University of Southampton

A Phenomenological-Connectionist Theory of Computational Agency

Volume II of II

by

Daniel W. Joyce

A thesis submitted for the degree of
Doctor of Philosophy

in the
Faculty of Engineering and Applied Science
Department of Electronics and Computer Science

April 2001

University of Southampton

ABSTRACT

Faculty of Engineering and Applied Science

Department of Electronics and Computer Science

Doctor of Philosophy

A Phenomenological-Connectionist Theory of Computational Agency

Volume II of II

by Daniel W. Joyce

This thesis presents a theory of computational agency. Computational agent theory differs from 'classical' artificial intelligence by committing to the view that a computational artifact is situated, and that its rationality is limited by the constraints of this 'situatedness'. Contemporary literature is surveyed and models of situated computational agency placed in their philosophical contexts. From this, a critical reconstruction of the notion of an agent is given from a phenomenological perspective. It is proposed that everyday *routines of activity* underpins agency and computational implementations of this substrate can take the form of connectionist networks.

The proposal is tested in technical practice on two domains; agents for multimedia content-based navigation/retrieval, and a simulated environment which explores the key properties of the proposed phenomenological agent theory. Recent proposals for goal-directed behaviour in connectionist systems (largely from the cognitive and behavioural neurosciences) are critically evaluated, and integrated into an agent architecture. This results in an architecture utilising suitably controlled reinforcement learning. The architecture implemented is then evaluated against the agent theory, and examples of 'routine behaviour' analysed in stationary and non-stationary environments.

Semiotic analyses are then proposed as an alternative theory of representation, as they are compatible with, and simultaneously possess explanatory power at a level beneath, the usual sentential/propositional level.

The thesis contributes a phenomenological theory of agency and gives examples of its influence on technical practice. Outlines of connectionist architectures are presented, implemented, and evaluated with respect to the agent theory proposed.

University of Southampton

Intelligence, Agents and Multimedia Research Group, Department of Electronics and Computer Science

Technical Report No. ESTCR-IAM004

Modifying Situated Automata Theory for Connectionist Agents

Dan W. Joyce

April 2001

ABSTRACT

This technical report describes the use of Rosenschein and Kaelbling's situated automata theory (SAT) in the context of connectionist-based agents. The philosophical commitments implicit in the approach adopted in SAT include a strong epistemic constraint; namely that information content is defined as internal state correlated with external events, and propositions being truth-conditional on this correlation. It is argued that this requires modification for agents where internal state is adapted over time. The report concludes by examining solutions to problems that arise when automata-based models are used to characterise agents.

Contents

1 Introduction

There are currently a variety of theoretical constructions predominant in agent theory. For example, those grounded in practical reasoning (Rao and Georgeff, 1995), formal logic and "purely reactive" systems e.g. (Brooks, 1986). A more thorough review can be found in (Wooldridge, 1999). Arguably, the philosophical commitment underpinning these methods (with the exception of Brooks' work) is a formal system of symbolic computation. As Agre (1997) argued, this has led to an understanding of agency, intelligence (natural and artificial) and behaviour as predicated on assumptions derivative from those in computation; for example, optimality, perfect rationality, and internal representation.

In (Agre, 1995; Agre, 1997), the notion of interactionism as an agent theory was espoused. This concept, where agent epistemology is grounded in everyday interactions with the environment, has been formalised in the situated automata theory of (Rosenschein and Kaelbling, 1995). Interactionism is, we might say, pre-ontological; it begins by asking questions of the notion of agency that ignore the extant examples of artificial agency. While Agre and others then use computational frameworks to realise an instantiation of the agent theory, the resulting design process and artefacts have quite different shapes.

The interactionist approach contrasts with previous agent theories in its explicit attempt to locate epistemology across the usual agent/environment divide (rather then in either the agent or environment). While the agent and environment are still identified as being demarcated by a "physical" boundary, the engineering of agents derivative from such an interactionist stance forces the designer to consider minimalist, necessary and functional representations which enable action in context, rather than striving for abstractions which generalise to (say) a predicate or modal logic.

As an example, in an imaginary blocks microworld, an interactionist design might find that a necessary representational token is $\texttt{the}-\texttt{red}-\texttt{block}-\texttt{in}-\texttt{front}$. The motivation for this token would be that it is necessary for a certain routine interaction (e.g. a regularity in the agent/environment coupling). Therefore, it is said to be *functional.* Agre also describes representations which are *indexical*, meaning that their apparent specificity, they transfer to a variety of situations the agent might encounter. In the example above, a primitive indexicality is the detachment of the functional representation from a specific temporal location, so whenever their is a red block in front of the agent, the same representational token will facilitate routine activity. In contrast, the traditional generalisation being to produce a predicate, such as :

$$\texttt{in}-\texttt{front}(a,b) \tag{1}$$

$$\texttt{colour}(x,y) \tag{2}$$

where, to represent the functional token $\texttt{the}-\texttt{red}-\texttt{block}-\texttt{in}-\texttt{front}$ the necessary bind-

ings for the variables would be:

$$a = \texttt{block} \tag{3}$$

$$b = \texttt{me} \tag{4}$$

$$x = \texttt{a} \tag{5}$$

$$y = \texttt{red} \tag{6}$$

In this report, it is proposed that interactionism might capture the essential qualities of connectionist-based agents (i.e. where the "internal mechanism" of the agent is a connectionist network, such as a multi-layer perceptron or some more sophisticated architecture). This is because connectionist (at least prima facie) appears to adhere to the conceptual basis of interactionism rather than the alternative symbolic logic.

Situated automata theory (SAT) is a valuable tool for reasoning about the properties of such agents, but it will be argued, it is philosophically bound to assumptions of symbolic computation; namely, provable properties such as tractability and verifiability. Such properties of a formal theory which constrain the implementation are clearly desirable – see for example the discussion of (D'Inverno, Fisher, Lomuscio, Luck, De Rijke, Ryan and Wooldridge, 1997). The intentions of SAT are that after defining an agent in a broadly interactionist fashion, one is able to derive an implementation of the agent (a realisation). The separation of specification and executable machine enables a theorist to adopt SAT without actually committing to implementation.

Ideally, SAT would model connectionist agents wholesale, however, the epistemological commitments in SAT are too "strong" to enable its use without modification. In modifying SAT to cope with connectionist agents (or, more generally, any adaptive system), it is necessary to abandon the strong epistemelogical constraints. Notably, this is because of the dynamic nature of correlating internal state in connectionist agents. Finally, conclusions will be drawn on the utility of modified SAT as an agent theory.

2 Background

Recently, Wooldridge (1999) described a framework and overview of intelligent agents. Further, he included a typology of agents which includes reactive systems. Throughout, reference is made to an abstract agent framework which is broadly similar to those used in reinforcement learning, situated robotics and other areas which (while not mainstream software agents) certainly fall under the broad category of agents research. Both Wooldridge (1999) and (Rosenschein and Kaelbling, 1995) still attempt to unify knowledge, behaviour and uncertainty in a fundamentally symbolic way (using his perception, visibility and knowledge calculus).

It is for this reason that this technical report begins with an automata-oriented view of agency. It then moves on to amplify the relationship between automata, pattern matching and

the essential features of reactive behaviour. This may seem far removed from any connectionist or artificial neural network discussion but it is interesting to note that historically, the McCulloch-Pitts model was given an automata interpretation by Arbib (1964) and von Neumann's seminal work on reproducing machines (Burks, 1966) (viz. populations in a multi-agent system) were also automata-theoretic. The pattern matching ability of connectionist networks and the similar ability of a finite state acceptor suggest that this analogy for reactive agency is worth pursuing with regards to suitably defined behaviour.

An agent is often said to perceive and effect actions in the environment. The typical "perception-action" or sensory-motor loop engages the agents in a direct relationship with their environment (Sharkey and Heemskerk, 1997; Neisser, 1976). Before formalising the details a first approximation serves to introduce the central notions. The behavioural details will be presented in two parts :

- *distal* or *extensional* details which describe the behaviour of the agent without recourse to internal mechanisms (often, in terms of an observer)
- *proximal* or *intensional* descriptions which detail the internal mechanisms of the agent's behaviour

Figure 1 shows the agent A in an environment, where a property of the environment p can be sensed by the agent – that is, the agent is suitably equipped to detect the property in a certain modality. Extensionally, the agent effects a response R (corresponding to an action) in the environment. From such distal observations, we may perhaps predict behavioural regularities based on assumed correlations between properties of the environment and observed actions taken by the agent. Note that here the observer might employ ascription of properties (McCarthy, 1978) and intentional stance (Dennett, 1978; Dennett, 1987) as explanatory tools. Also, there is no commitment to internal state or the nature of representations. By analogy with psychological behaviourism, we can only rely on our inductive explanation of the agent's behaviour without invoking notions of representations of the world or cognitive states.

If we resort to the intensional or proximal description, we have a causal explanation of the relationship between stimuli and elicited responses. One possible method for specifying this causal relationship is by positing internal states which (functionally) contribute to the elicitation of responses. The acceptance of internal state being causally related to behaviour is compatible with functionalist philosophy (Block, 1980), but still, does not commit to any *realisation* of internal state. For example, qualitatively, we can propose that a state of "disatisfaction" drives an agent to change its behaviour without specifying the mechanisms (biological or otherwise) that produce the behaviour we associate with disatisfaction. This enables any number of causally inert interpretations to be assigned to the internal state. A differing interpretation (in the observer) has no effect on the efficacy of the functional internal state.

An example of such a commitment to realisation would philosophically correspond to representational theories of mind (Fodor, 1998; Heil, 1998) such as the language of thought

(sentential-symbolic) and its technical counterpart of formal epistemic logics. Alternatively, a connectionist explanation would identify transient patterns of signalling behaviour in artificial neurons with world properties. Either is a realisation (or substrate) of internal state.

The *situated automata theory* (SAT) complements the model presented here by similarly dividing the environment and agent while contributing to the notion of internal state, but making no further commitments.

3 Related Work

Automata theories can be found (in the context of situated reactive agency) in the work of Rosenschein and Kaelbling (1995), where their situated automata theory (SAT) combines both models of the environment and the agent as finite (but sometimes nondeterministic) automata. A similar approach, but motivated by the philosophy of Heideggarian AI, can be found in the work of Agre and Horswill (1997) which concentrates on agent-environment interaction as a finite state model, instead of separating agent and environment and modelling them individually. In both cases, automata are engaged as a means of characterising the agent-environment interactions. Wilson (1991) uses a a Mealy machine (identical to the one used here) but inverted so that the environment is modelled and the input is the agent's motor action and the output is stimuli to the agents perceiving the environment – in essence, behaviour is defined as a function of structure in the environment.

Finite state automata (FSA) feature in earlier work such as Arbib's *modular net* formalisation of the McCulloch-Pitts formal neuron as a FSA (Arbib, 1964). Indeed, Arbib has been a keen advocate of the *multiple levels of explanation* approach to modelling cognition and agency; see for example, (Arbib, 1989) and (Arbib, 1995) where he describes control and automata theory as higher level explanations of fundamental modular units in cognitive processes. Brooks (1986) used finite state machines as the fundamental task achieving behaviour modules in the subsumption architecture because of their analytical simplicity. This is congruent with an automata theory of agency.

However, connectionist networks possess a significant property which defeat this proposal. If the neuronal model dynamics can be described by FSA, in addition, connectionist networks have modifiable connections (weights or idealised synapses) which reconfigure the functional relationships between the neurons.

Agents in modelling and simulation also have a traditional association with automata models. Here, instead of the models posessing any explanatory power in a conjectured theory (as Arbib's automata do), they are convenient models of observed phenomena in artificial settings that enable agent or multi-agent experimentation. For example, (MacLennan and Burghardt, 1994) presents a study of the evolutionary emergence of simple linguistic ability in a multi-agent system, where each agent (called a *simorg*) is a finite state automaton. From here on, the Mealy automaton will be used, related to SAT and then modified to cope with the demands of

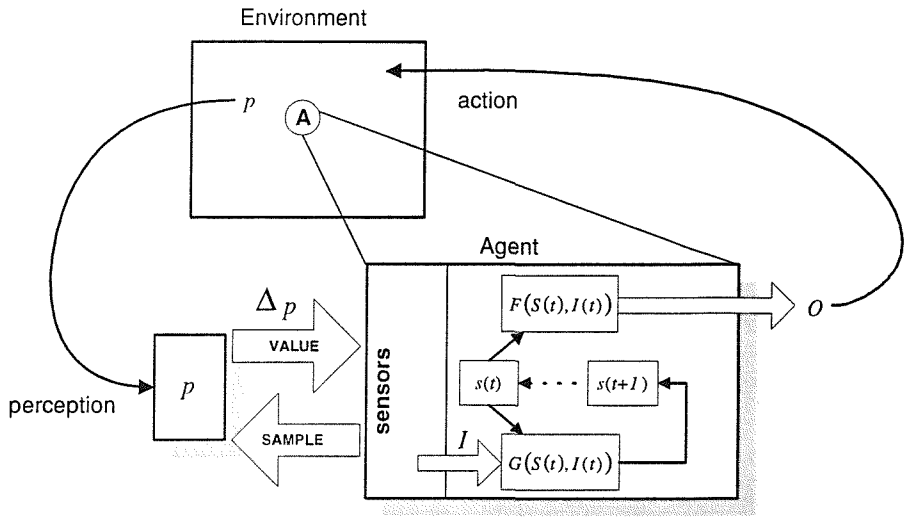

Figure 1: The Agent-Environment Relationship

connectionist implementations of long-term memory.

4 A Simple Automaton Model of an Agent

A suitable first approximation for agent behaviour can be easily visualised as an finite state automaton (FSA) such as that shown in Figure 1. We define the following Mealy machine (Sampson, 1976) as:

- a finite set of *input states* $I = \{\imath_i \mid i \in \mathbf{N}\}$
- a finite set of *output states* $O = \{o_j \mid j \in \mathbf{N}\}$
- a finite set of *internal states* $S = \{s_k \mid k \in \mathbf{N}\}$
- a function relating current internal state $s(t) \in S$ to $s(t+1)$ as $G : S \times I \rightarrow S$
- a function relating current internal state $s(t) \in S$ to an output state $o(t) \in O$ as $F : S \times I \rightarrow O$

In a SAT context, the agent's internal mechanism could be interpreted from the Mealy model above. The set I will correspond to a countably finite set of stimuli incident on the agent and induced by the environment. The sensor "reporting" the presence of a certain environmental feature consistutes a member of I. The output states O will correspond to actions in the environment, for example, a set of action primitives such as "approach food". The internal states S will simply be transient states used to functionally correlate the inputs to outputs.

However, the two functions G and F are more interesting. They effectively decompose the behaviour of the agent into two mechanisms : G transforms internal state according to perception of environmental events, and F maps internal state to actions. We see that internal state exists simply as a repository for state that allows a causal association between the input and output (collectively, the behaviour) of the agent. This technical definition is philosophically compliant with decompositional functionalism cf. (Block, 1995; Block, 1980).

The technical utility of this model is that it naturally provides for a reactive model of agency. The use of the term "function" in its declarative mathematical interpretation (representing, in effect, shorthand notation for a relation between two sets) suggests an explicit mapping from I to O with some functionally intervening states. Again, we are not forced to commit to a purely reactive philosophies. The description afforded by a Mealy machine and its relation to pattern matching provides an initial insight into the conjecture that reactive behaviour and pattern matching are isomorphic. The entire behaviour of the agent is elegantly partitioned by the model. The function G is effectively perception, and the function F action.

5 Stimuli

In order to discuss the general notion of behaviour for reactive agents, some working definitions must be introduced. No one definition of behaviour, stimulus or response will suffice, and terminologies differ depending on discipline.

Here, the definitions chosen are carefully selected to correspond with the types of modelling activity likely to be engaged in when designing or specifying an agent's behaviour. A concise and relatively unambiguous set of definitions is offered by Catania (1992) and congruent with (Arkin, 1998):

- stimuli : events occuring in the environment
- response : an action defined in relation to the environment
- behaviour : a regularly observed causal relationship between a class of stimuli and a class of responses

It is important to note that a behaviour is not an *explanation* of the causal relation; it is simply a label denoting that one exists.

5.1 Preliminary Definitions

There are two concerns in defining a stimulus :

- structural issues about how events in the environment map to percepts in the agent
- functional issues concerning the existence criteria for a stimulus with respect to the agent and environment

Here, we will adopt the "softbot" model (Etzioni, 1993) of an agent. An agent is equipped with sensors which transduce physical (or virtual) phenomena incident on the agent. The environmental event (e.g. a light source becoming illuminated) produces a change in the environment (the intensity of ambient light in a region of space increases) which is only accessible to an agent if it is equipped with the necessary sensor. The distinction is two fold; an environmental event and the transduction of the event's consequences to the agent.

A structural definition of a stimulus would include the two fold distinction; that is, the environmental events exist in some objective world (and therefore, so do stimuli). Their existence is unquestionable, but the agent may not be equipped to interpret (or receive) them. However, such apparent objectivity is phenomenologically unsound as well as being technically intractable; if an agent cannot sense the modality within which the environment has changed (e.g. light intensity) then the agent cannot be aware of this change. This is an instance of structural coupling, where the agent and environment are mutually defined by functional relationships. This notion of a stimulus defined with respect to the mechanics of the agent's perceptual machinery is similar to Dretske's notion of systemic representation – that which is produced by a certain environmental event causing activity in an agent's perceptual machinery; it has no "content" (primitively, semantics) because it is not functionally in use for a particular activity which Dretske argues is the origin of meaning (Dretske, 1995; Dretske, 1985).

An incident stimulus is transduced by the function G and this representative value might reasonably be called a perceptable stimulus if the sensor responds (for example, if the stimulus occurs above a threshold level denoting the sensitivity of the sensor in a physical agent (Arkin, 1998) pp. 91). Therefore, we might reasonably define a stimulus to have occured if and only if a sensor has changed in response to a conjectured change in the environment. For the time being, we ignore the case of misrepresentation – e.g. see (Dennett, 1987) – where an incorrect percept is produced by the agent's perceptual machinery (at least as far as design intentions are concerned).

This functional definition is alluded to in Figure 1 where we define some environmental property p. A change Δp is transduced by a suitable sensor on the agent. This results in a stimulus being produced $\iota \in I$ which is shown as if it were produced by the agent's sensing faculties. This stimulus is then an influence on the function G which yields the next internal state (loosely, the agent's perception of the current environment). The function F subsumes all other internal mechanisms and state to produce a response. In this scheme, the stimulus exists only as an environmentally induced "cue" for an automata-like theory of an agent's internal mechanics. This enforces the treatment of stimuli as being functionally inseperable from the agent's embodiment (i.e. the arrangment and specification of its sensors). A similar supporting definition is given by Rosenschein and Kaelbling (1995) as their *correlational definition of information.*

Definition 5.1 *A perceptable stimulus is defined as* $\iota \in I$ *resulting from the agent sensing the*

environment where the modality of the sensor corresponds to some property of the environment.

5.2 Stimuli, Internal State and Correlation

The assumption in SAT is that some percept induces an internal state which co-occurs with the truth-functional semantics of a proposition about the environment. For example, the truth-functional semantics of a proposition such as "a tree has fallen" is objectively dependent on the environment. If an agent can sense this, and there is an internal state $s \in S$ that co-occurs with the stimulus, then Rosenschein and Kaelbling assert that the agent *carries the information* "a tree has fallen". This is an example of content-ascription semantics, see (Dennett, 1987), or "as-if" intentionality (Roy, Petitot, Pachoud and Varela, 1999) i.e. the agent behaves "as if" it carried the information. Note this is somewhat different to Dretske's approach, which only defines a content-carrying state in the context of its use. So in Dretske's framework, if the agent has a sensor and it detects falling trees, and this causes some tree-avoiding behaviour, then the sensor is said to systematically represent the *motion of objects* and the use gives it the *acquired* meaning of detecting tree-falling in the service of avoiding falling trees.

The model developed here differs from SAT, in that correlational information has a *strong* epistemic condition relating ι induced by an environmental event Δp and internal state s. Using the definitions developing here, we restate the condition from (Rosenschein and Kaelbling, 1995):

Definition 5.2 Strong correlation *: For an environmental property p that undergoes some change (i.e. the event) Δp, a proposition about the world ϕ is true. An agent A with internal state s carries information ϕ iff whenever ϕ is true, the agent is in internal state $s \in S$.*

For example: at some time t, if stimulus $\Delta p(t)$ occurs and corresponds to the proposition ϕ ="the light is on", then for an agent to carry this information, we must find that :

$$\exists! s' \in S \mid \phi = true, s(t) = s' \tag{7}$$

So, whenever the stimulus $\Delta p(t)$ occurs we can assert that the agent carries or knows the information "the light is on". The property of the environment p is related to internal state by the functional relationship $G(s,\iota)$ which we have coarsely defined as the agent's perceptual apparatus. This is shown in Figure 2. Note that ascription is "realised" by an internal state, but does not *cause* the ascription. Effectively, the proposition ϕ is causally inert under the ascriptional scheme e.g. intentional stance. A similar notion is present in some BDI models. For example, (de Lioncourt and Luck, 1999) is an instantiation of a BDI architecture. The beliefs, desires and intentions are all ascriptions to lower level mechanisms and equally are inert in actually causing actions. That is not to say that the internal state *itself* is causally

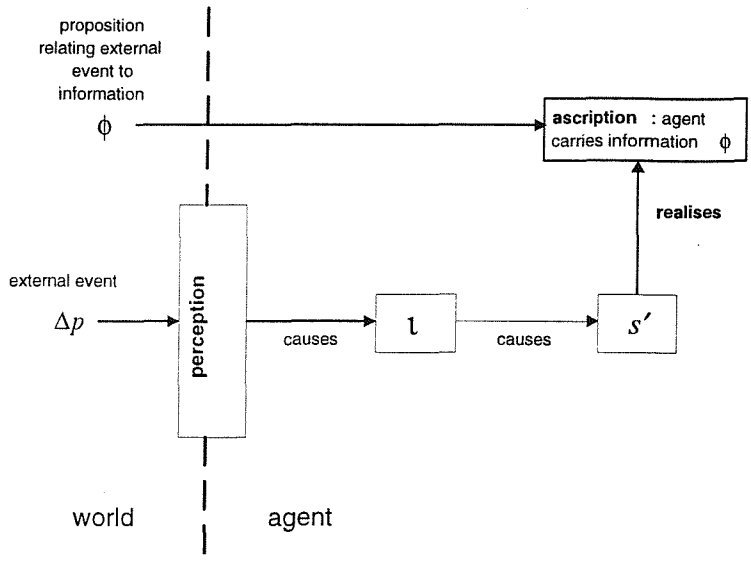

Figure 2: Strong Correlation : Causation and ascription shown in relation to agent internal state and environment

inert; by functionalism, it must be true that the internal state is causally implicated in causing a response. The *content* is only ascriptional and therefore inert.

