s-1})}{\alpha + w_s} \tag{D5}.$$

Consider the effect of the parameter λ. Three cases naturally arise:-

i) $\lambda = 0$,

ii) $\lambda = 1$,

iii) $0 < \lambda < 1$.

For $\lambda = 0$, from equation (D3), $I = w_{s-1}$, and from equation (D5)

$$T_s(I) = \frac{w_{s-1}}{\alpha + w_s}.$$

Also, $T_{s-1}(I) = \frac{w_{s-1}}{\alpha + w_{s-1}}$ by equation (D4).

Now, from equation (D1), $w_s > w_{s-1}$ which implies that $T_{s-1}(I) > T_s(I)$, and node s-1 wins as expected.

For $\lambda = 1$, $I = w_s$, $T_{s-1}(I) = \frac{w_{s-1}}{\alpha + w_{s-1}}$, and $T_s(I) = \frac{w_s}{\alpha + w_s}$.

So, by the monotonic property of $T(I)$, $w_s(I) > w_{s-1}(I)$ gives $T_s(I) > T_{s-1}(I)$ and node s wins as expected.

For $0 < \lambda < 1$ a question naturally arises as to where the decision boundary for adjacent exemplars lies.

Equating $T_{s-1}(I)$ and $T_s(I)$ gives $\frac{w_{s-1}}{\alpha + w_{s-1}} = \frac{w_{s-1} + \lambda(w_s - w_{s-1})}{\alpha + w_s}$ and solving for λ gives

$$\lambda_b = \frac{w_{s-1}}{\alpha + w_{s-1}} \tag{D6}$$

where λ_b is the boundary value of λ.

Thus λ_b is slightly less than one and depends upon α. This means that all inputs in the range given by equation (D2) map to node s-1 unless they are within a small distance of node s. This is proved in the following theorem:

Theorem:

$\forall I$ such that $w_{s-1} \le I < w_{s-1} + \lambda_b(w_s - w_{s-1}), w_{s-1} > 0,$

where λ_b is given by equation (D6), I

maps to the s-1 th category.

Proof:

Let $\lambda = \gamma\lambda_b, \quad 0 < \gamma < 1,$

i.e. $0 < \lambda < \lambda_b$, as required, so that,

$$T_{s-1}(I) = \frac{w_{s-1}}{\alpha + w_{s-1}},$$

and,

$$T_s(I) = \frac{w_{s-1} + \gamma\lambda_b(w_s - w_{s-1})}{\alpha + w_s}.$$

Now,

$w_s > w_{s-1}$.

Multiplication of both sides by $(1-\gamma)$ and further application of the algebra of inequalities leads to,

$$w_{s-1}(\alpha + w_s) > w_{s-1}(\alpha + \gamma w_s) + (1-\gamma)w_{s-1}^2$$

and,

$$\frac{w_{s-1}}{\alpha + w_{s-1}} > \frac{w_{s-1}(\alpha + w_{s-1}) + \gamma w_{s-1}(w_s - w_{s-1})}{(\alpha + w_{s-1})(\alpha + w_s)}$$

$$= \frac{w_{s-1} + \gamma\lambda_b(w_s - w_{s-1})}{(\alpha + \omega_s)}$$

giving, $T_{s-1}(I) > T_s(I)$

for $0 < \gamma < 1$.

The condition $T_{s-1}(I) > T_s(I)$

requires

$$\frac{w_{s-1}}{\alpha + w_{s-1}} > \frac{w_{s-1} + \lambda(w_s - w_{s-1})}{\alpha + w_s}$$

giving

$$w_{s-1} > \lambda(\alpha + w_{s-1})$$

which leads to

$$\lambda < \frac{w_{s-1}}{\alpha + w_{s-1}}.$$

Also, $\lambda > 0$ and $\alpha > 0$ finally giving

$$0 < \lambda < \frac{w_{s-1}}{\alpha + w_{s-1}} < 1.$$

Therefore,

$T_{s-1}(I(\lambda)) > T_s(I(\lambda))$, for λ in the above range. ■

Thus, all inputs between exemplars w_{s-1} and w_s map to category s-1 except for those in a small exclusion zone $(w_{s-1} + \lambda_b(w_s - w_{s-1}), w_s)$ determined by α.