From this, we can partially resolve the apparent conflict between the knowledge-based approaches of deliberative agency, and the situated reactive approach (as SAT is intended). SAT is often described as a model-theoretic approach to engineering agents; that is, using formal logic in the Tarskian sense of a "model" following from Frege's notion of sense (Agre, 1997). Each statement in a model is assigned a truth interpretation, and the truth of consequent or composed sentences (e.g. those built from the original sentences or deduced by the application of theorems or axioms) depends on these original truth assignments (compositional and truth-functional semantics). (van Gelder, 1990) calls this concatenative compositionality; the composition (and truth functional semantics) relies on concatenating atomic propositions such as $P \vee Q$.

If we take internal state and the ascription of information such as ϕ to be a mentalistic property of the agent, then there is an analogy with reductive explanations of the mental and the identity theory and co-occurence of physical properties and mental properties see (Smart, 1959; Chalmers, 1996; Heil, 1998; Block, 1997). However, just as philosophical challenges have been made to the identity thesis and reductive explanations, here the strong correlation definition must be weakened accordingly.

In its simplest form, the identity thesis states that a physical property P is a mental state M. This is due to the work of (Smart, 1959) in attempting to establish the physical basis of mental events. However, it is too simplistic to be plausible. Instead of *identifying* M with

P, attempting to reduce M to a set of properties P requires that there are bridge laws, which relate the physical realisers P with actual mental events. (Kim, 1998) pp.218 cites the popular example of taking neural activation in certain structures of the brain to be P and the mental property to be "in pain". We can either identify P with M (to be "in pain") or attempt to find correlations (certain P that co-occurs with M) and then provide bridge laws which explain why P gives rise to M.

A compelling reason to reject the identity and the reductive thesis in the cognitive sciences is the notion that such a strong correspondence is impossible to empirically validate because it is hard to identify unique physical states which correspond with mental states. An analogy is if the internal state s is "pain" caused by the agent impacting on a sharp object, then under the identity thesis and the Rosenschein and Kaelbling definition of correlative information, only then would the agent carry the information ϕ ="sharp object encountered".

The difference between Smart's project and SAT is that the former attempts to reverse engineer a complex system, and the latter to forward engineer a system to undertake certain tasks. Our argument is about content-ascription in artificial agents. We must put to one side the nature of content (e.g. its origin in artificial systems) because it is controversial; some have argued – e.g. (Dennett, 1987) – that if an agent A is complex enough to require another agent B to assume intentionality, then this is reason enough to assume the agent A possesses original intentionality. Dennett also argues that under his intentional stance, Searle's description of original and derived intentionality is artificial. Whether or not an artificial agent can possess something like original intentionality is not directly relevant here.

What remains relevant is how agent epistemology can be grounded without an identity thesis-like commitment. The reasons given above (i.e. the locating of unique realising states) hold for an adaptive agent. In an adaptive system, such identities will be equally difficult to find, not least because the percept coded for by a set of activities over neuron groups (fields or pools) are functionally dependent on weights which change over time. Effectively, the kind of discretisation necessary to implement strong correlation removes the possibility of accounting for adaptive systems and forces the interpretation of internal states as content-ascriptional.

Having separated the technical and philosophical use of the idea of correlational information, we can procede with the argument for weakening the correlation condition. Intuitively a number of states could justify the agent carrying the information ϕ. This means the agent can be in a variety of states $\{s_1 \vee s_2 \vee \ldots \vee s_{k'}\}$ each of which can potentially justify the ascription of ϕ. This problem was tackled philosophically by the introduction of not just a single machine/automaton state realising the agent carrying or knowing ϕ, but a disjunction of a number of states might co-occur in the agent and world. For the model here, we borrow the notion of heterogeneous disjunctions e.g. see (Kim, 1998), where a number of distinct states might warrant the ascription of ϕ to the agent's epistemological state.

The justification is as follows : If an agent A must always be in internal state s simultaneously with the environmental state or property p so that the correlational information ϕ is

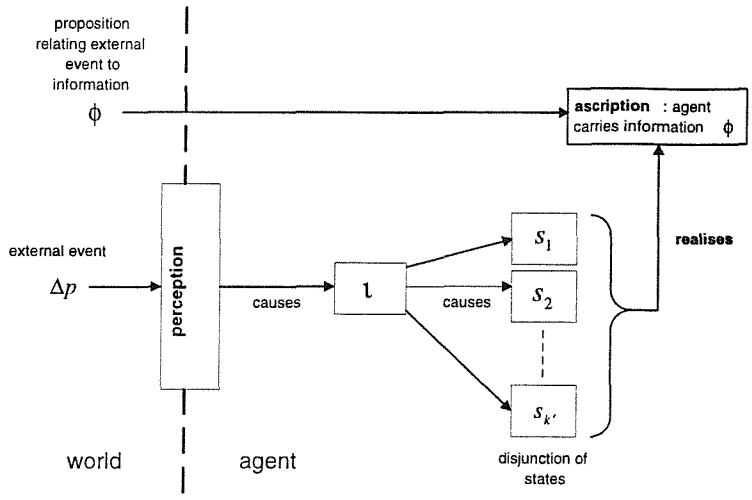

Figure 3: Weak Correlation : Causation and ascription shown in relation to agent internal state and environment

implied and carried by A, then how can we account for a non-deterministic agent or one that changes it's behaviour as a result of adaptation to an environment, where some new internal state s' now co-occurs with or instead of s ? The information carried by the agent as ϕ is now no longer a simple correlation. An agent might, therefore, need a kind of heterogeneous disjunction which states a weaker kind of correlation. This is shown in Figure 3.

Definition 5.3 Weak correlation *: An agent carries information ϕ whenever the agent's internal state $s(t) \in S'$ where $S' \subset S$ and states S' co-occur with $\phi = true$*

So, the identification of a unique state and information carried by agent is relaxed so that :

$$\exists s' \in S' \mid S' \subset S, \phi = true, s(t) = s' \tag{8}$$

Note that it is an open question (both in the salvaging of the identity thesis and technically for an automata theory of reactive agents) whether or not the disjunction really qualifies as a state in itself.

The philosophical solution of the heterogeneous disjunction is debated because *any one* of the set of states $s_i \in S$ could realise the mental state M. The argument halts because this set is potentially infinite, and necessarily so, because how can we possibly guarantee that every possible realiser of M is captured by the set S ? (Kim, 1998) states that there "isn't likely to be a single scientific theory that can deal with the hopelessly heterogeous disjunction ..." pp. 220.

This diverse debate is a necessary component of the agent theory presented here because we wish to explicitly account for adaptive behaviour in sub-symbolic processes like artificial neural networks. In this case, SAT cannot be applied as either a conceptual framework or an engineering principle (enabling, say, specifications in a model theory and automatic derivation

of an automata to produce the required reactive behaviour) because it seems not to cope with the heterogeneous disjunction of internal states. In fact, the provable properties of a SAT-based model *demand* strong correlation.

However, the ascription of (causally inert) propositions using weak correlation is an intuitive principle for adaptive agents employing sub-symbolic processes. In fact, it can be argued this is precisely the kind of ascription used when (for example) interpreting a neural network classifier. The structural interpretation is the weights (artificial idealised synapses) connecting nodes (artificial neurons) with certain efficacies or strengths. The transient activities of neurons during the processing of an input are analogous to internal states arising as a result of the input (stimulus). However, these properties of the system only take on meaning when used in the task of assigning classifications to inputs. Therefore, the weights and the nodes of an adaptive artificial neural network take on qualitative meaning by ascription in the service of the task of, say, classification.

6 Correlation and Control State

Recall that the key difference between a classic epistemic representation system (in a formal language such as predicate calculus and implementation in an expert system) and an agent is its condition of being situated in its environment.

This suggests recourse to a different but complementary notion of representation. Sloman (1996) calls the kind of theory of representation most useful here to be *control state*. This has been alluded to in this technical report, but not explicitly introduced as a means for discussing agent internal state.

What has been described this far is a notion of *pragmatics* of a control state. The internal states mentioned exist and cause the agent to take certain actions. This is the control function. However, we have (to be compatible with agent theories generally) attempted to usefully apply intentional stance and ascriptional properties to these states also. The former control function is what Sloman calls the pragmatics of control state. Sloman states that "This notion of function or purpose does not require a human designer or any conscious intention" pp. 127. However, nearly all agent theories (especially BDI-inspired systems) require something more in respect of correspondance of these states with the world. Take a simplistic expert system built in Prolog. The control states exist, those intemediate states (e.g. propositions, data structures such as trees, etc.) which contribute to the process of, say, theorem proving. These are the pragmatics of control states. However, some of those states are engineered to stand in a denotative correspondance with the world – hence have some kind of function which maps them to concensually agreed (usually, in the culture and language of mathematics) interpretations in an epistemic sense. Sloman calls such mappings the syntax or grammar of the states. Similarly, Dretske adopts a division which separates mechanical properties of representation from those acquired by use. Dretske's acquired representations are historically dependent on the processes

that shape them and the use the representing device is put to. Sloman argues that the grammar of such acquired states are engineered to correspond in a way meaningful to the designer and user of a device.

Here, then, we have seen the agent's internal state as a pragmatics of functional control states. We now attempt to bring a theory of a more syntactic kind to bear on these states.

7 Example Automata Models

In the following sections, we explore the implications of automata models of agents and the results of applying the weak epistemology of modified SAT. We begin by quantising stimuli and responses, and then show the kinds of problems which arise, relating these to other work in this area.

The definition of a perceptable stimulus needs some formal rigour in order to be useful. An agent will usually sense the environment through a number of sensors over a number of modalities. Hence, a stimulus might be better defined as an ordered tuple. Note that in following this path, already the model becomes tied to a specific, discrete interpretation of the agent and its environment. Effectively, in introducing the notion of a *formal* stimulus we are discretising the agent and the environment.

Definition 7.1 *A formal stimulus is a percept defined over M sensors/modalities as the ordered tuple* $\iota = \langle x_1, x_2, \ldots, x_M \rangle$

Each input state $\iota \in I$ now corresponds to a tuple. In binding the stimulus to a specific set of sensed modalities, we encounter another problem. Each element of the ordered tuple can then be a quantifiable measure and in correspondance with some aspect of the environment. This fact was one of Brooks' concession on representation in AI, that sensor arrangements deliver signals that are in some part representations of the environment (Brooks and Stein, 1993). For example, a force sensor may be represented as an element with nominal values such as $0, 1$ referring to the presence or absence of an object incident with the sensor, or it may represent some magnitude of presence in which case it may assume real values $(0, 1)$.

In effect, then, a formal stimulus to an agent is a member of the Cartesian product of all possible states of each virtual sensor. Such a stimulus ι engages in a causal relationship with the internal state which then causes action to be taken. We now consider a geometric interpretation which lends itself to probabilistic interpretation, moving the model beyond a discrete, finite automata to a model more congruent with connectionism and adaptive behaviour. For generality, no specific connectionist realisation is given – instead, a probabilistic automaton is given. Mapping such an automaton to a connectionist implementation is a matter of selecting the correct training model for a connectionist system (including connectivity of nodes, activation and transfer functions for the nodes).

8 Stimuli State Space : A Geometric Interpretation

We can naturally pair the set-theoretic notion of the set of input states I with a geometric interpretation by assuming that each tuple $\iota \in I$ represents some point in a vector space $\Re^M$. For example, imagine an agent equipped with two sensors, one sensing light and the other moisture in its environment. Assume also that the sensors heavily quantise the environment so satisfactory light or moisture is conveyed to the agent as a tuple $\langle x_1, x_2 \rangle$ such that $x_1, x_2 \in \{0, 1\}$ where 1 is satisfactory, 0 requires action. From the phenomenal perspective of the agent, it either perceives light or moisture (or a combination of the two) or not. This results in the following set of possible input states (including one for "no observation"):

Input State	x_1	x_2
ι_0	null	null
ι_1	0	0
ι_2	0	1
ι_3	1	0
ι_4	1	1

If the set I as defined above were visualised in $\Re^2$ then we would arrive at a stimulus state space as shown in Figure 4. The motivation for this was originally McFarland's notion of internal state space (see (McFarland, 1996) for an introduction) and Arkin's independent treatement of Stimulus-Response models of reactive behaviour (see (Arkin, 1998) Chapter 3 on the stimulus plane). Naturally, this can be extended to stimulus state spaces in $\Re^M$ and this becomes valuable where computational-geometric interpretations of artificial neural network systems become a necessity.

Notice that the set I is suitably enumerated in Figure 4 and the table above. The finite set I can now be used as a basis for a systematic design task to engineer a specific reactive agent to solve a certain problem. The assumption is an unchanging environment, yielding a total of four different stimuli. It is such assumptions that make this a non-adaptive model of agents – adaptive behaviour seems to decay assumptions about strictly enumerable stimuli spaces and connectionism copes with this by adapting representations according to functional need.

9 Responses

Continuous real valued responses are most compatible with a robotic agent having a finite set of actuators (e.g. left and right motors) which can be used to control translational velocity. However, it is conceivable that a virtual agent might possess such virtual actuators where the strength (real value) of the response indicates the weight, gain or "confidence" which with the actuators is driven. These actuators are in fact discrete actions.

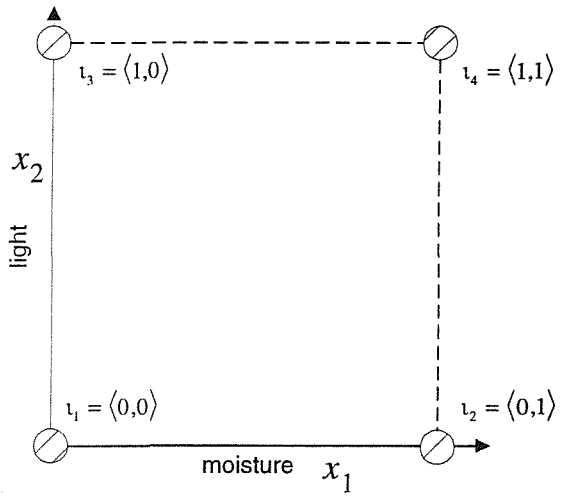

Figure 4: Example Stimuli State Space

For example, consider a simple agent responsible for maintaining a certain level of moisture and light. The stimulus state space is shown in Figure 4. A corresponding discrete enumerated set of atomic actions (primitives) might be $\langle light-off, water-off, no-op \rangle$, where no-op signifies do nothing.

Output State	Action primitive
o_1	$light-off$
o_2	$water-off$
o_3	$no-op$

This would effectively reflect the designers intentions and perception of how the agent should interact with the environment while implicitly defining any virtual actuators. This is a fairly appropriate paradigm for virtual agents, since the usual constraints on physical actuators do not translate to the software domain. In an idealised scenario, the engineer of a software agent could specify the reponse set and the corresponding stimulus set. The engineer would then design the reactive rules governing mappings from the stimuli to the responses (the behavioural specification).

9.1 Atomic Responses

In this section, we deal with a definition of a response and a geometric interpretation congruent with stimuli state spaces. Note that we define the responses as an enumerated set of discrete actions in the virtual agent paradigm; this effectively abstracts us away from a relationship

between a response, its strength and the resulting actuation. Also, we situate the definitions in automata terminology, thus :

Definition 9.1 *A* response *is an* action *taken by the agent in the environment.*

A response *set* can then be defined :

Definition 9.2 *A finite set of responses or actions* $O = \{o_1, o_2, \ldots, o_j\}$ *such that each o corresponds to one action executable in the environment in some instantaneous time t.*

Such a definition contains two important limiting assumption:

- that the time taken to effect any action is instantaneous. This is to say that the time taken is negligible in comparison to a sequence of actions taken over an extended period of action. This is a similar assumption to assuming that a physical robot is holonomic and is therefore able to turn on a singular point and there is no translational latency due to motor inefficiency. Such assumptions are obviously incorrect for physical agents, but these limitations are not as considerable for virtual agents, except in the case where the result of an operant action in the environment does not produce immediate feedback indicating the effect of the action.
- that only one action can be taken – congruent with the Mealy model output function F which can be in only one state

9.2 The Action Selection Problem

In the previous section, the notion of an atomic set of responses was introduced and the assumption and ramifications buried beneath such a primitive treatment were outlined. The implications for a theory of reactive agency are that firstly, an agent can only choose one action and hence, arbitration of the available responses must be implicit in the output function. Secondly, the automata is deterministic – given a stimulus, it always chooses the same single response.

This raises a difficulty when an internal state arises demanding two actions. As an example, consider the case for the light/moisture control agent: if the stimulus is ι_1 then simultaneously we can assert the agent carries information pertaining to the fact that "it is both too moist and too light".

This calls for two atomic responses : "water-off" and "light-off". Now, the theory must account for the following options:

1. *Sequence the actions*: "water-off";"light-off" or vice versa
2. *Define an extra atomic response corresponding to a joint action*: "water-off-light-off"
3. *Arbitrate between actions*, hoping that effecting one will cause the agent's internal state (by virtue of the perception of the new environmental state) to reflect a different state

which can then be treated using the another atomic action – hence we pick between "water-off" and "light-off", effecting (say) "water-off" hoping that at some time $t+1$ the internal state will correspond to simply "too much light" and no longer "it is both too moist and too light".

The automata theory now converges with features of a deliberative planner, to schedule the actions. This may be taken as an indication of the boundary of situated, purely reactive behaviour, and we could therefore exclude it from the scope of the research. The argument might progress by assuming that the limitations of an automaton are sufficient and necessary definitions to prevent the unification of deliberative and reactive agents. However, it is felt that some discussion of a resolution is necessary.

The first option (sequencing actions) is the least desirable : it implies that we effect both actions without "resampling" the environment between each action. In this case, sequencing of behaviours reduces to the second option.

The second option contains the implication that all motor systems (virtual actuators) of the agent are independent and can be used in parallel to effect an action. This is clearly not possible in physical agents – an agent wishing to move forward would invoke a certain set of signals causing two motors to produce forward motion, which mutually excludes a complementary set of signals causing reverse locomotion being activated simultaneously. If forward and backwards locomotion are discrete actions, then they must be mutually exclusive. The behavioural final common path of McFarland and Sibly (1975) states that at any moment in time a single action is effected. This has impliciations for the granularity of actions for an agent so adopting McFarland and Sibly's temporal definition also restricts what we might reasonably define as an action given a particular environment.

The third option is to arbitrate, but then in a deteministic automaton, the same action will always be picked first. This may be unsatisfactory if the relationship between environmental conditions inducing the internal state of the agent are not independant, for example, if some kind of heating action was available to the agent, then moisture and light might be dependant on heat. This would suggest some priority to the action affecting the temperature of the environment to see if this corrects the problem before effecting "water-off" or "light-off".

The solution to the third option's dichotomy is to introduce some arbitrating factor and preferably, an adaptive one. The choice of effected action depends on the weight assigned to the action (a strength or confidence) and arbitration then takes place independently of output function F. This requires a more sophisticated notion of a response space and the relaxation of determinism in the output function.

In effect, this is a formal realisation of the action selection (AS) problem (Tyrell, 1993) under an automata based theory of reactive agency. Given a choice of available actions, which would best satisfy the goals of the agent ? The action selection mechanism (ASM) must resolve this problem. As we have described it here, the agent will benefit equally from either action.

However, as indicated with the dependancy of environmental factors above, this could be an incorrect assumption.

One of Tyrell's design criteria for any action selection mechanism is that it does not impose a system level winner take all (WTA) function over systems of the behavioural or task acheiving modules. The selection of an action from a set of mutually exclusive and exhaustive actions depends on a function of each actions "viability" represented as a real-valued number. Formally, for an ordered set of actions corresponding to output states in the automata model have a corresponding set of real-valued instantaneous viabilities $\langle v_1, v_2, \ldots, v_j \rangle$. The chosen action is governed by :

$$action = \arg \left[\max_i v_i \right] \tag{9}$$

This principle, pick the action with the largest viability and ignore the rest, can be realised by the well documented "softmax" or normalised exponential function found in neural network literature (Bridle, 1990) and (Kohonen, 1997) Chapter 4. The function is typically :

$$action = \frac{\exp(\lambda v_i)}{\sum_{v_{i'}} \exp(\lambda v_{i'})} \tag{10}$$

Where a crisp winner-take-all function is acheived as $\lambda \rightarrow \infty$. Tyrell (1993) Chapter 10 suggests this should not be present at the system level, that is, because of the corresponding loss of information a WTA function conveys to subsequent or dependent processing modules. Here, it is argued that it must be present at the behavioural level, or else responsibility is shifted to something which ultimately is beyond the control of the agent.

Tyrell's observation and it's realisation in equation 10 suggest that atomic behaviours are reasonable (as governed by the behavioural final common path) but with a modification that enables choices among viable alternatives. This modification is essentially the admittance of non-determinism and real-valued weightings for the responses or actions.

A similar definition to equation 10 is used in (Humphrys, 1997) as a basis for one type of reinforcement learning of action selection "viabilities". Humphrys[1] dealt explicitly with "*W*-learning", where the agent learns the response viability (or weight) and the values of taking actions given environment state (loosely, a stimulus in our terminology) using Q-learning (a type of reinforcement learning). In effect, an agent should learn abitration associations as part of the sensori-motor loop which engages with the environment.

One final question that remains open is whether or not the AS problem is *just* an artefact of the modelling approach. A recent paper (Seth, 1998) adopts the view that action selection is in fact implicit in a continuous and coherent stream of perception and action. A similar

[1] Humphrys' thesis was concerned with an agent as a member of a decentralised model of an abstract "mind" in the sense Minsky (1986) used the term agent. In this report, an agent is some autonomous actor in its environment – its internal mechanisms are composed of what Minsky and Humphrys term agents.

notion is embodied in (Braitenberg, 1984) where parsimonious collections of direct sensori-motor connections enable the agent's behaviour. The answer to this question depends on what is being sought. An ethological study of animal behaviour aims to find a model with explantory power for observed, codified behaviour such as (Tinbergen, 1951). The act of interpreting the behaviour necessarily imposes and divides the continuous locomotive activity (the observable, distal behaviour) into discrete actions. So, we might argue that the resulting model of action selection is only necessary because of the constraints of the behavioural final common path, where the animal or agent selects only *one* of the available actions after internal arbitration.

Effectively, in the ethological context, it is possible that the mode of analysis instantiates a discrete action selection problem in the explantory model of the animal or agent's behaviour. Likewise, the manifestation of the action selection problem has been shown for the situated automata approach.

The above answer is one of biological fidelity in the resulting explanatory model. It *could be* that models resulting from a process as described above are in fact artificially instantiating an action selection problem precisely because of the analysis and subsequent modelling approach used. However, this is not an issue when considering the engineering of artificial agents, where the environment or the actions *are necessarily* discrete. In this sense, the AS problem is a real concern in the context of engineering softbots. Such agents will interact with a discrete world and will effect discrete actions and as such, the *actual level* of the behavioural final common path is arbitrary but tied to the agent's environment and design goals.

In addition, the agent model developed in this thesis has its origins in neural network models and we have defined the agent in terms of stimuli and responses. We have then discretised them and this has introduced the action selection problem. The AS problem manifests itself because of the modelling process or analytic framework brought to bear on the problem. Discretising typical *SR* models and taking the responses to be actions introduces the problem, as does an analysis grounded in ethological theories.

Congruent with Seth's proposal, AS will be treated here as being implemented by a mutual continuous routine emerging from the dynamics of the environment and the agent's goal directed mechanism.

9.3 Atomic Responses with Real-valued Weighting

If some notion of response strength or viability is introduced it can be interpreted as a weight or confidence in the response or action. This can later be used in response choice arbitration, to decide on a singular action to be taken cf. the suppression and subsumption in (Brooks, 1986) and behavioural fusion (Arkin, 1998) pp. 111.

This leads to the response *space* interpretation, which for a given set of available actions O associates a real value indicating a measure of the strength of the response. Note the key distinction between this and the definition of a stimuli space : each of the enumerated set

of responses O are not tuples containing the strength or presence of a particular actuator (as they were for individual sensors). Instead, the actions in O are primitives such as "light-on", "forward" or "kick".

Definition 9.3 *A* response set *is a set of pairs* (o, ρ) *where* $o \in O$ *is some action and* $\rho = [0, n \in \Re]$ *is a measure of confidence or strength of the response where appropriate.*

Further, a response space is functionally dependant on internal state. For each internal state $s(t)$ we can expect a corresponding response space (for example, a set of realised activations on actuator/output nodes in a neural network). This reflects the dynamics of how responses (and therefore behaviour) is different depending on the agent's perception of the environment. It is argued here that the limitations of automata models are when such non-determinism is present in both the environment and the agent's means of behaving in the environment. The demands of such situations strongly suggest a need for adaptation in reactive systems. This is because while an automata model *describes* a reactive system with some analytical precision and captures the essential characteristics of reactive behaviour, it cannot fully account for the *situatedness* of the agent and what this means for its behaviour.

10 An Example of a Reactive Agent using Revised SAT

Taking the revised SAT model developed thus far, we specify fully the agent from the previous sections. The Mealy machine is divided in two and represented diagrammatically as Figure 5 for the function F (the perceptual machinery of the agent) and Figure 6 for the function G (the virtual actuators of the agent). The initial states are shown as double circles and events shown alongside the arcs showing allowable state transitions. For convenience, the ascriptions are simple propositions and we have defined them as follows : ϕ_0 is a null proposition, $\phi_1 \stackrel{\text{def}}{=}$ "too much moisture in the environment", $\phi_2 \stackrel{\text{def}}{=}$ "too much light in the environment", $\phi_3 \stackrel{\text{def}}{=}$ "the environment conditions are satisfactory". The final column shows which of these ascriptions are true for any internal state.

The next internal state function is trivial : irrespective of current state, the next state wholly determined by the current input ι. Table 1 shows the state transitions for Figure 5.

We now consider the more difficult case of the output function G. We have already stated that the output function will be intrinsically non-deterministic. Figure 6 shows the state transition diagram for the output function. Each output state (node) $o \in O$ corresponds to an action or response, and the events on the arcs of the graph are the internal state contingencies. Note that more than one internal state can cause an output state to be reached (shown s_i/s_j on the diagram) and probabilities for transition when there is non-determinism is shown as (p) denoting the probability of the output state being reached. The case where the agent is in state s_2 presents a dichotomy. Since s_2 corresponds to the joint observation that there is "too much

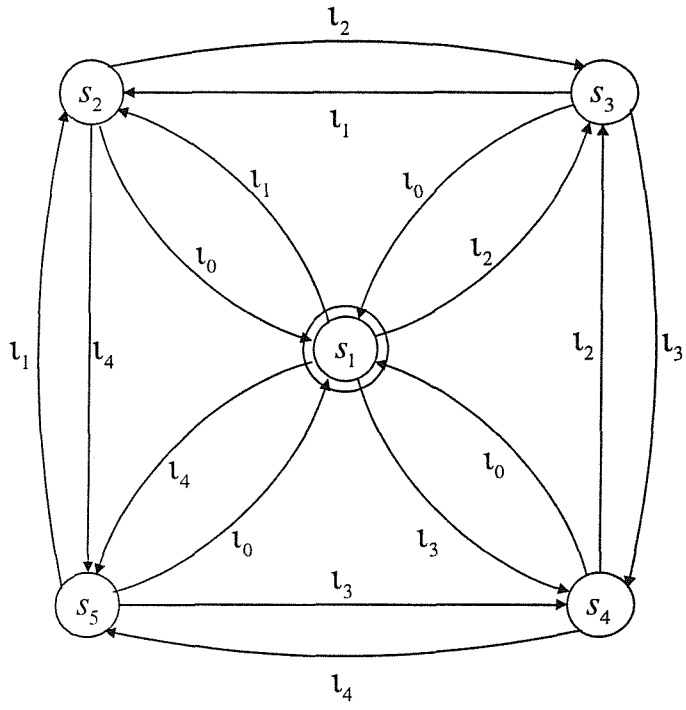

Figure 5: STD showing agent perception (formally, the function F) – transitions from a state to itself are ommitted for clarity

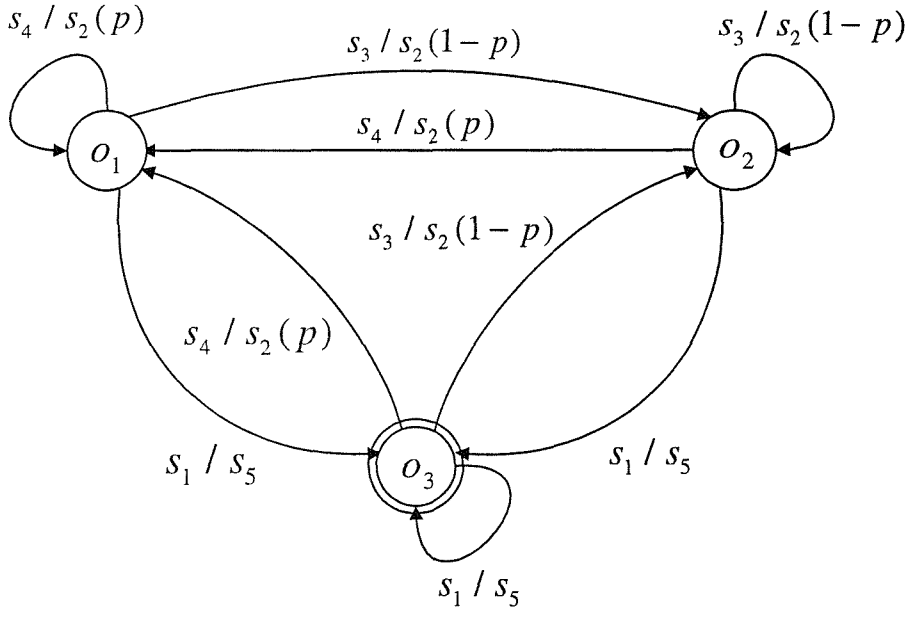

Figure 6: STD showing agent actions (formally, the function G)

Input State	x_1	x_2	Internal State	Ascribed Information ϕ
ι_0	null	null	s_1	"no observation" (ϕ_0)
ι_1	0	0	s_2	"too much moisture" and "too much light" ($\phi_1 \wedge \phi_2$)
ι_2	0	1	s_3	"too much moisture" and "light okay" (ϕ_1)
ι_3	1	0	s_4	"moisture okay" and "too much light" (ϕ_2)
ι_4	1	1	s_5	"moisture and light satisfactory" (ϕ_3)

Table 1: Specification for F

light" and "too much moisture" ($\phi_1 \wedge \phi_2 = \mathit{true}$ we have the action selection problem outlined in section 9.2).

The action selection dichotomy here is whether to effect "light-off" or "water-off", given that both are suggested by the current internal state of the agent. The strategy presented in section 9.3 was to weight the responses. A complementary approach, most compatible with probabilisitic automata and Markov processes, is to define some normative probabilities to the output component of the automata entering a state corresponding to the two actions. Take the transitions from state o_3 in Figure 6. The events that realise state o_1 are s_4, s_2 and for state o_2 they are s_3 and s_2. If (while in state o_3) the agent's internal state is s_3 or s_4 then a deterministic transition is defined. However, should the agent arrive at internal state s_2 then with probability p the agent chooses o_1 and $(1-p)$ chooses o_2.

The justification for assigning p and $1-p$ respectively can come from two sources :

1. the agent designer
2. the agent's own experience due to participation and situation in the environment

With either option, the agent is now a stochastic entity and no longer wholly determined by the internal state transitions. As noted earlier, it is then a matter of defining a suitable means of adapting the probabilities of certain actions when the agent finds itself in certain internal states. One method would be to implement a probabilistic function over relative strengths of nodes in a connectionist or more generally, any activation spreading system. The structural components that then most significantly influence the chosen output are the weights which cause activations on nodes.

11 Conclusion

This technical report attempted to introduce an automata-theoretic notion of reactive agency. Philosophically, this is appropriate because of the pattern recognition / driven nature of reactive behaviour and its specification in terms of S-R models. While the automata model has been

considered in the agent literature e.g. (Wooldridge, 1999) it has not been specialised to cope with reactive or adaptive behaviour explicitly.

The apparently disparate practices of traditional software agents and reactive automata-like systems was begun by (Rosenschein and Kaelbling, 1995) in SAT, but was extended here to cope with potentially adaptive behaviour. The modification was acheived by relaxing the strong constraint on correlational information, and introducing an analogy with multiple realisability and heterogeneous disjunctions. Effectively, the mathematically tractable and elegant framework introduced in SAT was broken to favour realisable adaptive agents. The notion of representation in reactive systems was approached similarly.

The extensional characteristics of a general reactive agent were proposed with reference to Arkin's behaviour-functional formulation. Stimuli and response state space interpretations were introduced (to move away from the assumption of finite, enumerated environments) and their utility was demonstrated when considering a simple exemplar reactive agent which faced an action dichotomy when a certain internal state occurred from its perception of the environment.

In conclusion, SAT commits to a strong epistemological constraint which may not hold when internal state is adapted. Correlational definitions work only when they can be heterogenous disjunctions of postulated internal states. SAT provides a means of specifiying the agent in abstract, and then deriving a working machine to implement it. Behaviour-functional definitions are a similar model for SR inspired systems such as those given in (Arkin, 1998). To realise an adaptive agent defined in a behaviour-functional way requires that simple automaton models be abandonded in favour of models which can acquire action probabilities. A simple means of achieving this is to enable the agent to acquire these probabilities by experience in the environment. Future work will need to explore kinds of mechanisms which are suitable. Reinforcement learning methods (Kaebling, Littman and Moore, 1996) have been shown to be plausible for this purpose, but it is unclear how they will cope with changes in the environment.

References

Agre, P. (1995). Computational research on interaction and agency. *Artificial Intelligence 72*, 1–52.

Agre, P. (1997). *Computation and Human Experience*. Cambridge University Press.

Agre, P. and I. Horswill (1997). Lifeworld analysis. *Journal of Artificial Intelligence Research 6*, 111–145.

Arbib, M. A. (1964). *Brains, machines and Mathematics*. McGraw-Hill, Inc.

Arbib, M. A. (1989). *The Mataphorical Brain 2: Neural Networks and Beyond*. John Wiley and Son.

Arbib, M. A. (1995). Levels and styles of analysis. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks*, pp. 11–17. The MIT Press, Cambridge, MA.

Arkin, R. C. (1998). *Behaviour-Based Robotics*. The MIT Press.

Block, N. (1980). Introduction: What is functionalism? In N. Block (Ed.), *Readings in philosophy of psychology*, pp. 171–184. Harvard University Press.

Block, N. (1995). The mind as the software of the brain. In D. Osherson, L. Gleitman, S. Kosslyn, E. Smith, and S. Sternberg (Eds.), *An Invitation to Cognitive Science*. The MIT Press.

Block, N. (1997). Anti-reductionism slaps back. *Philosophical Perspectives 11*, 107–132.

Braitenberg, V. (1984). *Vehicles:Experiments in Synthetic Psychology*. The MIT Press, Cambridge, MA.

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs with relationships to statistical pattern recognition. In F. Fogelman Soulié and J. Herault (Eds.), *Neurocomputing*. Springer Verlag, Berlin.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation 2*(1), 14–23.

Brooks, R. A. and L. A. Stein (1993). Building brains for bodies – A.I. Memo No. 1439. Technical report, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Burks, A. W. (Ed.) (1966). *Theory of Self-reproducing Cellular Automata.* University of Illinois Press.

Catania, A. C. (1992). *Learning.* Prentice-Hall International Editions, Inc.

Chalmers, D. J. (1996). *The Conscious Mind.* Oxford University Press.

de Lioncourt, S. W. and M. Luck (1999). Motivating intelligent agents for virtual environments. In *Proceedings of the Second Workshop on Intelligent Virtual Agents*, pp. 79–99. ISSN 1467-2154.

Dennett, D. C. (1978). *Brainstorms.* Bradford Books.

Dennett, D. C. (1987). *The Intentional Stance.* The MIT Press.

D'Inverno, M., M. Fisher, A. Lomuscio, M. Luck, M. De Rijke, M. Ryan, and M. Wooldridge (1997). Formalisms for multi-agent systems. *Knowledge Engineering Review 12*(3), 315–321.

Dretske, F. (1985). Machines and the mental. *Proceedings and Addresses of the APA 59*, 23–33.

Dretske, F. (1995). *Naturalizing the Mind.* The MIT Press.

Etzioni, O. (1993). Intelligence without Robots (A reply to Brooks). *Artificial Intelligence Magazine 14*(4), 7–13.

Fodor, J. (1998). *Concepts.* Oxford University Press.

Heil, J. (1998). *Philosophy of Mind : A Contemporary Introduction.* London: Routledge.

Humphrys, M. (1997). *Action Selection methods using Reinforcement Learning.* Ph.D. thesis, University of Cambridge, Computer Laboratory.

Kaebling, L. P., M. L. Littman, and A. W. Moore (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research 4*, 237–285.

Kim, J. (1998). *Philosophy of Mind.* Westview Press, Oxford.

Kohonen, T. (1997). *Self Organizing Maps* (2 ed.). Springer-Verlag, Berlin, Heidelberg, New York.

MacLennan, B. J. and G. M. Burghardt (1994). Synthetic ethology and the evolution of cooperative communication. *Adaptive Behaviour 2*(2), 161–188.

McCarthy, J. (1978). Ascribing mental qualities to machines. Technical report, Stanford University AI Laboratory.

McFarland, D. and R. Sibly (1975). The behavioural final common path. *Philosophical Transactions of the Royal Society 270*, 265–293.

McFarland, D. J. (1996). Animals as cost-based robots. In M. A. Boden (Ed.), *Philosophy of Artificial Life*, pp. 179–205. Oxford University Press.

Minsky, M. (1986). *The Society of Mind.* The MIT Press, Cambridge, MA.

Neisser, U. (1976). *Cognition and Reality: Principles and implications of cognitive psychology.* W.H. Freeman.

Rao, A. S. and M. P. Georgeff (1995). BDI-agents: From theory to practice. In *Proceedings of 1st International Conference on Multi-Agent Systems (ICMAS-95)*, pp. 312–319.

Rosenschein, S. J. and L. P. Kaelbling (1995). A situated view of representation and control. *Artificial Intelligence 73*(1-2), 149–173.

Roy, J.-M., J. Petitot, B. Pachoud, and F. J. Varela (1999). Beyond the gap: An introduction to naturalising phenomenology. In *Naturalising Phenomenology*, pp. 1–89. Standford University Press.

Sampson, J. R. (1976). *Adaptive Information Processing.* Springer-Verlag.

Seth, A. K. (1998). Evolving action selection and selective attention without actions, attention or selection. In *Proceedings of the 5th International Conference of the Society for Adaptive Behaviour (SAB98)*, pp. 139–147.

Sharkey, N. E. and J. N. H. Heemskerk (1997). The neural mind and the robot. In A. Browne (Ed.), *Neural Network Perspectives on Cognition and Adaptive Robotics*, pp. 169–194. Institute of Physics Publishing, Bristol.

Sloman, A. (1996). Towards a general theory of representation. In D. Peterson (Ed.), *Forms or Representation*, pp. 118–140. Intellect Books.

Smart, J. J. C. (1959). Sensations and brain processes. *Philosophical Review 68*, 141–156.

Tinbergen, N. (1951). *The Study of Instinct.* Clarendon Press, Oxford.

Tyrell, T. (1993). *Computational Mechanisms for Action Selection.* Ph.D. thesis, University of Edinburgh.

van Gelder, T. (1990). Compositionality: A connectionist variation on a classical theme. *Cognitive Science 14*, 355–384.

Wilson, S. W. (1991). The animat path to AI. In *From Animals to Animats: Proceedings of the First International Conference on the Simulation of Adaptive Behavior*. MIT Press/Bradford Books.

Wooldridge, M. (1999). Intelligent agents. In G. Weiss (Ed.), *Multiagent Systems*. MIT Press.

University of Southampton

Intelligence, Agents and Multimedia Research Group, Department of Electronics and Computer Science

Technical Report No. ESTCR-IAM005

Experiments with Connectionist Models of Reinforcement Learning for Software Agents

Dan W. Joyce

April 2001

ABSTRACT

This technical report reviews some literature on reinforcement learning and neural networks. It juxtaposes the tradition of machine learning with neural modelling techniques which reproduce similar phenomena (namely, reinforcement and operant conditioning). The report then goes on to describe an implementation of an agent which uses a combination of discrete-time Gaussian adaptive resonance theory and an Hebbian associative search network to implement pattern classification. A further experiment is described which attempts to look beyond the limitations of the classification agent as a model of situated or embodied agency. The classification agent lacks some of the requisites to study the continuous interactions between agents and their environments (albeit in a simulation). The experiment and a brief outline of two agent connectionist architectures (both based on connectionist realisations of reinforcement learning) are provided, as are results from the simulations.

Contents

1 Introduction

This technical report contains a survey of reinforcement learning methods tested and applied in an two applications; one connectionist-based agent for classifications in a multimedia environment and another for exploring connectionist implementations of a simple non-stationary environment akin to a one-armed bandit. Much of this document supports and details the experimental work described in the draft document (Joyce, 2001b).

The report begins by describing some of the history and literature of reinforcement learning. It then describes the common frameworks of reinforcement learning (as defined in the machine learning community) and then describes alternatives originating in neural network models of psychological phenomena. The network models described were all used, in some way, for the implementations of MAVIS2 (described later) and the simulations of embodied activity in a simple virtual agent.

The report concludes by describing a simulation, briefly describing the kinds of network architecture used for agents in the environment and then reproduces the experimental data with commentary on the results for the two agent implementations.

2 Literature Review: Reinforcement Learning and Conditioning Models

From initial explorations in the MAVIS2 environment, it was decided to undertake a literature review of the relationship between automata models, agents and the use of reinforcement learning techniques. Much of the work conducted by Grossberg and colleagues (Grossberg, 1972a; Grossberg, 1972b; Grossberg and Levine, 1987) on neural network models of reinforcement phenomena (i.e. those observed by experimental psychologists rather than the machine learning definition) *implement* systems with similar behavior to the machine learning community's definitions of reinforcement learning. In terms of observable agent behaviour, the two approaches converge despite different implementation methods and lineage.

After initial work on using drive nodes as reinforcement sources internal to the agent (Chang and Gaudiano, 1998) – see also, section 3.3 and section 4 of this report – attention was turned to the use of machine learning reinforcement techniques and their realisation utilising connectionist models.

2.1 Stochastic and Learning Automata

The strong link between agents and automata enables us to connect easily with the *learning automata* models explored in the mathematical psychology of learning (Hilgard and Bower, 1966). A comprehensive survey of early work (Narendra and Thathachar, 1974) reveals that the assumptions are similar to the model presented here. The environment is regarded as a source of teaching signals and the agent as a stochastic automaton: that is, an automaton whose output function selects actions according to probability assignments.

2.1.1 Definitions

A stochastic learning automaton (SLA) is where choice of actions are initially random (each has uniform probability of being selected) but as experience accrues, the agent modifies the probability of selecting certain actions based on the environment's feedback on their consequences cf. (Dennett, 1996) where the SLA would be a Skinnerian agent with limited perception. Hence, a learning automaton is fully described by the environment's *penalty set* (denoting consequences), the agent's *choice probability vector* (indicating the probability of an action) and a reinforcement scheme (Narendra and Thathachar, 1974) pp. 326.

The SLA model can be formalised by simplifying the Mealy automaton model adopted in (Joyce, 2001a) as follows:

- an input set $I = \{0, 1\}$ indicating a reward (implying the agent only ever perceives the environment as a state which represents a reward (0) or punishment (1))
- a set of internal states $S = \{s_k \mid k \in \mathbf{N}\}$

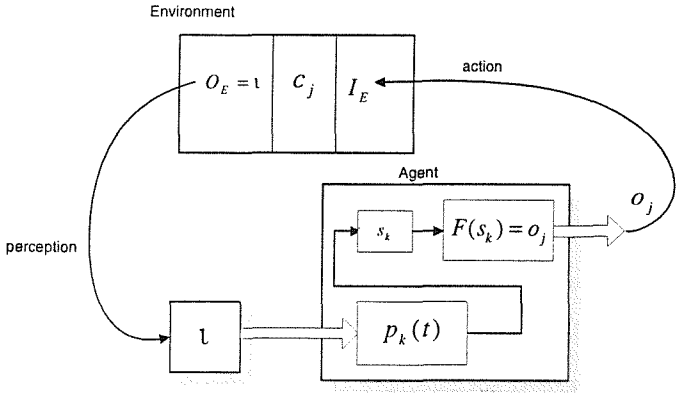

Figure 1: Learning Stochastic Automata

- an action (or output) set $O = \{o_j \mid j \in \mathbf{N}\}$ where $j \leq k$
- a state probability vector $P = \langle p_1(t), p_2(t), \ldots, p_k(t) \rangle$ where $p_k(t)$ is probability of the agent's internal state being s_k at time t
- internal states are identified with actions (e.g. $k = j$) such that *only* an output function $F : S \rightarrow O$ is required.
- an algorithm A (reinforcement or updating scheme) which generates $p(n+1)$ from $p(n)$

In this context, adaptive behaviour is the modification of the state probability vector under the interpretation (cf. behaviour generation) of F and the update algorithm A. This is shown in Figure 1.