Figure D2. Two adjacent categories in the real line illustrating the exclusion zone near to category s.

Note that the above only determines the winning node through $T(I)$ and not category membership which depends upon the match criterion .

Match Criterion

Equation (2) states the match criterion

$$\frac{|\mathbf{I} \wedge \mathbf{w}|}{|\mathbf{I}|} \geq \rho,$$

which gives $w_{s-1} \geq \rho I$ for node s-1.

Thus, $I \leq \frac{w_{s-1}}{\rho}$ is required for a match to occur.

D2. Category Proliferation

Consider what happens when

$\ldots < w_{s-2} < w_{s-1} < I < w_s < w_{s+1} < \ldots$

By previous results, $T_{s-1}(I) > T_s(I)$, but, $I > \frac{w_{s-1}}{\rho}$ ensures that node s-1 is inhibited.

Again, $T_{s-2}(I) > T_s(I)$, by previous results, but $I > \frac{w_{s-2}}{\rho}$, causes inhibition of node s-2.

Thus, all nodes, $l \leq s-1$ are inhibited.

Now,

$$T_k(I) = \frac{I}{\alpha + w_k}, \quad \forall k \geq s$$

giving

$T_s(I) > T_{s+1}(I) > T_{s+2}(I) > \ldots$ as $w_s < w_{s+1} < w_{s+2} < \ldots$

So, by the above, all nodes, l with exemplars $w_l < w_s$, $l < s$, are inhibited so node s is selected giving $T_s(I) = \frac{I}{\alpha + w_s} > \frac{I}{\alpha + 1}$ for an uncommitted node.

This means that the next available node is selected which has its exemplar w_s replaced by I as the match criterion gives $\frac{|\mathbf{I} \wedge \mathbf{w}_s|}{|\mathbf{I}|} = \frac{I}{I} = 1 > \rho$ for $\rho < 1$, regardless of the distance between I and w_s.

Thus, as I<w_s., exemplars drift towards the origin as their magnitudes are reduced. This causes the creation of more categories in areas of input space made devoid of exemplars by this drifting effect.

Although stable by the monotonic decreasing of weights, the network suffers from proliferation of category nodes unless complement coding is used.

Appendix E: Further Fuzzy ARTMAP and PROBART Results

Mean results are based upon a sample size of 5 RMSE or MAXAE values from separate runs which are averaged to give an indication of performance. Maximum and minimum values are included to indicate the range of variation between runs.

Simulation 1

No. of categories		Error measures	
ARTa	ARTb	RMSE	MAXAE
298	52	0.0074	0.01

Table E1.1. Mean results of fuzzy ARTMAP performance with noise-free training and test data derived from the test function of Figure 3. Training and testing is off-line using 1,000 pattern pairs. See main text for parameter values.

Error measures			
RMSE (TE)[1]	Error range[2].	MAXAE (TE)	Error range.
0.0076	3.31% - 0.99%	0.01	4.36% - 1.3%

Table E1.2. Worst case bounds for the simulation data described in Table E1.1.

Error measures			
RMSE (TE)[3]	Error range.	MAXAE (TE)	Error range.
0.0073	3.18% - 0.95%	0.01	4.36% - 1.3%

Table E1.3. Best case bounds for the simulation data described in Table E1.1.

[1]Highest RMSE value of sample.

[2]As a percentage of minimum and maximum values of test signal respectively.

[3]Lowest RMSE value of sample.