The agent-environment interface is the update algorithm A's relationship with the environment. The environment is specified as an automaton which takes as input agent actions and responds with penalties:

- an environment input set $I_E = O = \{o_j \mid j \in \mathbf{N}\}$
- an environment output set $O_E = I = \{0, 1\}$
- the environment's penalty probability vector $C = \langle c_1, c_2, \ldots, c_j \rangle$ where c_j denotes the probability of action j generating a penalty signal (i.e. a member of O_E indicating punishment)

The environment's state signal is peculiar in that it only reports a neutral reward or noxious penalty cf. reinforcement learning where an environment state is reported along with a reward obtained from the last action taken. Also, if the environment's penalty probability vector $C(t) = C(t+1)$ then the environment is classed as stationary. That is to say, the probability of an action yielding some reward or penalty is time invariant.

2.1.2 Adaptive Behaviour in a SLA

Initially, each internal state (and therefore action) has equal probability $\Pr_k(t=0) = 1/k$. At time t, the agent will possess the state probability vector P which governs its next choice of internal state, and action. For a given P at time t the mean penalty can be defined as :

$$M(t) = \sum_{i=1}^{j} p_i(t) c_i \tag{1}$$

assuming that the environments output (and the agent's input) are binary penalty/reward schemes. Naturally, at $t = 0$ this would correspond to :

$$M(0) = \frac{1}{r} \sum_{i=1}^{j} c_i \tag{2}$$

A SLA adapts (or learns) if, asymptotically, the average penalty $M(t) < M(0)$ as $t \to \infty$. (Narendra and Thathachar, 1974) also give constraints for optimality of the learning process. Essentially, the SLA should minimise the penalty just as for minimisation of the error function in supervised neural network learning algorithms, and the definitions of optimality in reinforcement learning. Note that goal directed behaviour is formulated as some optimality criteria, and the actual implementation of a behaviour mechanism is then inscribed with this assumption.

The reinforcement scheme updates the probabilities of a certain action being effected. A reinforcement scheme (or algorithm) A is defined as :

$$P(t+1) = T[P(t), o(t), \iota(t)] \tag{3}$$

Where $P(t)$, $o(t)$ and $\iota(t)$ are the action probabilties (state probability vector), action taken and input to the agent respectively for some time t. T is an operator that produces $P(t+1)$ from the agent's current state/action and the penalty received. Reinforcement schemes are either linear functions of $P(t)$ or non-linear. The issue of reinforcement functions and learning rules will be dealt with later. In either the linear or non-linear cases, the principle is that the probability of an action increases if it resulted in reward and decreased otherwise cf. the Hebbian and anti-Hebbian postulate of learning (Hebb, 1949) and (Brown and Chatterji, 1995).

2.2 Reinforcement Learning

At this stage, it is more than apparent that the perspectives surveyed here are related to the general machine learning approach of re-inforcement learning (RL). In this section, we attempt to show that the models presented here (specifically, automata and behaviour-functional methods)

are in fact members of a set of techniques which are all reasonably subsumed by the label reinforcement learning. The aim here is to show how the *formulation* of the reinforcement learning problem is applicable.

A survey of reinforcement learning can be found in (Kaebling, Littman and Moore, 1996) and the introductory principles in (Ballard, 1997) and (Sutton and Barto, 1998). The field is far too broad to be fully considered here, so we will limit the scope of the discussion to those techniques under the RL banner which are most similar to the connectionist approach (associative reinforcement learning) and the formalisation of the agent in RL. That is, agents that use a control policy which is model free and agents must aquire experience of the world by sampling it directly. This is related to the RL methods which fall under incremental Monte Carlo techniques from mathematical estimation theory.

Most significantly, the amalgam of associative connectionist techniques with the principle of reinforcement learning provides a method of i) breaking the supervised teacher-agent relationship which exists in the purely supervised learning methods common to feedforward connectionist models ii) the requirement to enumerate the entire state-action space in reinforcement learning.

2.2.1 Definitions

As for SLA, the agent-environment interface is modelled, but the environment state *and* a reinforcement signal are provided. In RL literature, it is convention to talk of the environment state as being something that might be considered a stimulus in this thesis. However, the direct relationship between the environment state and internal state has been explored in (Joyce, 2001a), where the output of the perceptual machinery provides the agent with a "cue" for responding. At the level of machine learning mechanisms, this is analogous to environment-state generalisation in RL. For this reason, we will not use the same notation for environment state and stimulus here.

A reinforcement learner is specified as:

- a finite set of *environment states* $X = \{x_i \mid i \in \mathbf{N}\}$ representing states of the environment as perceived by the agent cf. the output of the perception mechanism
- a finite set of *output states* $O = \{o_j \mid j \in \mathbf{N}\}$ representing actions
- a *policy* which is a function relating perceived state to probabilities of actions $\pi_t(x, o) = \Pr[o(t) = o \mid x(t) = x]$
- a scalar *reward* signal $r_t \in \Re$

So, at some time t an agent will find itself perceiving the environment as state x and will take an action o with probability $\pi_t(x, o)$ which will result in a reward or punishment r at time $t+1$.

One important contribution of the RL formulation is the introduction of the discrete Markov property for the environment state – that is, a stochastic process whose past has no influence on the future if its present is specified. One immediately obvious limitation of reactive agents is that *all* information upon which the agent chooses its response or action must be available in the perceived stimuli (Jennings, Sycara and Wooldridge, 1998), although a connectionist-based theory of agency appears to defeat this argument. A formalisation of this property can be given in terms of a discrete Markov process over the (input) states given the agent's last action and reward. The Markov property for a discrete random variable **X** is :

$$\Pr[\mathbf{X_{t+1}} \mid \mathbf{X_t}, \ldots, \mathbf{X_1}] = \Pr[\mathbf{X_{t+1}} \mid \mathbf{X_t}] \tag{4}$$

So, in terms of the input to the agent (the reward r and stimulus x) and the last action taken o_t :

$$\Pr\left[x_{t+1} = x', r_{t+1} = r' \mid x_t, o_t\right] \tag{5}$$

This states that the probability of a certain input $x' \in X$ and reward $r' \in \Re$ at $t+1$ given that the environment was in state (strictly, in the phenemonological theory, this means the agent perceived the environment and arrived at some coded internal state) x_t and action o_t was taken. So, the state transition dynamics depend *only* on the previous state. This promotes the principle that the agent must receive sufficient information in its sensing of the environment for it to be able to effectively learn and act. Also, we can now state that if the probabilities of state transitions does not change over time, then the environment is stationary. Note that the environment is non-deterministic, in that transitions are stochastic, but those probabilities of transitions do not change over time. Most RL techniques take as a basic assumption that the environment is stationary (Kaebling, Littman and Moore, 1996).

2.3 Adaptive Behaviour in RL

Broadly speaking, the categories of RL techniques are :

1. immediate reward techniques : direct interaction with the environment at time t resulting in a contingent reward r at $t+1$

2. delayed reward techniques : direct interaction with the environment at time t but rewards delayed until some later time $t+n$ – effectively, this is a special case of immediate reward, except that the rewards are temporally sparse.

3. model-based learning : possesion of a model of the state transitions probabilities and reinforcement function

4. model-free learning : no possesion of a model of the state transitions probabilities and reinforcement function

In any case, the aim of a RL algorithm is to find an optimal policy π governing the mapping of states to actions for every possible state. Following (Sutton and Barto, 1998) we measure the reward obtained over a period of time as the *return* gained. For example, assume that a sequence of actions under some policy π generates the following reward sequence:

$$\langle r_1 = 1, r_2 = 1, r_3 = 0, r_4 = 0, r_5 = 1, r_6 = 0 \rangle \tag{6}$$

the reward gained over $t = 1 \ldots 6 = 3$. A policy should maximise this return. Assume that the above sequence is the expected sequence generated if we follow policy π from time $t = 0$. π tells us which action to take given the currently perceived input ι and we receive the reward r_1 after the action is taken, then follow policy π thereafter. Then, this is one of many possible reward sequences. Rewards later in the sequence (e.g. at $t > 1$) are considered to be "long term" rewards, but not as immediately important as ensuring short term reward. So, future rewards are *discounted* in the calculation of the overall return (Kaebling, Littman and Moore, 1996) to emphasise that policies should favour immediate reward.

This is achieved by geometrically discounting the rewards in the sequence by a factor γ^k, where $0 < \gamma < 1$ and k indexes the rewards, assuming $t + k$ is the kth reward in a sequence. Note that this notation, due to (Sutton and Barto, 1998) is to allow for both episodes of activity (which terminate) and continuous interactions. Hence, the duration over which the return is being measured is T so that an episode can be defined relative to t and the index k runs from 0 to T. So, the actual return under the discounting system as given by (Sutton and Barto, 1998) is:

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \tag{7}$$

Note that this assumes we are measuring return over a known sequence of actions following a policy π. In applications, this value can rarely be calculated *a priori* and only in limited theoretical cases can this value be used directly to find the best policy in a (simple) environment.

Typically, a *state-value* function is also defined, which measures how "useful" a state is to the agent attempting to perform in the environment. This is defined by resorting to the expected value of the return R_t for a specific policy and defining the *state-value function for policy* π:

$$V^{\pi}(x) = E_{\pi}[R_t \mid x_t = x] = E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, x_t = x \right\} \tag{8}$$

where E_{π} the expectation operator. Similarly, we need to define a function which yields the value of taking a certain action if the environment is in a certain state (e.g. the perceived

stimulus for the agent). This is the *action-value* function and is defined as follows :

$$Q^{\pi}(x,o) = E_{\pi}[R_t \mid x_t = x, o_t = o] = E_{\pi}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| x_t = x, o_t = o \right\} \tag{9}$$

Hence the value function for a policy is related to the Q value by (Ballard, 1997):

$$V^{\pi}(x) = \max_{o}[Q^{\pi}(x,o)] \tag{10}$$

because the value of a state is determined by the most likely action the agent will take in that state under policy π. Therefore, the central tennet of reinforcement learning is the maximisation of V by manipulating the policy π. Although analytically elegant, this informs us of little about its application in agent learning. It should also be noted that the expected value of the rewards in equations 8 and 9 is fundamentally dependant on a stationary environment.

2.4 Learning under the RL Formulation

The question of *how* to acquire a policy π is still open. One method is to use theoretical results to derive an optimal policy such that :

$$V^{\pi*}(x) = \max_{o}[Q(x,o)] \tag{11}$$

where $\pi*$ is the optimal policy that guarantees that the *best* action to take for any state is that with the highest action probability. However, such derivations (see (Sutton and Barto, 1998) for examples) only hold in either simple theoretical cases or when dynamic programming can be applied. Alternatively, exhaustive sampling of the state-action space by selecting a state, which is then used to exhaustively search for the best action given this state. Neither of these approaches is particularly useful in agent applications, since they imply off-line training (e.g. an agent which is not structurally coupled to its environment).

An alternative is a derivative of Monte-Carlo based estimation and policy iteration. In this approach, the agent samples the space as it encounters it. The danger is that some high reward states may not be visited because the agent begins to visit regions of the state space which give positive rewards immediately. However, the agent gradually refines $\pi : X \rightarrow O$ in an attempt to maximise V.

This is described by the algorithm in Figure 7. Note that the update of Q values uses the resulting state's best action estimate from that resulting state. This suggests that at each update of Q values, rewards are effectively propagated backwards to early visited states. This "chaining" of rewards and Q values enables sequences of actions to be learned.

In agent applications, the agent must explore and act in the environment and acquire estimates of Q so that it eventually samples enough of the state space to arrive at a policy that yields roughly $V^{\pi*}$ for all states. The algorithm assumes that a large number of samples will

1. Initialise all estimated values $Q(x,o)$ to small random values
2. Given x is the current state, choose action o according to $\arg\max_o [Q(x,o)]$ occasionally choosing an action which has a lower Q-value (to encourage exploration)
3. Execute action o
4. Observe new state y and reward r_t
5. Update Q estimates such that

$$Q_t(x,o) = (1-\eta)Q_{t-1}(x,o) + \eta\left(r_t + \gamma \max_{o' \in O} Q_{t-1}(y,o')\right) \tag{12}$$

   where η is the learning rate, r_t is the reward and y is the state resulting from the action in step 3.
6. Update policy π such that $\pi(x) = \arg\max_{o' \in O} Q_t(x,o')$
7. Repeat from step 2 until converged

Figure 2: Algorithm for Iteratively Learning of a Policy while Estimating Q and V (based on Q-learning)

cause the averages to converge on the theoretical values. However, this does not define a mechanism to implement the policy or how the policy is iteratively refined. This is a broad field and many different approaches have been taken (Kaebling, Littman and Moore, 1996).

Connectionist models represent one way of implementing policy as implicit action selection (e.g. softmax functions over the output nodes of a connectionist network) and then updating the weights according to either a Bellman residual-based error function (for MLP networks using back-propagation of errors) or a localist network where individual weights are trained to represent the value of actions in a mapping from percepts / internal state to actions.

3 Implementing Reinforcement Effects

This section describes different approaches to implementing reinforcement effects. The techniques described here were used to inform the design of MAVIS2 agents and the simulations described later.

3.1 Temporal Difference Methods

In (Barto and Sutton, 1981; Barto, Sutton and Anderson, 1983) the notion of eligibility is introduced, to record or code a period of concomittant activity in abutting neurons so that the US (unconditioned stimulus) and CS (conditioned stimulus) can be processed associatively in the absence of the CS. This is the principle of delayed reinforcement learning – how an adaptive neural system can process the relationship between stimuli presented at some temporal distance, e.g. how a negative consequence of an action can be associated with some stimulus occuring much earlier in training. Sutton (1988) later attempted to unify this as "learning to predict" using temporal difference (TD(λ)) learning. In this framework, certain classes of supervised learning can be viewed as incremental increases in weight towards a desired prediction z. Unlike traditional supervised learning, the successive weight updates are based on the difference between temporally successive predictions (where a factor λ weights the value of these and all previous errors), instead of the difference between a prediction and the desired value z (as in the classic back-propagation supervised algorithm).

(Levine, 1991) Chapter 7 discusses the role of these techniques in relation to psychological principles and control theory. He states that TD learning is a class of incremental supervised algorithms which attempt to predict the environment so as to minimise the amount of negative reinforcement (in the machine learning sense) received. This fact is emphasised by the treatment given by (Sutton, 1988), where the formulation of weight updates is in terms of the partial derivatives of the predictions with respect to the weight vectors (apparently typical of the control theoretic approach used in the development of the Widrow-Hoff procedure which aims for optimality in predictive accuracy). A weight update at time t is given as (Sutton, 1988):

$$\Delta w_{ij}(t) = \eta\,(P(t+1) - P(t)) \sum_{k=1}^{t} \frac{\partial P(k)}{\partial w} \tag{13}$$

where P indicates the prediction made by the learning system. Now, if we wish to discount the value of earlier predictions:

$$\Delta w_{ij}(t) = \eta\,(P(t+1) - P(t)) \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial P(k)}{\partial w} \tag{14}$$

It can be shown that the back-propagation and the Widrow-Hoff procedure are specialised versions of these general forms (see (Ballard, 1997) Chapters 11 and 8 and (Sutton, 1988) section 2). Note that the TD framework has its origin in optimisation theory. The partial

derivatives also assume that all nodes have access to the activations or signalling of every other node. This shared feature of both TD and back-propagation weight update methods makes them less desirable as plausible Hebbian update rules.

Another much cited example is (Tesauro, 1992; Tesauro, 1995) on TD-gammon. Tesauro used temporal difference learning with an MLP to learn to play back-gammon. The success of the system was unprecedented. However the difference between Tesauro's work and an agent is clear. In equation 14, their is only predictive error measured between successive predictions. TD learning, therefore, hinges on the notion of a terminal state in the agent-environment interaction (and this was stated clearly in Sutton's work). The *final* value of P is given as the final reward (indicating how successful the game play was) for the learning algorithm. This can be thought of as a sequence of predictions, computed incrementally, where the ultimate prediction is measured against either "win" or "lose" (in the back-gammon example). Hence the final update of the network incorporates the discounted sum of all previous prediction error derivatives (with respect to the weights) and the value of the final state of the game.

Such optimisation theoretic approaches limit the possibility for ultimately long-term plasiticity in an agent. If optimisation or predictions is desired, then TD learning is perfectly applicable. Sutton (1988) has even shown that TD learning outperforms back-propagation in certain tasks. The localised update rule given here is predicated on the desire to enable individual, microstructural components of the network to self-organise over time. The measure of optimality is less relevant, and probably ill-defined. If the agent's learning algorithm can be shown to approach (by gradient descent or ascent) the optimal region of the weight space, then we assume one final state of the environment (e.g. a game state) that indicates the optimality of the result of the route taken through the space. Such indicators may not be present except in temporally localised periods of agent-environment interaction. In a continuous sequence of interactions with a dynamic environment, it is too constraining to state that a single terminal goal state exists. It is also unreasonable to expect a finer-grained specification of state rewards; it is easy to envisage the opposite extreme of Sutton's 'final state yielding a reward' as 'each state must have an associated scalar reward value'. The latter, of course, being a model of the value function.

Perhaps a more important factor is the long-term achievement of goals, which is independant of the learning algorithm used, such as the amount of resources collected from an environment.

3.2 Related Work on Implementation

(Sun and Peterson, 1998) used an MLP to approximate the Q function. They use back-propagation of errors to calculate weight modifications, where the error on the output nodes

(which is recursively fed backwards in the MLP previous weight layers) is calculated as:

$$error_i = \begin{cases} r + \gamma \max'_a Q(y,a') - Q(x,a) & \text{iff } a_i = a \text{ was the chosen action} \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

where $error_i$ is used instead of the usual difference between desired and actual output, and the remainder of the back-propagation algorithm remains the same. In a similar context to this thesis, Sun and Peterson equate TD learning to playback of the complete sequence of events the agent encounters during an episode of action (which in the Dyna architecture of (Sutton, 1991) was equated to "replaying" of the agent's mental plan).

(Ackley and Littman, 1991) used a bi-partite network intepretation, and then trained it using the back-propagation algorithm. There purpose was to provide both input and response generalisation. Their system (complimentary reinforcement back-propagation) defined the error such that weight updates cause the network to search in the opposite direction of the erroneous output. For example, if the network output for some node i was 1, and this resulted in punishment, the learning rule would try to adjust the weights so that the output moved away from 1 (e.g. towards 0). Essentially, there method is a heuristic which attempts to implement both a plausible searching method over the action space and generalisation over input and action spaces.

Ackley and Littman's work does represent one of the first methods to fully integrate the notions of perception and action generalisation using connectionist principles. Its goals are somewhat different to those here since software agents are largely concerned with discrete responses or actions and generalisation across actions is not necessary. Our approach has been to break the problem in two, focusing on the appropriate neuronal and weight update models to solve the problem of immediate and delayed reinforcement signals in a plausible local connectionist network.

3.2.1 Drives, Goals and W-Learning

We now turn to the recent work of (Humphrys, 1997)[1]. Humphrys solves the problem of learning and action selection in two parts. Firstly, Q-learning enables the agent to learn its behaviour. After this has happened, the so called W-learning procedure enables continuous action selection over the learned behaviour, according to conflicting goals.

According to the distinction between learning how (to perform certain sequences of action) *vs* learning when such behaviour is appropriate, under goal driven behaviour in the connectionist work of Chang and Gaudiano, 'learning how' versus 'learning when' are temporally interactive but structurally separate. However, in the computational reinforcement learning work of Humphrys the learning of behaviour is both structurally and temporally separate; that

[1]This comparison owes a dept to the anonymous reviewer of the paper (Joyce and Lewis, 1999b) who suggested that this similarity be explored

is, his agents learn to behave (using Q-learning) and then learn the action selection problem using W-learning.

Humphrys' work divides the Q-learning problem into sensory sub-spaces, where a number of Q-learning modules use a sub-space of the entire stimulus space and either share the same actions or have separate responsibilities for sub-sets of actions (as shown in Figure 3). In Humphrys' work, he termed these modules "agents", following (Minsky, 1986). A similar approach is implicit in (Chang and Gaudiano, 1998), where the sensors are grouped according to the functional relationships between the relevant CS, US and actuators. However, each component of Chang and Gaudiano's network is simultaneously active and learning. In effect, each Q-learning agent promotes an action (hence, decides on the presence of the action in the response space using the interpretation given in (Arkin, 1998; Joyce, 2001a)) with strength W. However, learning of the individual behaviours must be performed individually before they are allowed to compete in a parallel action selection scheme.

The learning occuring in each of Humphrys' internal agents is mediated by a localised reward function. Hence, for any environment, there will be a group of sensors (corresponding to a stimuli sub-space), a Q-learning component and a corresponding reward function. Potentially, this system is scalable provided the local reward functions can be found. However, Humphrys' agents require (for one environment) 8 local reward functions. To design such reward functions, and to simultaneously choose the necessary learning rate parameters for the Q value estimates and any other parameters associated with the multi-layer perceptron used to approximate the Q values, Humphrys used a genetic algorithm. After iteratively evolving populations, he found a set of reward functions which reflect the relative significance of each reward function to another. Note that the actual rewards delivered to each of the individual internal agents must reflect the relative importance when considered as a population of agents coping with different aspects of the gloabl desired behaviour.

This perhaps highlights the difference between the two modelling approaches. Purely connectionist systems borrow from biological theories of learning, implemented in a way that makes no claim or indeed reference to optimality of behaviour. Chang and Gaudiano assessed their agent's performance qualitatively, by observing whether or not the agent has learnt the desired behaviour. They also only have two goals.

In contrast, those developed from the machine learning perspective begin with a definition of optimal behaviour, then derive necessary models from these basic assumptions. Hence, they tend to have artificial constraints such as a distinction between learning behaviour and then action selection. Humphry's does make the connection with drive theory, however, arguing that the W values are akin to drives in a Hull's systematic theory of behaviour. This is valid, but only if the drives are seen as an amalgamation of the whole set of interacting intervening variables – see (Hilgard and Bower, 1966) for a description of the complete Hull theory.

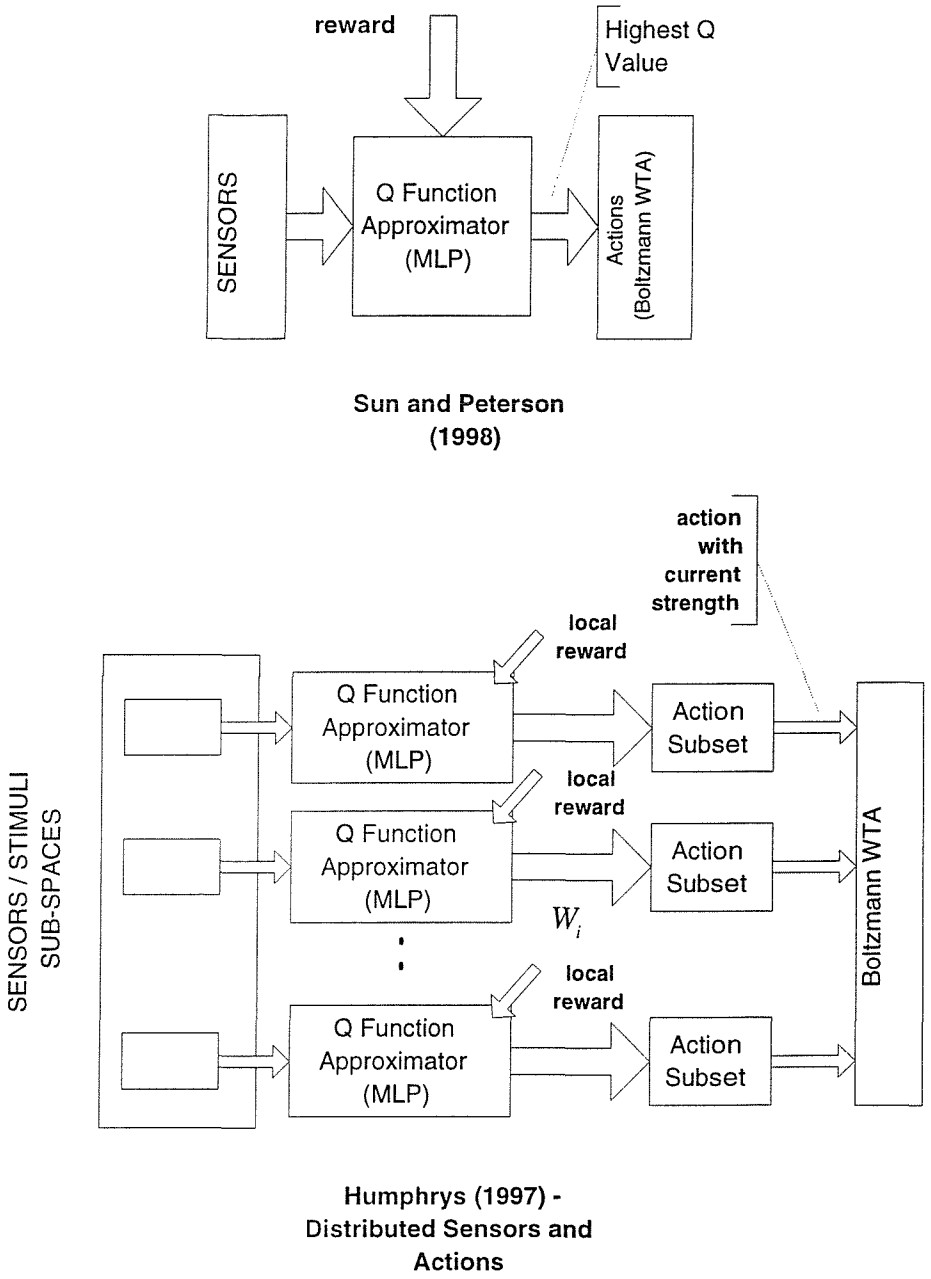

Figure 3: Comparison of archictures – Top: Whole-MLP approximation of Q; Bottom: Distributed Sensors and Actions of W-learning

3.2.2 Learning Action Selection

(Humphrys, 1997) tested many methods for learning W values. They included:

- simple majority rule voting options, in which the action for which most agents voted for are picked
- W value (promotion strength) equals the maximum Q value (that is, the action suggested by the agent with the highest $Q(x,a)$ for any state x is picked)
- learning W values by discrete updates according to the following equation:

$$W_i'(x) = (1-\eta)W_i(x) + \eta\left(Q_i(x,a_i) - \left(r_i + \gamma \max_b Q_i(y,b)\right)\right) \tag{16}$$

  where W_i is the weight promoting action i from a Q-learning agent observing state x and arriving in state y receiving reward r_i. Note the similarity to Q learning update rules.

Humphry's conclusions are somewhat opaque, given that he tested a large variety of methods and it is not clear if these methods parameters or design space were exhaustively explored. However, Humphry's fundamental approach is fruitful because it emphasises the (holistic, single) agent learning behaviour and learning when to select appropriate actions.

The most significant difference to the work presented in this thesis is that we set out to avoid the distinction between learning action selection and learning behaviour. Humphrys' solution divides them to achieve the necessary functionality in his test environment.

3.3 Low-Level Theories of Reinforcement and Operant Conditioning

At this juncture, we pause and consider how other modellers have incorporated reinforcing "signals" in neural network models. There is no universally agreed neural mechanism that implements the kind of generalised reinforcement spoken of in both machine learning and psychological discourse. However, several workers have taken seminal theories and used them as motivation for neural network models. The reason for this digression is that in the machine-learning view of RL, reinforcement is a composite signal, abstracting a number of causal mechanisms into one global indicator of the consequences of actions. This hides the complexity of motivation and drives.

3.3.1 The Response-Selection Method

Response-selection refers to a neural network model which learns responses according to the reinforcement of actions in certain stimuli scenarios. It is the association of reward or punishment *given* responses and the stimuli configurations causing those responses (see section 3.3.4 later).

We mention Grossberg's approach to this problem (Grossberg, 1972a; Grossberg, 1972b) because a goal for this work was to consider motivational mechanisms to produce responses in a connectionist setting. Later, this will be considered as a control architecture for an agent that enables the integration of goal directed behaviour into an agent employing connectionist techniques; the correct motivational mechanism would affect the production of responses in the model. The difficulty is reconciling the multiple levels of analysis (e.g. single-neuron models through to abstract network models such as the MLP) and the correct technical approach.

Response-selection methods for operant conditioning have been implemented by others such as (Chang and Gaudiano, 1998)[2]. They presented a connectionist model based on Grossberg's theory of operant conditioning which directly incorporates motivations and drives. It is from this publication that we draw definitions for the remainder of this work. Initial work in this direction was reported in (Joyce and Lewis, 1999a). A simplified view of stimuli and reinforcement was used and used as the basis for an architecture which would support this type of adaptative behaviour in computational agents. However, learning was implemented as associations between state and action nodes using signal correlation Hebbian learning and the perceptual machinery was implemented based directly on the model of perception and behaviour given in (Page, 2000) and (Usher and McClelland, 1995). It relied on an architecture which assumed that some application domain specific module reported the desirability of the last action. This then placed the agent's structural coupling at a level in which a designer must participate to specify the heurstics which indicate the correct action. It was an agent which used supervised learning, but implemented as a real-time neural network using only signal correlation Hebbian learning.

It is worth noting that "purely" connectionist modellers, e.g. the work summarised in (Schmajuk, 1997) and (Chang and Gaudiano, 1998) on animal-like behaviour, rarely connect with the "purely" reinforcement learning literature such as (Sutton and Barto, 1998). For this reason, the connnection sought here is somewhat difficult. Similarly, the notion of the action selection problem rarely appears. This seems to support the claim that it is a feature introduced by engineering factors or the mode of analysis of behaviour although both problems are still pertinent to the software / virtual agents – see (Joyce, 2001a) for further discussion.

3.3.2 Definitions

Any given stimulus is either a *conditioned* (CS) or *unconditioned* (US) stimulus. Conditioned stimuli have no significant consequences for the agent. The unconditioned stimulus has significance and gives rise to an unconditioned response (UR). This is akin to a rat being consistently shocked (a US) after presentation of a light (CS) causing it to retreat (UR). Eventually, the light will cause the UR as the rat begins to anticipate (fear) the onset of the US. This example

[2]Chang and Gaudiano claim that neither traditional supervised or reinforcement learning mechanisms are sufficiently sophisticated to function in truly unstructured environments. Their's was a robotic agent in a physical environment

is classical conditioning because the agent (rat) is being presented with two stimuli, neither of which are the result of its actions. Purely associative connectionist algorithms can easily model certain aspects of classical conditioning.

The concern of this section is with uniting reinforcement principles with operant conditioning into a connectionist network implementation, where the connectionist learning rule is based on Hebbian associative learning.

Unconditioned stimuli are treated as stimuli which command an unconditioned response, such as retreat. As such, they may be analogised with *a kind of* generalised reinforcement signal if they are combined with a driving mechanism; that is, a node which represents the affective (i.e. emotional) character of the US on the learning mechanism (Grossberg, 1972a; Grossberg, 1972b). However, they are not as general as the reinforcement signal in traditional RL. A US will model a specific event such as shock from a collision. Another may represent an appetitive stimulus, such as food. These are treated as different polarities of the same RL signal (e.g. the appetitive US would be a $r_t > 0$ whereas the shock would be $r_t < 0$) which then modulate learning.

Fundamentally, in operant or instrumental conditioning the US is presented to the agent (by the environment) after the CS and the response have been produced. Hence, any appetitive or aversive consequence (US) is a function of the environment and the agent's action.

3.3.3 Chang and Gaudiano's Avoidance Network

In Chang and Gaudiano's implementation of Grossberg's response-selection mechanism, unconditioned stimuli (such as shocks) are conveyed to a separate reward/punishment module which then modulates learning. That is, if the agent collides with something in the environment, this collision US is conveyed to the reward/punishment module at the same time as the CS (e.g. sensory input) is conveyed to the network. The modulation can then be thought of as activation of a *drive node*. If both the drive node (representing the significance of the US) *and* the sensory node (representing the CS) are sufficiently active, the strength of their association is strictly increased. All other weights connecting non-active nodes simply decrease over time due to decay.

Figure 4 shows a simple diagram illustrating this arrangement. Note that triangular nodes are polyvalent, requiring simultaneous input from more than one source before becoming active. Also, note that the input (CS) nodes are connected one-to-one with the field of polyvalent nodes to the right (whose activations are denoted x_{2i}).

The dynamics of the network reveal how the drive node modulates the learning behaviour. Assume some CS has caused activation x_{1i} and simultaneously, a US was encountered – the subsequent affective state is represented by activation of the drive node. Currently (i.e. before learning) the association z_{1i} between the CS node and the drive node will be close to zero. Assuming a Hebbian update rule, then we wish to indicate a strong association between this

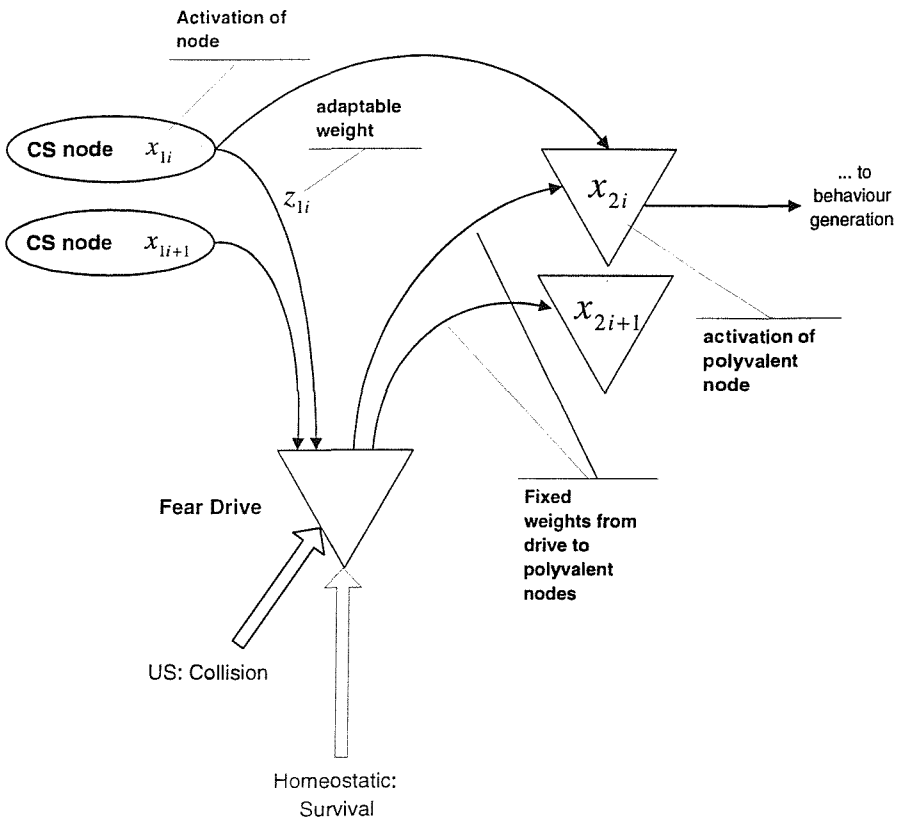

Figure 4: Chang and Gaudiano's Implementation of Grossbergs Theory of Operant Conditioning

particular CS occuring and a fear-inducing state (caused by collision).

Formally, the activation of the drive node is:

$$y(t) = \sum_i x_{1i}(t) z_{1i}(t) - T_y + US(t) \tag{17}$$

This implements the requirement that the CS *and* the US are required for high activation of the polyvalent drive node. T_y is simply a suitable threshold that ensures this polyvalency condition holds. The activation of the recipient polyvalent node which will eventually cause or inhibit certain behaviour (denoted by x_{2i} in the diagram) is determined by the strength of the incoming CS (the activation x_{1i}) and the strength of the signal delivered by the drive node. If both are high (e.g. indicating presence of the CS and a strong activation of the drive node indicating fear) then this node will activate according to:

$$x_{2i}(t) = x_{1i}(t) f(y(t)) \tag{18}$$

where f is:

$$f(y(t)) = \begin{cases} 1 & \text{if } y(t) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

Finally, the only weights which are updated are those from the CS nodes to the drive nodes – effectively, they code the association of "fear" with the presence of certain CS as they grow larger:

$$z_{1i}(t) = Dz_{1i}(t-1) + \eta x_{1i}(t) f(y(t)) \tag{20}$$

where D is the decay rate for the weight and η is the learning rate. Hence, the only method of *increasing* the probability of a certain response is by Hebbian decay of the weight from the CS to the drive node. Counter-intuitively, this is the opposite rule to the reinforcement or associative learning method, where weight increases are associated with promotion of a certain response. The weights in the Chang and Gaudiano network simply *learn to inhibit* by associating CS and US contingencies via the CS-drive node weights.

Different USs can be defined as drive nodes corresponding to goals such as "collect resources". In addition, it is more intuitive for an agent engineering task due to the explicit representation of drives which correspond to goals.

However, we can no longer reasonably discuss reinforcements or punishments which promote or demote certain actions (as in RL) because they are explicitly represented in the model. This is because using Chang and Gaudiano's architecture, weight increments are purely Hebbian and only decay can "punish" a certain weight over time. Such decaying of weights then indicates (to the agent) that the corresponding CS has no significance; e.g. that high activation of a light sensor (a CS) is rarely accompanied by a US (hence, no activation of a fear drive node) indicates that the CS is harmless.

This fundamentally indicates the distinction between low-level neural models and the higher level reinforcement learning models. Reinforcement signals are gross simplifications of the drive node and US pairing that occurs in Grossberg's usual level of analysis (i.e. the neuronal and network realisation of phenomena).

Figure 5 shows an abstract version of Chang and Gaudiano's network, and illustrates how multiple goals (cf. Humphry's W learning model) can be incorporated into the model. Each behaviour (approach or avoidance) is produced as continuous activation of the actuators, but is shown as two networks which control avoidance or approach behaviour respectively. We can (and have) discretised the problem to aid analysis. Assume that the discrete action "approach" is represented by one action (in a softbot) and likewise, "avoid" is implemented as another. If an action is activated then some other system converts this action to continuous motor action. Chang and Gaudiano have implemented these behaviours as two competing systems, each with a separate drive system and different CS concerns. Action selection is therefore implemented as parallel actions competing using the opponent architecture of the neural network. The system they used is a combination of polyvalent nodes, and gated dipole arrangements.

The agent's sensory faculties are divided into two functional units. Those sensory neurons (CS) that are relevant to the production of an avoidance behaviour and are naturally paired with the "collision" US drive node. Likewise, those associated with a pleasurable or appetitive

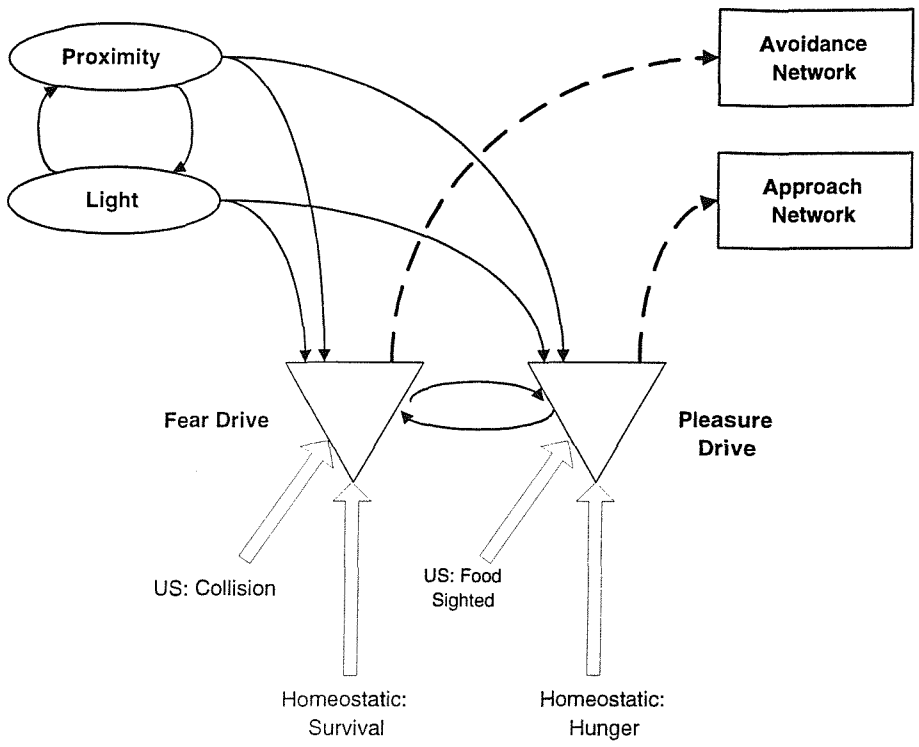

Figure 5: Chang and Gaudiano's Network for Learning Behaviours using Drive Models and Operant Conditioning

motivation (e.g. food sighted) are paired with the light sensors.

In effect, we can see that Chang and Gaudiano have partitioned the whole network into two discrete, competing networks. Each network has a functional role and responsibility and competes with the other. If we reflect upon the design of Q-learning neural net, this suggests that a similar division will be necessary. This is because of the following:

- Drives (low-level) are analogous (but not equivalent) to reinforcement signals (higher-level)
- Reinforcement signals relate to single tasks – e.g. the value of a certain action in a certain state
- Multiple goals implies multiple measures of values of actions – e.g. mutliple Q-value estimates
- Reinforcement signals cannot represent consequences with respect to multiple goals, the values of actions (and states) in achieving those goals and are abstractions upon multiple interacting drives

To summarise, the reward/punishment module is used as a kind of interpreter which knows how to generate an affective response to a US and which then affects or modulates learning. These two features together form what is traditionally represented as the bipolar reinforcement signal representing reward or punishment in the RL literature. The heursitic critic element introduced by (Barto, Sutton and Anderson, 1983) is a similar mechanism which enables the modulation of learning given a collective reward or punishment.

3.3.4 Polemics of Drives and Reinforcements

Historically, according to Levine (1991) pp. 163, there has been tension surrounding whether drive representations (e.g. the USs and drive nodes) are necessary. Grossberg's early work posited the existence of a unified mechanism for both classical and operant conditioning, based on drives and their representation by unconditioned stimuli causing an artificial neuron to respond with activation. Klopf (and therefore, the unifying theory of temporal differences) proposed that the notion of drive representations are not necessary. A drive is simply equated with a sufficiently strong stimulus. Such a strong stimulus could be the reduction in a drive due to its satiation.

Similarly, the notion of optimal behaviour is endemic in Sutton and Klopf's formulation. Both use the notion of differences in activation as the empowering mechanism in learning, and both use predictive accuracy (e.g. Sutton derives the general TD formula by first showing its equivalence to the LMS/Widrow-Hoff linear filter). This implicit behaviour adaptation as optimisation assumption has broad ramifications for agent theory.

The machine learning version of reinforcement is bipolar. Reinforcement which causes the agent to more strongly associate a certain percept with a response is *rewarding* reinforcement. From (Catania, 1992) we find the the psychologist's vocabulary of reinforcement is appropriate if:

1. a response produces consequences (e.g. the agent's environment reflects its actions)
2. the response occurs more often than when it does not produce those consequences (e.g. the response is more likely to occur if the consequences are present)
3. and the increased responsing occurs *because* the response has those consequences

For example, if the agent takes an action, and is consequently presented with an appetitive stimulus such as food, it is more likely to produce that response (given all other factors are equal). In effect, the term "reinforcement" refers to a stimulus (debatably separate from the sensory stimuli discussed) which causes the agent to increase responding in a certain way. Punishment is the complement of reinforcement in psychological literature. Catania reminds us that a stimulus which *acts* as a reinforcer is a description of its role rather than an explanation of how it functions in that role.

No matter what implementation strategy is used (neural network, RL, control theory) the agent is essentially learning via stimulus-response-stimulus temporal contingencies, where the latter stimulus is some indicative of the consequences of a response. In agent terms, some stimulus causes a response which is then followed a short time later by either an:

- appetitive stimulus (rewarding reinforcement)
- aversive stimulus (punishing reinforcement)

This corresponds to the biological and behavioural theory of operant conditioning. According to (Schmajuk, 1997) pp. 179, there are two approaches to the explanation of operant conditioning. An agent learns to produce responses according to either:

1. stimulus-approach: agents learn that certain stimulus situations are appetitive (rewarding) and they aim to approach such situations
2. response-selection: agents learn because a particular response is reinforced in the presence of a particular stimulus situation

where we may take the stimulus situation to be the combination of a perceptual stimulus impinging on the agent (e.g. the state in RL terminology) and a reinforcing stimulus (or reward signal in RL terms). Schmajuk suggests that the approach of (Barto, Sutton and Anderson, 1983) and Grossberg's theory (Grossberg, 1972a; Grossberg, 1972b) are all response-selection. This strongly suggests that the neural nets considered here are also.