		Error measures					
No. of categories		RMSE			MAXAE		
ARTa	ARTb	TR	TE(NF)	TE	TR	TE(NF)	TE
798	62	0.0131	0.291	0.293	0.0871	0.0717	0.0679

Table E1.4. Mean results for fuzzy ARTMAP trained using a noisy data file of 1,000 items and tested using a noise-free data file also consisting of 1,000 data items.

Error measures			
RMSE (TE)	Error range.	MAXAE (TE)	Error range.
0.0304	13.25% - 3.95%	0.0698	30.41% - 9.06%

Table E1.5. Worst case bounds for the simulation data described in Table E1.4.

Error measures			
RMSE (TE)	Error range.	MAXAE (TE)	Error range.
0.0278	12.11% - 3.61%	0.0648	28.24% - 8.41%

Table E1.6. Best case bounds for the simulation data described in Table E1.4.

Simulation 2

No. of categories		Error measures	
ARTa	ARTb	RMSE	MAXAE
113	53	0.0175	0.0783

Table E2.1. Mean results of PROBART performance under the same conditions as those of simulation 1 and using the same noise-free data file.[4]

Error measures			
RMSE (TE)	Error range.	MAXAE (TE)	Error range.
0.0185	8.06% - 2.4%	0.085	37.04% - 11.03%

Table E2.2. Worst case bounds for the simulation data described in Table E2.1.

[4]Note that map field vigilance does not apply to PROBART simulations.

Error measures			
RMSE (TE)	Error range.	MAXAE (TE)	Error range.
0.0169	7.36% - 2.19%	0.0729	31.76% - 9.46%

Table E2.3. Best case bounds for the simulation data described in Table E2.1.

No. of categories		Error measures					
		RMSE			MAXAE		
ARTa	ARTb	TR	TE(NF)	TE	TR	TE(NF)	TE
111	62	0.0316	0.195	0.0206	0.1005	0.0815	0.0839

Table E2.4. Mean results for PROBART trained under the same conditions as simulation 1 using the noisy training file.

Error measures			
RMSE (TE)	Error range.	MAXAE (TE)	Error range.
0.0228	9.93% - 2.96%	0.0974	42.44% - 12.64%

Table E.2.5. Worst case bounds for the simulation data described in Table E2.4.

Error measures			
RMSE (TE)	Error range.	MAXAE (TE)	Error range.
0.0196	8.54% - 2.54%	0.0729	31.76% - 9.46%

Table E2.6. Best case bounds for the simulation data described in Table E2.4.

Simulation 3

No. of categories		Error measures	
ARTa	ARTb	RMSE	MAXAE
509	243	0.0015	0.0073

Table E3.1. Mean results obtained by PROBART using increased vigilance. Both training and testing were carried out using the same noise-free data file used in simulations 1 and 2.

Error measures			
RMSE (TE)	Error range.	MAXAE (TE)	Error range.
0.0016	0.7% - 0.21%	0.0084	3.66% - 1.09%

Table E3.2 Worst case bounds for the simulation data described in Table E3.1.

Error measures			
RMSE (TE)	Error range.	MAXAE (TE)	Error range.
0.0015	0.65% - 0.19%	0.0061	2.65% - 0.79%

Table E3.3. Best case bounds for the simulation data described in Table E3.1.

Simulation 4

		Error measures					
No. of categories		RMSE			MAXAE		
ARTa	ARTb	TR	TE(NF)	TE	TR	TE(NF)	TE
513	279	0.0193	0.0199	0.0197	0.0541	0.057	0.0566

Table E4.1. Mean results for PROBART trained using the noisy data set used previously in simulations 1 and 2 with parameters set as for simulation 3 i.e. $\rho_a = 0.999, \rho_b = 0.999$

Error measures			
RMSE (TE)	Error range.	MAXAE (TE)	Error range.
0.0206	8.98% -2.67%	0.0648	28.24% - 8.41%

Table E4.2. Worst case bounds for the simulation data described in Table E4.1.

Error measures			
RMSE (TE)	Error range.	MAXAE (TE)	Error range.
0.0189	8.23% - 2.45%	0.0498	21.7% - 6.46%

Table E4.3. Best case bounds for the simulation data described in Table E4.1.

Simulation 5

		Error measures					
Categories		RMSE			MAXAE		
ARTa	ARTb	TR	TE(NF)	TE	TR	TE(NF)	TE
1131	620	0.0265	0.0089	0.011	0.0814	0.0225	0.0426

Table E5.1. Mean results for PROBART obtained using the parameters of simulation 4 with the noisy training file increased to 10,000 items.

Error measures			
RMSE (TE)	Error range.	MAXAE (TE)	Error range.
0.0117	5.1% -1.52%	0.0472	20.57% - 6.13%

Table E5.2. Worst case bounds for the simulation data described in Table E5.1.

Error measures			
RMSE (TE)	Error range.	MAXAE (TE)	Error range.
0.0103	4.49% - 1.52%	0.0388	16.91% - 5.04%

Table E5.3. Best case bounds for the simulation data described in Table E5.1.

Appendix F The Cart-Pole Simulation

The cart-pole simulation was carried out as state in Barto *et al* (1983) with minor modifications. The state vector was reset to $x = \dot{x} = \theta = \dot{\theta} = 0$ after each trial; failure was indicated by a reinforcement signal of -1 when either the cart displacement, x or pole angle, θ left their ranges of [-2.4m, 2.4m] and [-12°,+12°] respectively. All trace variables were set to zero at the start of each trial. All weights were set to zero at the start of each run. Each run of the set of ten used random numbers from a different seed value. See Barto *et al* (1983) for further details.

The parameter values used for the ASE / ACE subsystems were a=0.8, b=0.5, δ=0.9 γ=0.98 and σ=0.01. Here, α and γ differ from the BSA implementation. As stated in the body of the text, the former was reduced substantially to prevent premature establishment of control actions. The latter was used to reduce the reinforcement prediction discounting but does not appear to have any significant effect; the change is noted here for completeness.

The simulation equations for the cart-pole system are the following non-linear differential equations (Barto *et al*, 1983):

$$\ddot{\theta} = \frac{g\sin\theta + \cos\theta \dfrac{-F - ml\dot{\theta}^2 \sin\theta + \mu_c \operatorname{sgn}(\dot{x})}{m_c + m} - \dfrac{\mu_p \dot{\theta}}{ml}}{l\left[\dfrac{4}{3} - \dfrac{m\cos^2\theta}{m_c + m}\right]} \qquad \text{(F1)}$$

$$\ddot{x} = \frac{F + ml\left[\dot{\theta}^2 \sin\theta - \ddot{\theta}\cos\theta\right] - \mu_c \operatorname{sgn}(\dot{x})}{m_c + m} \qquad \text{(F2)}$$

The parameters are those used in Barto *et al* (1983) with no changes, and the system is simulated using Euler's method with a timestep of 0.02 seconds. A control force is applied at every timestep until failure occurs. The neurocontroller only has access to the cart-pole system states in the form of a state vector. It does not have privileged access to a model or any pre-existing cost function.

Appendix G The EUCART Category Composition Theorem

The Category Composition Theorem requires several lemmas which will be developed here. First, a definition is required:

Definition: *Category Membership Property:*

An input vector, $\mathbf{x}_t \in X$ is a member of a category $C_t^{(i)}$ if it lies within a closed ball $B_t^{(i)} = \left\{\mathbf{p}: \left\|\mathbf{p} - \mathbf{c}_t^{(i)}\right\| \leq \rho\right\}$ where X is the input space, $\mathbf{c}_t^{(i)}$ is the category centre and ρ is the category radius.

The centre, $\mathbf{c}_t^{(i)}$ is determined by the category extent markers defined previously. This centre forms the centre of hyperrectangular category $H_t^{(i)}$ and hyperspherical category $S_t^{(i)}$ respectively.