3.4 Outline of a Localist Q-Learning Network

We summarise the approach developed so far as the diagram in Figure 6. Note the use of a drive node which "converts" US traces to a global reinforcement signal which modulates learning in the action node population. This represents an outline of the division of "perception" and the coding of perceptual categories to responses used in the MAVIS2 agents described in the next section. We explicitly define an US, which is then converted into a RL signal. In the MAVIS2 architecture, a more primitive model was used with a simple Hebbian learning mechanism used. This was achieved by presenting the reinforcement as an analogue of the learning rate η which either increases the weight between two active neurons or decreases it. In effect, a weight is only strengthened if both the category node and the response node are both active (e.g. have not been shut down by inhibition during the response search) and a reward occurs.

Also, the WTA mechanism is based on pre-synaptic inhibition e.g. (Yuille and Geiger, 1995) where the building of activation on each node is inhibited by similar activity in other nodes in the neuronal population. To explore this mechanism fully, we must consider the real-time dynamics of such idealised neurons. If one neuron builds activation (potential on the cell membrane) faster than others, then it will probably fire first. This firing activity then affects other "slower" neurons by inhibiting their further building of potential in preparation to fire.

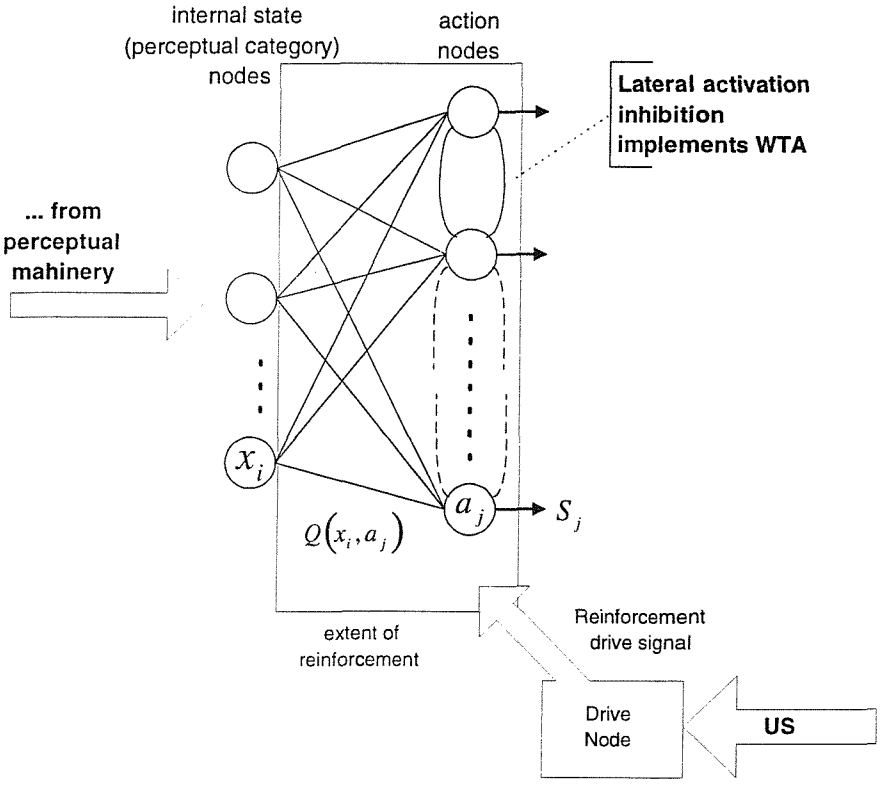

Figure 6: Neural Network model for Reinforcement Learning of Action

The question which must be addressed relates to Arbib's distinction (Arbib, 1995b; Arbib, 1989) of levels of analysis. RL techniques attempt to holisitically reconcile the learning of behaviour using a single indication of consequence – the reinforcement signal. Alternatively, models of operant conditioning based on neural networks and connectionist models (Levine, 1991; Schmajuk, 1997; Grossberg, 1972a; Grossberg, 1972b; Chang and Gaudiano, 1998) use multiple drives to effectively reinforce certain associations between stimuli and responses. They effectively partition the responsibility of the global reinforcement signal into multiple lower level mechanisms based on modulation of neuronal fields connected to motor actuators.

The next section of this report details the precise mechanisms used to implement the MAVIS2 agent, which is a derivative of the approach describe by Figure 6 above.

4 Connectionist Network Design in MAVIS2

This section describes the mechanics of the networks implemented in MAVIS2. The MAVIS2 system is described in detail in (Lewis, Davis, Dobie and Hall, 1997; Dobie, Tansley, Joyce, Weal, Lewis and Hall, 1999; Joyce, Lewis, Tansley, Dobie and Hall, 2000; Tansley, 2000). It continues work undertaken by Radhakrishan and Lewis (1998) by exploring ways of making the agents able to adapt to changes in the multimedia environment.

The agents act as classifiers which map low-level data (basically, multivariate input vectors derived from media objects by common feature extraction techniques) to high-level category labels (that is to say, human-author assigned catgeories). These labels are used to aid navigation in a complex multimedia information space. The techniques described here are concerned only with an implementation of the neural networks used as an attempt to make these agents able to cope with change in the multimedia information space, for example, change in categories or expansion of the available examples in the feature space.

The basic strategy is to use a modified adaptive resonance theory network (ART) which is specialised for Gaussian kernels (so called GART). The observation is that ART reduces to something similar to a radial-basis function network which learns local representations of the feature space. This is a basic unsupervised learning technique which aims to find and adapt clusters which then model regions of the feature space. A discrimination function is then defined which "outputs" a declaration of the feature class which most closely matches any given input. This is then fed to another network (a simple linear associator trained iteratively) which then learns to output class labels from any of the feature-space clusters/categories. What follows is a description of the implementations.

4.1 Modelling the Feature Space with Gaussian ART

The functioning of GART is similar to other unsupervised clustering techniques but with a Gaussian kernel e.g. (Kohonen, 1997). The GART algorithm (Williamson, 1996) implements an unbounded number of category nodes which are represented by a vector representing the mean of the category μ_j, a scalar representing the frequency with which the node *was* the chosen category node n_j, and its multivariate standard deviation σ_j (assumed to be symmetric in each dimension, and σ_{ji} is the deviation in the ith dimension of feature space for node j)

Williamson's work differs from other ART-based techniques because it uses a continuously differentiable kernel; other ARTs form hyper-rectangular discontinuous regions in the feature space (Carpenter, Grossberg and Reynolds, 1991). The major advantage, however, is the conceptual ease with which the sometimes awkward ART formulation can be related to traditional pattern classification through forming discriminant functions.

The probability of an input $\mathbf{x}$ belonging to a category coded by node j is:

$$\Pr[\mathbf{x} \mid j] = \frac{p(j \mid \mathbf{x}) \Pr[j]}{\sum_{j'} p(j' \mid \mathbf{x}) \Pr[j']} \tag{21}$$

where the likelihood $p(j \mid \mathbf{x})$ is assumed to be Gaussian [3]. The prior probability of a node being the source of the input vector is given by the normalised frequency of that node being chosen previously:

$$\Pr[j] = \frac{n_j}{\sum_{j'} n_{j'}} \tag{22}$$

In calculating $\Pr[\mathbf{x} \mid j]$ the denominator of equation 21 is the same for each category node j. We can therefore simplify the expression, and form a discriminant function using the log of the numerator. Hence, we ignore the scaling factors in the multivariate Gaussian likelihood and form the discriminant thus:

$$S_j(\mathbf{x}) = -\frac{1}{2} \sum_i \left(\frac{\mu_{ji} - x_i}{\sigma_{ji}} \right)^2 - \ln \left(\prod_i \sigma_{ji} \right) + \ln(\Pr[j]) \tag{23}$$

The best matching node, denoted J, is then:

$$J = \arg\max_j S_j(\mathbf{x}) \tag{24}$$

General ART techniques then decide whether the node is "activated" highly enough. For any given winning node J, we know that it is the *most* probable category, but is it a *sufficiently* good category ? This is controlled by a vigilance parameter (sometimes equated to the notion of attention in perceptual processing tasks). The implementation of this principle in GART is to check the winning node's match against the Gaussian kernel normalised to unit height; this gives an "independent" measure which ignores the fact that category J was the *best possible* match given the current categories, and now compares its absolute match to the category node's kernel. If this match is too weak, a new node is created.

This is necessary because to form the discriminant function, the logarithm (or any continuous monotonically increasing function) preserves relative magnitudes of $\Pr[\mathbf{x} \mid j]$ for all nodes j. However, for computational efficiency, it removes the need to calculate expensive exponentials and reduces the discriminant function to computation dominated by addition. This means that it will decide upon the *best possible* match, but will not inform us of the closeness of the match with the original Gaussian-shaped kernel. Taking logs of the normalised Gaussian results in a match value:

$$M_J(\mathbf{x}) = -\frac{1}{2} \sum_i \left(\frac{\mu_{ji} - x_i}{\sigma_{ji}} \right)^2 \tag{25}$$

$$\equiv S_J - \ln(\Pr[J]) + \ln \left(\prod_i \sigma_{ji} \right) \tag{26}$$

[3] The choice of the likelihood function is obviously crucial in determining the accuracy of the model. A Gaussian has been used simply because it has been applied generally in a variety of cases, is well understood and belongs to the more general class of exponential distributions, of which the MLP sigmoidal function is also a member

If $M_J < \rho$ the vigilance parameter, then the node is said to be reset and a new node is created to cope with the new input. It is this technique that permits the neural network to grow. Of course, setting the parameter ρ is a task which must be part of the classifier design process. If $\rho \to 1$ then the only the very closest matching categories are sufficient, resulting in most inputs causing a new node to be created (effectively over-fitting). If $\rho \to 0$ then the network subsumes almost every new input into an existing category (resulting in wrongful generalisation). Fuller justification (and more complete derivation of the above results) can be found in (Williamson, 1996; Carpenter and Grossberg, 1987). In the MAVIS2 implementation, the matching test is such that if $\exp(M_J) < \rho$ then a new node is created. Williamson's formulation resulted in massive over-fitting for any scalar $\rho \in [0,1]$.

When any category is chosen and passes the vigilance test, then the kernels of the winning node are modified such that:

$$n_J := n_J + 1 \tag{27}$$

$$\mu_{Ji} := \left(1 - \frac{1}{n_J}\right)\mu_{Ji} + \frac{1}{n_J}x_i \tag{28}$$

$$\sigma_{Ji} := \begin{cases} \sqrt{\left(1 - \frac{1}{n_J}\right)\sigma_{Ji}^2 + \frac{1}{n_J}(\mu_{Ji} - x_i)^2} & \text{if } n_j > 1 \\ \gamma & \text{otherwise} \end{cases} \tag{29}$$

where γ is the initial symmetric radius of all nodes. Note that this parameter can be easily determined by ensuring that unit normalised inputs are given to the network and then $\gamma = 1$. During learning, the radius shrinks to fit the data, so if all nodes begin with $\gamma = 1$, they will gradually shrink from encompassing the whole feature space to to a local region.

The above equations describe a Gaussian kernel-based ART network. We can then use a technique derived from associative learning theory to build a "prediction module" which maps category nodes to the classes. The technique presented is based on a reinforcement learning principle e.g. the network searches for outputs, tests them and then receives an indication of whether it was correct or not, but is never told the *actual* classification vector. In practice, any linear neural network capable of associating the category nodes and classes is usable. The simplest approximation would be to simply store a matrix which associated category nodes with classes. A Hebbian-based learning algorithm can then be used to locally modify associations between the classes and categories on a per-trial basis.

4.2 Classification Prediction Module: Mapping Categories to Arbitrary Symbols

Figure 7 shows the overall agent architecture given in (Joyce, Lewis, Tansley, Dobie and Hall, 2000). During training, the agent performs categorisation, and the resulting winning category node activates one of the *rC* nodes in the prediction module. The activations are then spread in two directions: horizontally via the fully connected weights to the *rR* nodes, and vertically to

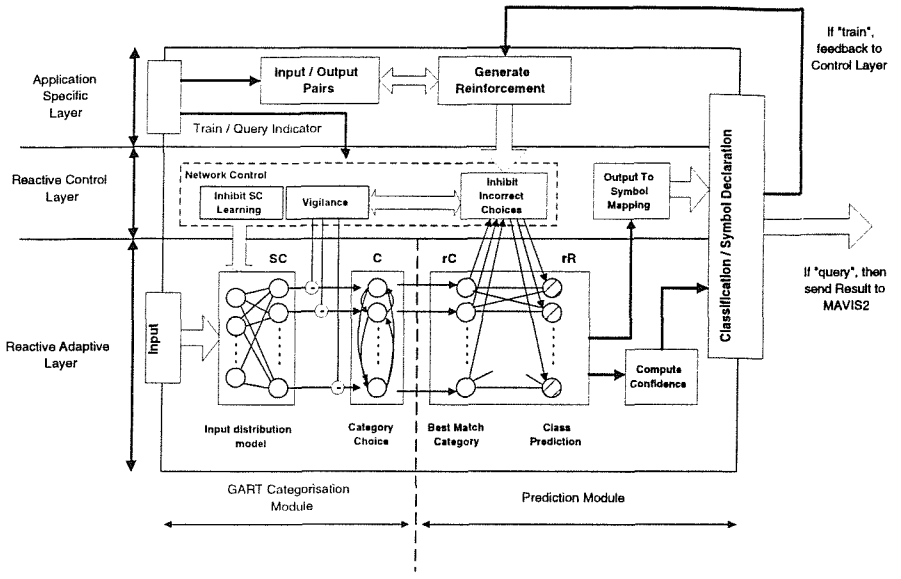

Figure 7: Overall Architecture of MAVIS2 Classifer Agents

the Control Layer which stores an "internal state" indicating the response nodes which should be prevented from activating in response to this category choice. This is because the neural network employed here was devised as a more generic model suitable for adaptive behaviour in agents and is based on the work of (Chang and Gaudiano, 1998) which is based on Grossberg's early work (Grossberg, 1972a; Grossberg, 1972b). Therefore, it was chosen to use associative-search learning principles to search for the correct response (given an indication of the consequence of the previous response, but no indication of the *desired* correct response) instead of the traditional error calculations of the back-propagation learning algorithm (where learning is a function of the actual and desired response). The neural network employs local representations and the weight learning is inspired by reinforcement learning principles. In combination, this enables us to grow the network as new data and class information is made available to the agent. The negative repercussions of this design choice is that we must provide training data as a data structure simulating an environment which gives positive reinforcement only when the agent responds with the correct choice.

4.2.1 Feedforward Prediction

The prediction module can be defined formally as follows. The winning category node excites the corresponding node rC_i in the prediction module. This spreads activation to the control layer which we will represent as a "drive node" which evaluates reinforcement provided by the training data to indicate a successful or unsuccessful prediction. We denote the activation of this node as a_D.

The nodes in the rR layer indicate the response chosen. These are then passed to a look-up

table which is purely an algorithmic device to report the symbol associated with the node (e.g. the text describing the class). The choice of an rR node utilises the notion of feedforward lateral inhibition for contrast enhancement in the response nodes. If a node rR_j is activated strongly by a category rC_i then rR_j forces the activations of all $rR_{k\neq j}$ nodes down, preventing them from becoming significantly activated. This can be algorithmically implemented by softmax-type activation functions. Let the signal (output) emanating from the winning rC node be $S_i = 1$ and I_j and E_j be, respectively, the inhibitory and excitatory input to node rR_j. The activation of the response nodes rR_j is then as follows:

$$I_j = S_D w_{Dj} \tag{30}$$

$$E_j = \sum_i (w_{ij} S_i) \tag{31}$$

$$a_{rR_j} = \frac{E_j - I_j}{1 + L_D \sum_{k\neq j} E_k - I_k} \tag{32}$$

where S_D is the activation of the drive node "inhibit incorrect choices". The activation of the drive node is determined by a *polyvalent* activation rule. This means both a reinforcement and learning signal must be present for the node to activate. For reinforcement, we denote $r = 1$ or $r = 0$ for incorrect and correct response respectively, and the train/query signal is defined as $L_D = 1$ implies training and $L_D = 0$ implies query. These signals are provided by the Control and Application specific layers of the agent architecture. Only if $r = 1$ and $L_D = 1$ does the drive node inhibit previous choices of response nodes rR. Note that during training, the lateral inhibition between rR nodes is switched on, forcing the network to choose a response node. This arrangement is shown in Figure 8 for one rC node.

4.2.2 Inhibiting Incorrect Choices

Denoting weights from every rC node to the drive node as w_{iD}, we can specify the activation and signalling of the drive node in response to the category node as:

$$a_D = rL_D \sum_i (w_{iD} S_i) \tag{33}$$

$$S_D = \begin{cases} 1 & \text{if } a_D > 0 \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

Effectively, the drive node only affects the response nodes if the response choice was incorrect. If it is incorrect, and the agent is in training (e.g. $L_D = 1$) it enters into a cycle of searching for the correct answer (e.g. until the reinforcement $r = 0$). This is effected by quick learning of the weights w_{iD} and w_{Dj} which are then reset when a correct response is chosen. The full dynamics of such a mechanism to implement this searching process are continuous-time dynamics over the weights from the category nodes to the drive, and from the drive to the response nodes. Such a mechanism enables gradual association of reinforcement with correct

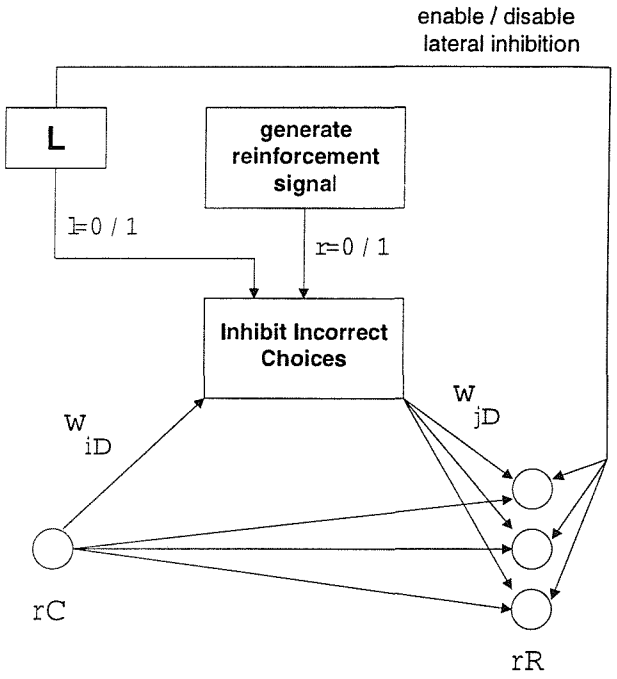

Figure 8: Relationship between Choice and Response nodes incorporating reinforcement mechanism

responses over a number of trials in response to inputs. However, for the MAVIS2 application, we accelerate this learning so that this gradual association is supplanted by instantaneous association of reinforcement with response nodes and the category nodes. This is necessary unless we are to present the same input a large number of times and wait for the w_{iD} and $w_{D}j$ weights to grow (as, for example, they would in a situated robot). A response node rR_j will be selected as a result of equations 31 through 34 and category node rC_i being activated at some cycle t. If this yields negative reinforcement, then at cycle $t+1$ the weights *from* the category node w_{iD} to the drive and from the drive *to* the response node rR_j must inhibit rR_j being selected activated again. This is effected as follows:

1. Forward propagate choice rC_i to drive node (via w_{iD}) and to response nodes rR (via weights w_{ij})
2. Calculate activations of response nodes: equations 31 through 35
3. "Winner takes all" selection of most likely response:
   (a) Select highest signal S_{rR_j} and call j the winning node
   (b) Set signals on all other nodes $k \neq j$ $S_{rR_k} = 0$
4. Assess reinforcement consequence (e.g. if j is the correct classification, then $r = 0$ else $r = 1$)

5. Update weights to drive node: $w_{iD} = S_D S_{rC_i} = 1$ so that the weight from chosen category node i equals unity

6. Update weights associating rC and rR nodes, w_{ij}: $w_{ij}(t+1) = w_{ij}(t) + \eta S_{rR_j} S_{rC_i}$, where $\eta = -1$ iff $r = 1$ and $\eta = 1$ otherwise.

7. If reinforcement is negative, goto 1.

The net result of this algorithm is that while learning, the response network suppresses w_{ij} where the choice of j is incorrect (e.g. generates negative reinforcement) and increases w_{ij} when correct. Step 7 above causes the whole system to cycle until the reinforcement is positive. Step 5 guarantees that the last response selection will be inhibited from being selected again during this search for the correct response.

The reader is reminded that this is a specialisation of a generalised architecture for autonomous software agents. Hence, the reinforcement-based algorithm above enables the network to be grown (because of the localisation of weight/node dependencies) and the use of a search process. The environment (providing correct classifications and reinforcements) could be replaced with a simpler mechanism that learns associations by providing the desired correct response node with high activation simultaneously with the GART module providing the categorisation of the input. We have noted, however, that this algorithm provides no significant time performance overheads in the domains tested.

4.2.3 Final Output Signals

During both training and query modes, the agent will declare an output based on the winning rR node. The final output of the prediction network is governed by each rR node using the following rule:

$$S_{rR_j} = \max\left\{0, \left(1 - \exp\left(-a_{rR_j}\right)\right)\right\} \tag{35}$$

Note that during learning (step 3 of the above algorithm) the control layer will suppress the activation of all but the winning node j and use lateral inhibition to force a winning node. However, during query, all nodes output according to the function above, which merely limits the otherwise unbounded activation values into the range $[0, 1]$.

In order to control the unbounded growth of weights, if the agent architecture detects that a period of training has terminated (e.g. the signal L_D transitions from 1 to 0) it normalises the weight matrix before the next input (query or training).

4.3 A Robust MLP Classifier

MLP networks can be interpreted as approximating the *a posteriori* probability $\Pr[C_k \mid \mathbf{x}]$ where C_k is the classification given an input vector $\mathbf{x}$ (Bishop, 1995; Ruck, Rogers, Kabrisky, Oxley

and Suter, 1990):

$$\Pr[C_k \mid \mathbf{x}] = \frac{p(\mathbf{x} \mid C_k)\Pr[C_k]}{\sum_{k'} p(\mathbf{x} \mid C_k)\Pr[C_k]} \tag{36}$$

The function used to compute the errors which are back-propagated from the output layer through the hidden layer shapes the function the network approximates. The resulting classifier is dependant on the joint distributions of both the target and feature space (see (Bridle, 1990; Bishop, 1995)). For n-layer networks, the classic error back-propagation training rule performs non-linear optimisation of the weights $\mathbf{W}$ by minimising the sum of sqaures error.

$$E_{SSE} = \frac{1}{2}\sum_n \sum_k \|O_k(\mathbf{x}^n) - t_k^n\|^2 \tag{37}$$

where $\mathbf{x}^n$ is the nth vector and $\mathbf{t}^n$ is the corresponding one-of-c classification (or target) vector.

Implicitly, this training process constrains the feature and target space to be normally distributed e.g. the form of the likelihood in equation 36 will be Gaussian. (Bishop, 1995) shows that such an approach is suitable for the approximation of multivariate real valued functions, but not for classification, since the labelling of classifications is typically discrete. For example, we use a one-of-c coded vector e.g. $\langle 0,0,1,0\rangle$ that indicates the classification is "class 3".

Given this, a more appropriate training rule for a general black-box classifier in MAVIS2 would be one that incorporated this target-space model information. Such a technique involves leaving the architecture of the network alone and modifying the training rule. The cross-entropy error function (see (Bishop, 1995) chapter 6) enables us to derive a classifier where the implicit model is closer to that of discrete classification:

$$E_{CEE} = -\sum_n \sum_k t_k^n \ln O_k^n \tag{38}$$

and in addition, the signals from the outputs can be interpreted to be the probabilities of class membership.

In addition to the modified error function, it is necessary to alter the activation function so that instead of being the logistic sigmoid, it is the softmax function. Fortunately, when this new activation function is incorporated into the back-propagation rule, the errors on the output nodes are the same as those for sum-of-squares error. The only modification to the network is that the output nodes must use softmax activations (requiring knowledge of the other nodes' activations) rather than independent sigmoids.

4.3.1 MLP Learning Parameters

The parameter space of the MLP techniques is taken to be the number of hidden nodes and the learning rate parameter. Tested combinations of 2, 4, 8, 16, 32 and 64 hidden nodes with learning rates of 0.01, 0.1, 0.5 and 0.9. In total, both SSE and cross-entropy error functions are tested with each one of the 24 possible network configurations.

4.3.2 GART-based Network Parameters

The local network agent has a simpler parameter space, namely the vigilance parameter, which dictates how precise categorisation should be. For comparative tests, the vigilance parameter was set at ten values in the range [0,1]. Note that because the algorithm is constructive, no other parameters need be set by hand. The GART network has a standard deviation parameter (or radius) for the Gaussian kernels/category nodes. However, all data was normalised between 0 and 1, and this parameter set to 1. The GART algorithm then gradually shrinks the kernels (rapidly at first) to fit the data as sufficient exemplars have been presented. The learning rate for the response network (since it is simply an associative single layer net) can be set according to the application domain and in some cases, a fast 'one shot' learning rule might suffice (e.g. setting all weight learning to be instant and the learning rate is therefore unity). Here, the signal correlation is the strength and direction of the update and a learning rate of unity was used. Weight normalisation after periods of learning activity enabled the weights to be kept bounded. In principle, both the radius and the learning rate for the associative network can be optimised. The vigilance is the only parameter which is varied. The agent must perform even in the absence of "good" data which would enable robust classification. An autonomous agent should attempt to provide the classification service even when the data is weak. It should be capable of learning incrementally, so it can improve if or when more data becomes available. For this reason, no attempt was made to optimise the local GART/associative search network parameters and it was tested 'as if' against the more robust MLP-based classifiers.

4.3.3 Data Sets

The data sets chosen were: a) iris data set from UCI(Blake and Merz, 1998) b) HSV-based histogram features from a MAVIS2 data set c) RGB-based histogram features from a MAVIS2 dataset d) 3 clusters of Gaussian distributed data. The rationale was as follows : a) is a standard machine learning data set, and is of relatively compact feature dimensionality with adequate samples. Both b) and c) are grossly under-determined datasets, each containing 208 samples (one per image) of 27 and 96 features respectively – this represents a case where the agent must try its best to perform classification in a difficult multimedia problem. The actual data is HSV (27 features per image) and RGB (96 features per image) colour histogram feature data obtained from the Victoria and Albert image data set, which was used as a test bed application for the MAVIS2 system. Test set d) is a simple two-dimensional problem, which contains enough samples (150) to train on and is present to test whether the classifiers and agent are working as expected.

4.3.4 Performance Measures and Training Schedule

Each result presented is the average over 10-fold cross-validation testing (Cohen, 1995); this enables us to assess the "memorisation" of the training set and (more significantly) its gen-

eralisation performance on unseen data. Validation sets are generated from 10 independent samplings drawn from the original data and are kept back during training. This ensures that generalisation performance on unseen queries is not attributable to fortuitous choice of a hand-picked validation set. Each data set is shuffled into random order and then divided into 10 sets (folds). The classifier is trained on 9 of these. The classifier memorisation performance is tested by presenting the data from the 9 training folds and generalisation is tested by presenting the held back fold. Then, a new classifier is generated and trained and a different fold held back for validation. This is repeated for each of the ten sets, and classification performances averaged over these 10 different training conditions. For each setting of MLP learning parameters, there are ten classifiers trained and the performances are averaged.

Performances are measured as follows. For each configuration of folds, the agent is trained by passing through the 9 training folds. One pass through each training pair in the 9 folds of training data is called a *pass* through the data set. Training proceeds by propagating errors back after each training pair is presented (instead of batch learning where errors are accumulated over the whole pass). An epoch is 50 passes through the training set. After each epoch, the 9 training folds *and* the generalisation test (the held back fold) are presented to the classifier and the classification performance measured on both. This is repeated after each training epoch. Performance measures are therefore on *actual* classification performance, rather than any internal error measure (e.g. by summing the output errors over an epoch). Generalisation and training performances are continually recorded and the result presented is the best performance obtained during the entire training period (which was 50 epochs).

Naturally, such an exhaustive search of the parameter space and training in the fashion described is impractical for an agent in practice, but it reveals the classification performance and training properties while also illustrating the difficulty of automatically training classifiers in multimedia environments. These tests are to assess the classification performance rather than suggest a reasonable method for training MAVIS2 agents *in situ*.

We present the results in the table below. The first column indicates the network used; cross entropy error (CEE), standard sum-of-squares error (SSE) and the local GART-based agent (AG). The second and third columns are best training and validation set performances over the whole parameter space tested. The remaining columns report the number of hidden nodes and learning rate (or vigilance in the case of the local network technique) which gave the best performances, plus the mean number of epochs required to train the networks and finally the standard deviation (s.d.) of this value. A low s.d. indicates that almost always, the best performance can be achieved in the mean number of iterations, whereas a high s.d. indicates a large variation in the number of iterations required over multiple trials. It could be argued that for autonomous agents, the training behaviour must be predictable and this value should be low.

ANN	Data Set	Best TS Performance	Best VS Performance	Hidden Nodes	Learn Rate	Best Performance at epoch	Standard Deviation of Best Epoch
CEE	Gaussian	100%	100%	64	0.01	1	0
SSE	Gaussian	100%	100%	64	0.01	1	0
AG	Gaussian	98.9%	100%	n/a	0.6	1.3	0.4
CEE	iris	94.5%	99.3%	64	0.9	4.4	3.6
SSE	iris	94.7%	98.6%	32	0.01	12.7	9.8
AG	iris	95.5%	99.3%	n/a	0.7	3.2	2.2
CEE	HSV	50%	50%	64	0.1	9.5	6.5
SSE	HSV	45%	49%	32	0.01	17.7	12.5
AG	HSV	24.4%	39%	n/a	0.2	3.9	3.1
CEE	RGB	28%	46%	8	0.5	20.3	15.3
SSE	RGB	32.6%	45%	32	0.01	9.6	11.9
AG	RGB	29.9%	38%	n/a	0.7	3.2	2.2

4.3.5 Discussion

The table shows that for all data sets except the artificial pure Gaussian data the standard deviation of the best performance epoch is lowest for the GART/associative network technique. However, the classification performance of the technique in its present form is below the MLP networks and further tuning is required. As expected, all three techniques worked well on the simple Gaussian data set. The HSV and RGB features sets indicate that for low numbers of features, the vigilance should be low and conversely, high for high numbers of features. For multiple-independent attributes and sum of squares error, the learning rate parameters seems to be less predictable.

More importantly, the implications for implementing classification agents in MAVIS2 are that despite poorer performance, the local GART-based network with a simple associative map is easier and quicker to train in situ. In the prototype implementation of the MAVIS2 system, the GART-based agent was implemented because of its ability to perform reasonably and only the vigilance parameter needed setting. In responding to queries, the most important problem that arose was while classifications were correct, the confidences reported were usually very low which would present problems for the future development of a fusing method based on probabilistic measures of class membership (if sufficient agents were trained over a variety of media types for each class, then voting could be used and this confidence measure is less significant). In addition, the prototype of MAVIS2 required agents to train quickly on the data available at the moment the system starts. The local network agent trained consistently faster in the comparative tests above and even with the arbitrary setting of radius (for the GART network) and the learning rate (for the associative learning net) the agents still performed well. Future work will need to focus on how an MLP network might be configured to cope with such "sufficing" tasks, where good performance can be guaranteed without the exhaustive search

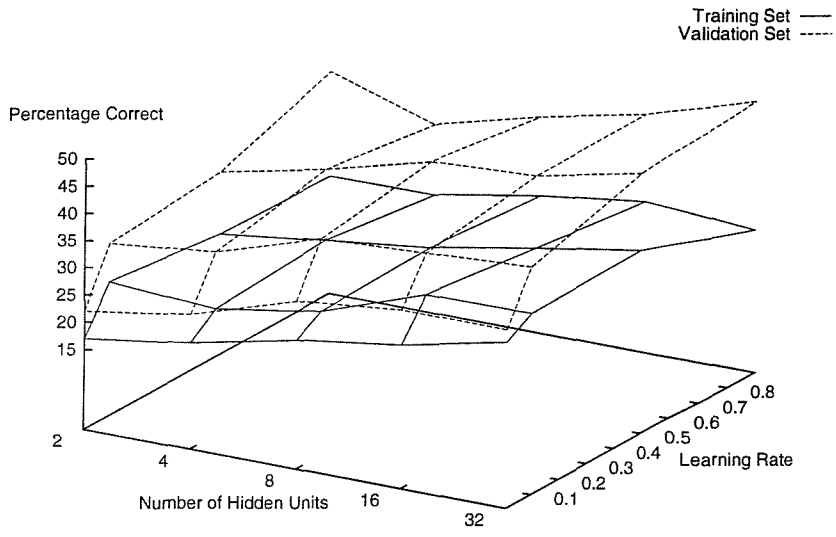

Figure 9: Validation *and* Training Set Performance – Parameter Space for MLP trained using back-propagation with Cross-Entropy Error Function

of the parameter space. While the data set available during the development of the MAVIS2 prototype is believed to be typical (e.g. under-determined in terms of class exemplars) it is possible that the GART-based agent performed well in a unique case. The exhaustive testing described above was an attempt to refute this claim, and performances obtained would suggest that the heuristic approach used to build the agent's classifier might be a fruitful basis for future development.

Further, some diagnostic data was collected to ascertain how difficult picking parameters for the MLP or the vigilance parameter for the local-network based agent might be. The dataset used for the example presented below was produced from a subset of the Victoria and Albert Museum Artifacts image base. This resulted in 100 individual feature vectors (samples) of 23 features, unevenly distributed amongst 12 classes. Foley's heuristic (Foley, 1972) suggests that three times as many samples *per class* are needed as there are features. Hence, for a well-determined dataset we seek over 800 samples. However, this serves as an interesting problem to examine the conditions of the parameter space which must be searched and is not untypical of the scenario we might be presented with in a real multimedia retrieval problem.

The surface shown in Figure 9 represents the best classification performance on the validation and training data sets. Note the scale of the Hidden Units axis. Recall that the validation performance is a better indicator of the network having learned a model of the data as opposed to simply "memorising" the training set (which results in poor performance on unseen inputs). The qualitative properties of the surface are that the best results appear to lie at the extreme of

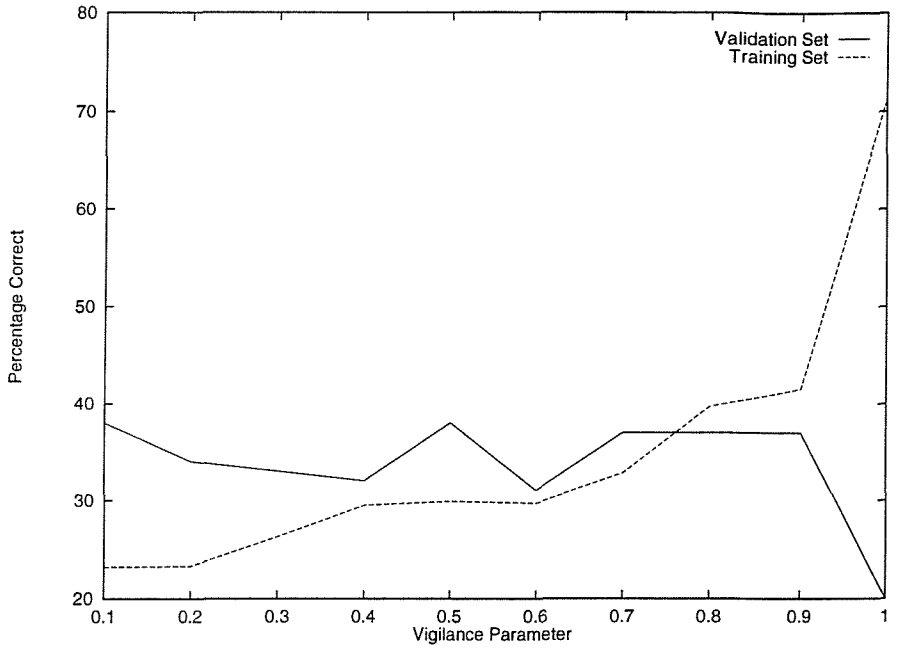

Figure 10: Parameter Space for Gaussian ART Network with Local Learning Associative Search Classification Network

the parameter space, with the exception of one other peak result at $\eta = 0.4$ and 8 hidden units. The surface certainly does not indicate a desirable global maximum, clearly distinguishable from the other network configurations.

This suggests that a further search process (other than exhaustive search) over the space of MLP parameters might be difficult (and more so if extensions to back-propagation are used such a momentum) due to the lack of obvious maxima in the classification / parameter space above.

A similar parameter space for the localist technique on the same data subset is shown in Figure 10. Note that at $\rho = 1$ the network is over-fitting, and therefore should be discounted. The optimum value would appear to be 0.75 since the network performs equally well on both training and validation data.

The trend in behaviour is more predictable on the same data set. While certainly not conclusive, it suggests that on sparse data sets, the relationship between generalisation and memorisation of the training set is easier to ascertain.

Further work would need to focus on improving the classification performance and the confidence measures used. For example, if voting is to be used, then large numbers of simple classifiers might be used, for example, simple sub-space classifiers might be more applicable for training using single-layer networks. Using heuristic methods (such as the combination of GART and a linear associative network) achieves some of the design goals of connectionist agents, but from initial results, robust statistical performance is compromised.

The final contribution was a first attempt at defining a realisable agent architecture which enabled "internalisation" of training e.g. was flexible enough to use associative search or supervised training in the form of complete feature / target vector pairs. The associative search network implemented was a first attempt similar to (Chang and Gaudiano, 1998).

5 A Simulation Environment

5.1 Introduction

In developing the proposals of (Joyce, 2001b), reference was made to situatedness; the embedded, continuous participation of the agent in its environment. In an attempt to study the relevant phenomena of situated agency (e.g. the establishing and manifestation of routine activity and intentional arcs), a simple environment was designed. This section recapitulates and describes that implementation, and concludes by illustrating how this simulation can be used in exploring reactive agency. In section 6 and 7, empirical data obtained from experiments with the following simulation are presented in their entirety.

5.2 A Simulated Environment for Reactive Softbots

Most reactive architectures are designed to cope with goal directed behaviour in spatial, topological environments. (Brooks, 1997) surveyed 42 papers appearing in the journal *Adaptive Behaviour* between 1992 and 1995. From these papers, he found that 5 papers dealt with agents without spatial location or a notion of topology e.g. adjacency, proximity and neighbourhood. Brooks' motive was to demonstrate the lack of "real" situated robotics research. However, this equally demonstrates that simulations and environment models which are most closely suited to the type of agents considered as software or virtual agents, are also somewhat scarce.

The principles motivating the choice of model are as follows :

- *Absence of spatial topology or environment geometry* – softbots may not have spatial locations. Imagine a software agent situated in its environment, but (as (Etzioni, 1993) illustrates) connected to the environment by discrete actions having no intuitive spatial interpretation such that to compute the Euclidean distance between two agents may be meaningless. An information retrieval agent will effect actions based on perceptions of the environment, although there is no logical interpretation or analogy of spatial context for the agent.

- *Adjacency (if any) is arbitrary* – softbots will, more than likely, inhabit domains where the analog of multi-agent spatial adjacency is defined only through communicative or social actions, and the medium for transmitting that information. Only if such media exist are agents aware of their peers or social groups.

- *Uncertain and dynamic environments* – although in this category, the manifestation of uncertainty is different. Firstly, the agent's virtual sensors are unlikely to be noisy. There will, however, be uncertainty in the communication medium which might reasonably be treated as noise for both the sender and recipient of communicative activity. Environments "lie" such that an agent might be informed that some action is permissible, but in

fact this is an inaccurate reflection of the consequences of that action because of implicit latency of the information arriving at the agent's sensing faculties. Dynamic environments are those which affect and are affected by the agent – this property underpins the establishing of routinised activity, since this provides opportunities for sequenced actions with utility which can be repeated.

- *Softbots must be situated* – the agent's action has an effect on the environment and the environment, which is sometimes reflected to the agent. This condition was used by (Jennings, Sycara and Wooldridge, 1998) to differentiate GOFAI from agents research. (Varela, Thompson and Rosch, 1991) describe this as characteristic of *enactive* cognition. However (Maturana and Varela, 1980) encapsulate the notion as structural coupling, being the recursive reproduction of systems (e.g. the relation between agent and environment as a couple) which are mutually affective.

- *model simplicity* – the environment must be as simple as possible to enable analysis. Brooks' criticism of 'toy' environments parallels that of 'micro-worlds' explorations criticised by (Dreyfus, 1992). These arguments are not really criticisms of methodology, but of the generalisations and illusory abilities of agents which are extrapolated from those simulations. Similarly, Agre (1988) used blocks-world simulations. In this respect, the simulation presented here enables analysis while containing the necessary features of a model enabling the study of reactive agency. To quote:

  > "Following the tenets of interactionist methodology, the focus is not on complex new machinery but on the dynamics of a relatively simple architecture's engagement with an environment" – (Agre, 1997),pp.105

These ideas are not difficult to imagine, particularly in the context of distributed information systems. Communication substrates or infrastructures are rarely fault free, so the agent must be tolerant to "noise" or uncertainty in the information conveyed to it by such a substrate or infrastructure. The source of this noise is irrelevant. The fact remains that either in the sensor or the environment, the information used for perception is not completely certain and reliable.

One model which *does* appear to comply with the principles above is (MacLennan and Burghardt, 1994). Their environment is designed to explicitly model evolutionary aspects of communication. It is an example of an environment which lacks an explicit topology. The model developed here was explicitly tailored to enable the exploration of artificial neural network techniques and architectures, migrated from situated robotic agents research, to be tested for viability in the 'virtual' or softbot domain. This can help answer the question of whether softbots are actually co-extensive with robotic agents in terms of a general theory of agent science as defined by (Huhns and Singh, 1998).

5.3 The Environment

The experiment here is based on the classic Skinner Box conditioning apparatus (Skinner, 1938) hybridised with the k-armed bandit models used in machine learning studies of reinforcement. An agent must learn that acting in a certain way will result in some resource or food (an appetitive stimulus) being delivered, but that alternative actions can provide punishment (aversive stimuli). Before further discussion, we note the following terminological points: a *resource* is something the agent is designed or goal-directed to collect or acquire, while a *reward* is a reinforcement that indicates no aversive consequence of an action. Therefore, this agent simulation is concerned with one goal directed behaviour (to acquire resources) while actually learning how to use its response/action repertoire to achieve this, *and* this learning process is operant in nature such that the agent can receive rewards and punishments.

In terms of *equipment* which can participate in absorbed circumscriptive activity, the agent is presented with a button which has toggle-state; it can be on or off. The agent will be goal directed by design, and it should try to collect resources from the environment. These resources are delivered when the button is depressed (on) and not when it is off. However, the agent has no model of the relationship between button states and consequences.

With this very simple experiment, the agent would (trivially) have to learn that depressing the button and leaving it in this state would result in maximal resource collections and, presumably, it would receive no punishments. This is, perhaps, the simplest stationary deterministic environment and easily soluble by traditional reinforcement learning and artificial neural network techniques. Similar experimental apparatus were described by Spier and McFarland (1998). They additionally tested an outcome-devaluation effect using a Skinner box-like experiment, but were concerned with the relationships between necessary cognitive capacity and performance. They also use drive reduction (e.g. a lack of arousal or agitation) as a modulator of reinforcement value.

However, the following changes are made: Whenever the button is depressed (i.e. in the 'on' state), a resource is returned. However, there are (at any moment in time) a finite number of resources available, which is decremented each time the button is in the 'on' state. By analogy, imagine a softbot agent trying to secure time on a multi-tasking operating system. A finite number of resources (time slices or quanta) exist and one agent, on behalf of a user, cannot possess them all despite its requests (the button being in the depressed state).

The environment avoids being maximally exploited by trivial greedy policies (e.g. press and hold the button indefinitely) by delivering both resources *as well as* rewards/punishments. At any moment in time, the environment can deliver a resource, but in addition, will deliver a neutral or punishing stimulus (by analogy, the agent might be given a shock in addition to the resource). This is governed by a simple rule; as the resources are depleted, the environment is more likely to shock the agent to prevent greedy policies from being learned and established as routines. Thresholds dictate the levels at which the environment will deliver punishments as

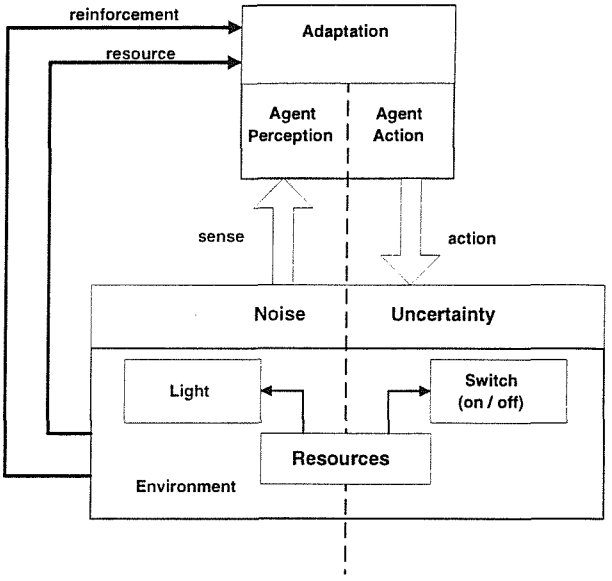

Figure 11: A schematic of the simulation environment

well as resources.