Lemma 1: A hyperrectangular category, $H_t^{(i)}$, determined by the same category extent markers $\left(\mathbf{u}_t^{(i)}, \mathbf{v}_t^{(i)}\right)$ as a hyperspherical category $S_t^{(i)}$ is a proper subset of $S_t^{(i)}$ i.e. $H_t^{(i)} \subset S_t^{(i)}$.

Proof:

By definition of a subset, it has to be shown that

$\forall \mathbf{x}, \quad \mathbf{x} \in H_t^{(i)} \Rightarrow \mathbf{x} \in S_t^{(i)}$.

The centre of $H_t^{(i)}$ and $S_t^{(i)}$, $\mathbf{c}_t^{(i)}$ is determined by the category extent markers $\left(\mathbf{u}_t^{(i)}, \mathbf{v}_t^{(i)}\right)$ through the definition $\mathbf{c}_t^{(i)} = \frac{1}{2}\left(\mathbf{u}_t^{(i)} + \mathbf{v}_t^{(i)}\right)$. The longest diagonal (intervertex distance) of $H_t^{(i)}$ is between the category extent markers $\left(\mathbf{u}_t^{(i)}, \mathbf{v}_t^{(i)}\right)$ and is given by $\left\|\mathbf{v}_t^{(i)} - \mathbf{u}_t^{(i)}\right\|$. This distance forms the diameter of $S_t^{(i)}$ with the common centre $\mathbf{c}_t^{(i)}$ which lies on the diagonal. Figure G2 illustrates the situation in two dimensions.

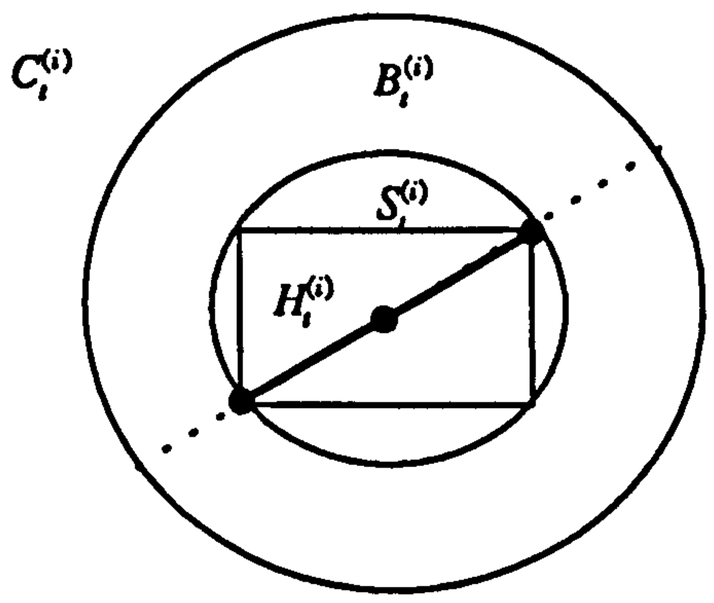


Figure G1 The hyperrectangle is contained within the inner hypersphere. Both are determined by the category extent markers and both lie within the outer hypersphere which denotes the maximum possible category extent given the current centre.

By definition, the radius of $S_t^{(i)}$ is given by $r = \frac{1}{2}\left\|\mathbf{v}_t^{(i)} - \mathbf{u}_t^{(i)}\right\|$. Because $\left\|\mathbf{v}_t^{(i)} - \mathbf{u}_t^{(i)}\right\|$ is the largest intervertex distance, the distance from any vertex of $H_t^{(i)}$ to the centre, $\mathbf{c}_t^{(i)}$ is always less than or equal to r. More formally, define $S_t^{(i)} = \left\{\mathbf{p}: \left\|\mathbf{p} - \mathbf{c}_t^{(i)}\right\| \le r\right\}$. The hyperrectangular category $H_t^{(i)}$ is determined by the category extent markers $\left(\mathbf{u}_t^{(i)}, \mathbf{v}_t^{(i)}\right)$which form two of the vertices. The largest intervertex distance which is divided into two equal parts by the common centre, $\mathbf{c}_t^{(i)}$ determines the largest possible distance between any input vector, $\mathbf{x}_p \in H_t^{(i)}$ and the common centre. So, $\left\|\mathbf{x}_p - \mathbf{c}_t^{(i)}\right\| \le \frac{1}{2}\left\|\mathbf{v}_t^{(i)} - \mathbf{u}_t^{(i)}\right\| = r$ and $\mathbf{x}_p \in S_t^{(i)}$.

Therefore, $\mathbf{x}_p \in H_t^{(i)} \Rightarrow \mathbf{x}_p \in S_t^{(i)}$ for some arbitrary point, $\mathbf{x}_p$. Thus,

$\forall \mathbf{x}, \quad \mathbf{x} \in H_t^{(i)} \Rightarrow \mathbf{x} \in S_t^{(i)}$, and by definition, $H_t^{(i)} \subset S_t^{(i)}$ as required. ■

Lemma 2: for any input $\mathbf{x}_t \in X$ belonging to a category $C_t^{(i)}$, $\mathbf{x}_t$ either lies within a contained hyperrectangle, $H_t^{(i)}$ or the complement of $H_t^{(i)}$ with respect to the closed ball, $B_t^{(i)}$, determining the maximum possible extent of that category at time instant, t.

Proof:

By lemma 1, $H_t^{(i)} \subset S_t^{(i)}$. The maximum possible extent of $S_t^{(i)}$ at time instant, t is $C_t^{(i)} = B_t^{(i)}$ so, $S_t^{(i)} \subseteq B_t^{(i)}$ and thus, $H_t^{(i)} \subset S_t^{(i)} \subseteq B_t^{(i)}$. The sets $H_t^{(i)}$ and $B_t^{(i)} \cap \left(H_t^{(i)}\right)^c$ form a partition of $B_t^{(i)}$ because $H_t^{(i)} \cup \left[B_t^{(i)} \cap \left(H_t^{(i)}\right)^c\right] = B_t^{(i)}$ and $H_t^{(i)} \cap \left[B_t^{(i)} \cap \left(H_t^{(i)}\right)^c\right] = \Phi$.

So, for any $\mathbf{x}_t \in C_t^{(i)} = B_t^{(i)}$,either $\mathbf{x}_t \in H_t^{(i)}$ or, $\mathbf{x}_t \in B_t^{(i)} \cap \left(H_t^{(i)}\right)^c$. ■

Lemma 3: hyperrectangular category growth is monotonic. Formally,
$H_t^{(i)} \subseteq H_{t+1}^{(i)}$

Proof:

For the category extent markers, $\left(\mathbf{u}_t^{(i)}, \mathbf{v}_t^{(i)}\right)$ at time, t, the following notation is used: let $u_{k,t}^{(i)}$ denote the k th component of $\mathbf{u}_t^{(i)}$, and let $v_{k,t}^{(i)}$ denote the k th component of $\mathbf{v}_t^{(i)}$. By definition of the category extent markers, $\forall k, \quad u_{k,t}^{(i)} \leq v_{k,t}^{(i)}$. At the next time instant, t+1 the category extent markers are denoted by $\left(\mathbf{u}_{t+}^{(i)}, \mathbf{v}_{t+1}^{(i)}\right)$. If the new input, $\mathbf{x}_p \in C_t^{(i)}$ but $\mathbf{x}_p \notin H_t^{(i)}$ then, by the definition of sub-section 3.3.2.5, the category extent markers extend to include the new input in the hyperrectangular category at time t+1. Let $x_{k,p}$ indicate the k th component of the p th pattern vector. The following three cases cover all possible relationships between the input vector components and the category extent marker components:

i) $x_{k,p} < u_{k,t}^{(i)}$

ii) $u_{k,t}^{(i)} \leq x_{k,p} \leq v_{k,t}^{(i)}$

iii) $v_{k,t}^{(i)} < x_{k,p}$

By definition of the category extent marker update of sub-section 3.3.2.5 , the three cases give the following:

i) $u_{k,t+1}^{(i)} = x_{k,p}$

ii) no change

iii) $v_{k,t+1}^{(i)} = x_{k,p}$

so, at time t+1, the following inequalities must hold:

i) $u_{k,t+1}^{(i)} = x_{k,p} < u_{k,t}^{(i)}$

ii) $u_{k,t+1}^{(i)} = u_{k,t}^{(i)} \le x_{k,p} \le v_{k,t}^{(i)} = v_{k,t+1}^{(i)}$

iii) $v_{k,t}^{(i)} < x_{k,p} = v_{k,t+1}^{(i)}$

Define closed real intervals $A_{k,t}^{(i)} = \left[u_{k,t}^{(i)}, v_{k,t}^{(i)}\right]$ such that

$$H_t^{(i)} = A_{1,t}^{(i)} \times \ldots \times A_{k,t}^{(i)} \times \ldots \times A_{N,t}^{(i)} = \prod_{k=1}^{N} A_{k,t}^{(i)}$$

$A_{k,t}^{(i)} \subseteq A_{k,t+1}^{(i)}$ which implies that $\prod_{k=1}^{N} A_{k,t}^{(i)} \subseteq \prod_{k=1}^{N} A_{k,t+1}^{(i)}$

or, $H_t^{(i)} \subseteq H_{t+1}^{(i)}$ as required. ■

The EUCART Category Composition Theorem: All inputs that are members of a given hyperspherical category remain within that category throughout the category growth process and beyond.

In other words, for an input space (state-space), X

$\forall \mathbf{x}_\tau \in X$ for some time, τ and some hyperspherical category, $C_\tau^{(i)}$ at time τ,

$\mathbf{x}_\tau \in C_\tau^{(i)} \Rightarrow \mathbf{x}_\tau \in C_{\tau+n}^{(i)}, \quad n \to \infty$

Proof:

$\mathbf{x}_t \in X$ at time, t. Without loss of generality, assume that a new category is not created. $\mathbf{x}_t$ belongs to some category $C_t^{(i)}$ at time, t.

By the category membership property and lemma 2, $\mathbf{x}_t \in C_t^{(i)}$, implies that

$$\mathbf{x}_t \in H_t^{(i)} \subset S_t^{(i)} \subseteq B_t^{(i)} = C_t^{(i)}, \text{ or } \mathbf{x}_t \in B_t^{(i)} \cap \left(H_t^{(i)}\right)^c.$$

If $\mathbf{x}_t \in H_t^{(i)}$, then by lemma 3

$$\mathbf{x}_t \in H_t^{(i)} \subseteq H_{t+1}^{(i)} \subset C_{t+1}^{(i)}.$$

If, however, $\mathbf{x}_t \in B_t^{(i)} \cap \left(H_t^{(i)}\right)^c$ then, by the category growth process, $\mathbf{x}_t \in H_{t+1}^{(i)} \subset C_{t+1}^{(i)}$. Either way $\mathbf{x}_t \in C_{t+1}^{(i)}$ giving $\mathbf{x}_t \in C_t^{(i)} \Rightarrow \mathbf{x}_t \in C_{\tau+1}^{(i)}$

By definition, $\mathbf{x}_0 \in C_0^{(i)}$, so by the principle of mathematical induction,

$\mathbf{x}_\tau \in C_\tau^{(i)} \Rightarrow \mathbf{x}_\tau \in C_{\tau+n}^{(i)}$, $n \to \infty$ as required ■