The level of resources available is reflected to the agent via a communication medium. In the Skinner box analogy, it might be a light indicating the level of resources currently available. The metaphorical light represents the level of resources in the environment, and the agent must learn that it is indicative of the probability of punishment (aversive stimuli), since as the consumption of resources increases, the agent is more likely to receive a punishment.

In the operating systems analogy, in order that the environment (the operating system) is not drained of resources, a sustainable number of free quanta are kept available to prevent a single task dominating the processor.

However, as noted before, this 'light' is not entirely reliable, and does not provide deterministic evidence of the chance of aversive stimuli being delivered. Figures 11 and 12 show the apparatus of the environment and the "agent requesting" metaphor respectively.

The fact remains that the agent's action should be informed by a perceptual indicator – the light. The probability of a punishment is proportional to the light's state, but with stochasticity. This perceptual information is inherently noisy and not necessarily enumerable. If the button is on, and the resources are below a threshold, a punishment is delivered. A resource might also be delivered, if any remain, but the agent is clearly "told" that this resource came at a price.

Finally, the environment is regularly refreshed. The original level of resources is replenished and the environment state reflects this, both in terms of what the agent can perceive via the 'light' and the punishments delivered along with resources.

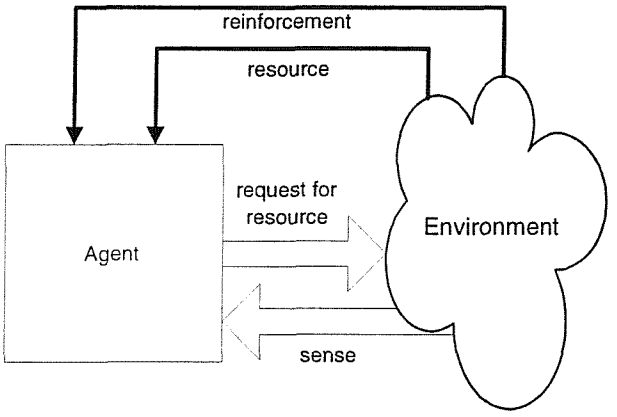

Figure 12: The traditional "agent metaphor" of resource allocation in an uncertain distributed system

5.4 Demands of the Environment

The agent must therefore adapt and learn to exploit the environment to attain its goal. This implies that:

1. the agent must learn relationships between the perceptual information available to it (i.e. the light state and the button state) and the consequences of acting
2. the agent must also learn the 'mechanics' of available equipment. That is to say, the button has state and the light reflects probability of punishment
3. the agent's actions will directly affect the success of a learned policy
4. the agent establishes routines of regular behaviours which reflect the risks of taking actions in different environmental states

5.5 Mechanics of the Simulation

As stated before, the button is really a switch and has two states; on or off. If the button is on, a resource is delivered and if the button is off no resource is delivered. In terms of disembodied softbots, we might reasonably see the button as a mechanism which causes a request for a resource. The resources are diminished linearly, with every resource dispensed.

The light or indicator of the resource level state, is given by a function of the resource level. Let $rs \in [o,n]$ denote the integer number of resources currently available, $ls \in [0,m]$ denote the integer-valued light state and $\tau_{danger} < \tau_{neutral} < \tau_{safe} \in [o,n]$ denote the thresholds such that:

$$ls(t) = f : rs(t) \rightarrow [0,m] \tag{39}$$

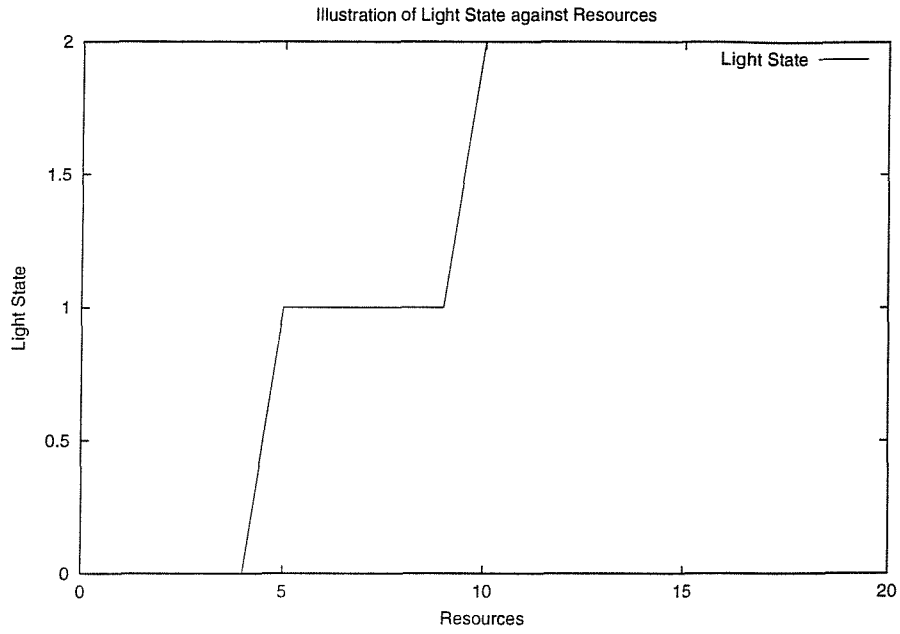

Figure 13: Example of light state as a function of resources

where f is a discontinuous function of rs mapping the resource state to the light state according to the thresholds, and m denotes the maximum integer indicating the "safe" state of the light for example with $m = 2$ then $\langle (danger, 0), (neutral, 1), (safe, 2) \rangle$. Figure 13 shows an example where $\tau_{danger} = 5$, $\tau_{neutral} = 10$ and $\tau_{safe} > 10$.

The following environmental rule is defined: if the light indicates "safe", then the environment will almost certainly *not* deliver a punishment with the resource (analogously, the food is delivered with no aversive stimulus). If the light state is "neutral", then the probability of a shock being delivered (alongside the unit resource) is approximately 0.5, so an agent would be taking a chance if it opted to continue requesting resources when the light shows "neutral". If the light state is "danger", the agent will almost certainly receive a punishment and *possibly* a resource (depending on rs).

This is implemented as sigmoidal function of $ls \in [0, 2]$. The implementation is as follows. Using a uniform random variable v, and denoting shock (aversive stimulus) as $r = 1$ and no shock as $r = 0$, the probability of $r = 0$ being delivered is:

$$\Pr[r = 0 \mid ls(t)] = \begin{cases} 1 & \text{if } B(ls, c) > v \\ 0 & \text{otherwise} \end{cases} \tag{40}$$

where the function B is a transformed sigmoidal function, and c is a variable indicating the

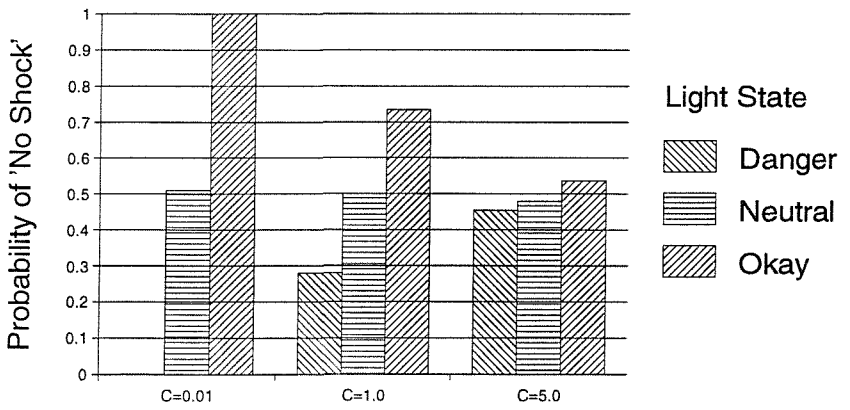

Figure 14: Histogram showing the probabilities of 'no shock' for different certainty values

certainty of the environment such that as $c \rightarrow 0$ then:

$$\Pr[r = 1 \mid \text{'danger'}] = 1 \tag{41}$$

$$\Pr[r = 1 \mid \text{'safe'}] = 0 \tag{42}$$

As c increases, these probabilities shift until the light state reflects very little about the probability of shocks being delivered for example:

$$\Pr[r = 1 \mid \text{'danger'}] \approx 0.5 \tag{43}$$

$$\Pr[r = 1 \mid \text{'safe'}] \approx 0.5 \tag{44}$$

This is implemented as:

$$B(ls, c) = \frac{1}{1 + \exp\left(\frac{-(ls-1)}{c}\right)} \tag{45}$$

Figure 14 shows the probability of $r = 0$ against values $c = 0.01$, 1.0 and 5.0 respectively computed from 1000 samples. Note that at $c = 5.0$, the light state has practically no utility in aiding an agent in deciding whether a shock is likely to be delivered, since the probability of shocks is practically equal with all light states.

There is no dispute that this environment is Markov. However, the utility of the light can vary, and in addition agent's actions change the probabilities of shocks. An agent must therefore assess both the utility of the light with respect to the environment, and then decide whether to proceed with resource requests or wait. The probability of getting a reward, given a light state, can also be shifted over time by altering c. Most learning algorithms assume that the environment will be stationary. This simulation enables non-stationary qualities to be explored by: varying c, changing the thresholds τ or altering the operational properties of the equipment (the button).

5.6 Agent Goals

We are now in a position to state what the agent's goals are. The agent must establish a set of behaviours which enable it to 'survive'. That is, some internal measure of goal completion (e.g. energy) which measures how successful it is in negotiating the environment to collect resources. Recall that it *cannot* simply maintain a greedy policy for two reasons. Firstly, this will incur a number of punishments. Secondly, we wish to encourage the agent to learn a maintainable routine, which might (for example) be something analogous to the proposition: "collect resources by holding the button on until the light state indicates (from experience) that it is too risky to do so; then adopt a routine which minimises energy loss until the environment is refreshed". Further details of how the agent's control architecture needs to be configured are given in (Joyce, 2001b).

5.7 Agent Actions

The agent is provided with three actions :

1. NO-OP : take no action
2. PRESS : activate switch, causing it to be in the "on" state if not already
3. RELEASE : de-activate switch, causing it to be in the "off" state if not already

The agent does not understand the relationship of its actions to the environment and equipment state. For example, there is no implicit coded knowledge about the reciprocal relationship between the PRESS and RELEASE actions. The agent will also pay a cost for each action. To effect either PRESS or RELEASE, incurs some penalty in terms of internal energy, but NO-OP costs nothing. The aim of this is to encourage the agent to recognise situations where doing nothing is the best policy to preserve internal energy and not incur further punishments.

5.8 Simulations and Performances

It is necessary to assess the behaviour as a repeatable phenomenon, since the design goal is not to learn and perform in the environment once, but to sustain performance *and* necessary learning during the agent's lifetime. The agent must acquire learned sequences of actions, and be able to repeat them as and when appropriate. The agent must minimise punishment, but attain its goal of collecting resources. In order for this to happen, the agent will receive some punishments, because they are one of the indicators of an action's consequences. However, we require a joint measure of performance that includes the punishments received and the resources collected.

Since the agent is attempting to sustain homeostatic state, measures based on optimality of resource collection are inappropriate. This will be evident when the agent's control architecture is explored. Briefly, the agent's internal energy is used as an indicator of goal

completion, but to encourage the establishing of routine actions and deter greedy policies, the secondary reinforcement (a joint measure of goal attainment and punishment) is modulated by an outcome-devaluation mechanism. This means the agent will not quickly learn to adopt state/action pairs associated with resource collection when the drive is satiated. Analogously, if an organism has just eaten, it is not as motivated to find food.

A *cycle* is defined to be a period of time between refresh periods where resources are present in the environment to support the agent. For example, if the environment is replenished every 100 iterations of the simulation, then a cycle is 100 iterations in length. For each cycle, we measure the agents punishments and mean drive/agitation level. These criteria will measure both the agent's ability to learn its task while ensuring it does not simply learn the trivial solution. The trivial solution will yield a first cycle performance, and then zero punishments and zero resources in all subsequent cycles. Also, in assessing agent performance, we may wish to discount the first few cycles, since the early exploratory phases of learning and participation may skew results.

In order to test the capabilities of connectionist agents in this environment, two designs were built and tested. Full details can be found in (Joyce, 2001b). In summary, the two implementations are:

- A localised bi-partite network consisting of a self-organising, constructive perceptual system (based on a simplification of ART and specifically, the GART mechanism described) and a localised Q-learning implementation (the configuration of which was inspired by motor/action circuitry in mammals –see (Joyce, 2001b) chapters 3,4 and 8). This architecture was later exploited to study routine behaviour and the adaptation of the agent to shifting environment configurations.

- A distributed MLP: this network takes the agent's perceptual apparatus (i.e. sensor arrangements) and maps this onto output nodes via an intervening hidden layer of neurons. The network is trained using the Bellman residual, e.g. see also (Sun and Peterson, 1998). This enables the MLP-based agent to approximate Q values by exploring the environment using a traditionally strictly-supervised back-propagation algorithm.

The MLP has been tested in a variety of situations (see the review in section 3 and 3.2 earlier) and proven to be stable. The next section describes some analyses of the perceptual mechanism for the local network based on ART. Essentially, the percept network uses Gaussian kernels in a way similar to radial basis function networks, but with the scalar vigilance parameter of ART. The network was tested by making the agent select actions randomly and the effects of this random action taking was statistically controlled over a number of runs. The aim of the experiments in the next section is merely to demonstrate the relative stability of the stimuli state space mapping produced by such local networks. A variety of vigilance, noise and environmental certainties were tested to ascertain the effects of these simulation parameters on

the growth of the network. Section 7 describes the comparative tests performed on the MLP-based agent against the full local network-based agent (e.g. where the Q-learning network as well as the perceptual network was enabled).

6 Mapping Stimuli Spaces - Experimental Results

The following pages show graphical results obtained from the perceptual mechanism of the agent. First, three settings of vigilance were tested in order to establish a reasonable basis for continued experimentation with the perceptual mechanisms; $\nu = 0.50$, $\nu = 0.75$ and $\nu = 1.0$. This revealed that, as expected, the high vigilance produced many more categories than low vigilance, and the medium vigilance parameter was chosen.

This was necessary so that the behaviour of the system could be understood in advance of trying to engineer a control architecture for the agent. Figure 15 shows the results in terms of catagory node recruitment for the three parameter settings.

The number of nodes created and the vigilance parameter will all have effects on the dynamics of the agent's perceptual novelty estimates.

Figures 16 through 19 show the results of increasing the sensor noise. Note that Figure 19 shows results where the level of noise can shift a given stimuli from its 'correct' category (e.g. that category node formed to cope with stimuli of that class) to another 'incorrect' category resulting in a category error. As expected, the higher the noise, the more probable the agent must recruit more nodes to cope with the diversity of noisy stimuli. In each case presented, the results show the average over ten runs, the standard deviation (which is often quite large) and the vigilance parameter was $\nu = 0.75$.

Figures 20 through 24 show the effects of shifting environmental uncertainty. In each case presented, the results show the average over ten runs, the standard deviation (which is often quite large) and the vigilance parameter was $\nu = 0.75$. As uncertainty rises, the number of nodes created in total (e.g. at the point where the curves begin to approach asymptote) grows and the recruitments occur at different (usually later) times during the experiment. This is attributable to the agent being less able to rely on reward or punishment and the subsequent altering of behaviour that results causes the agent to experience different and unfamiliar energy states. This requires the recruitment of more category nodes. The important qualitative result is that the recruitment behaviour does not radically change under uncertain environments.

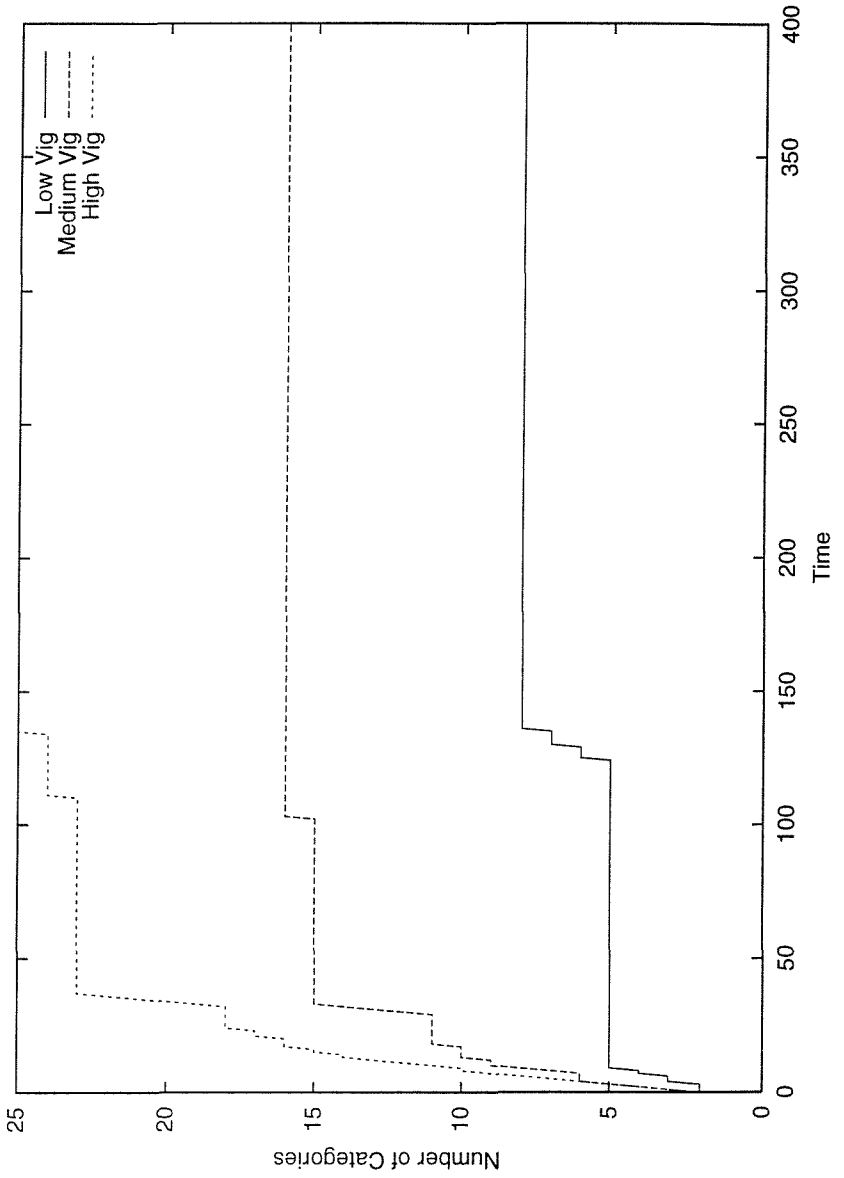

Figure 15: Low, medium and high vigilance parameter settings

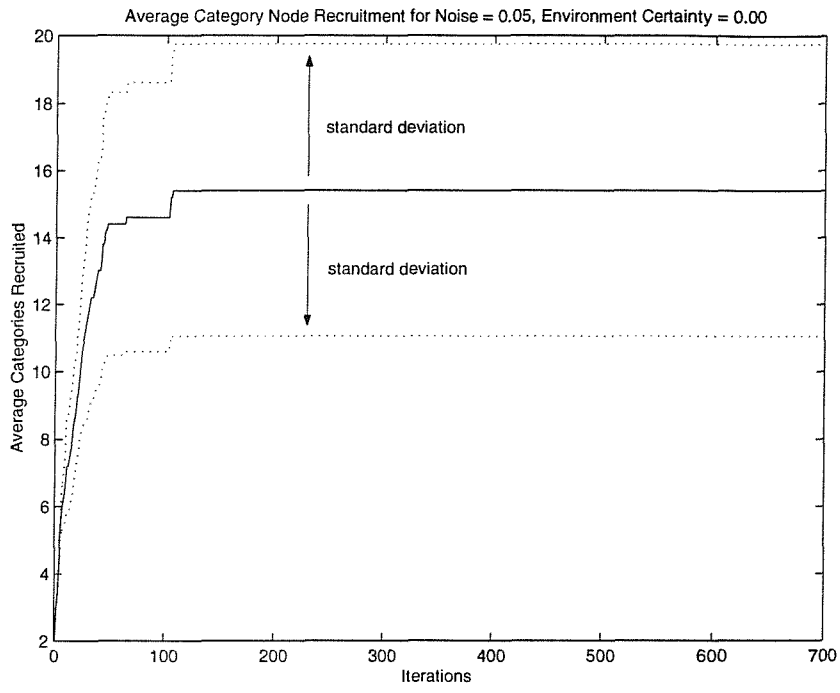

Figure 16: Noise constant at 0.05 and certainty held constant; averaged over 10 runs

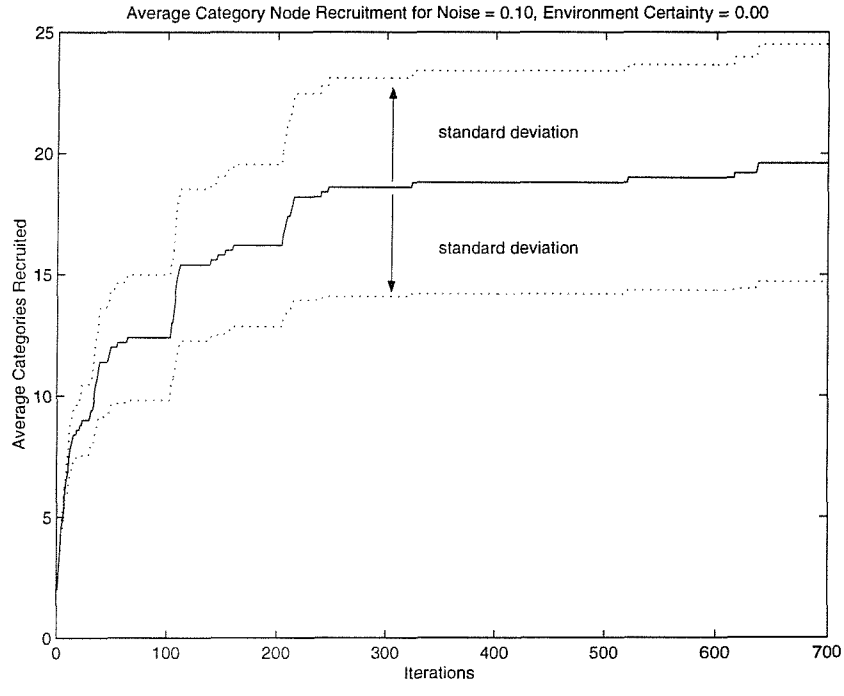

Figure 17: Noise constant at 0.10 and certainty held constant; averaged over 10 runs

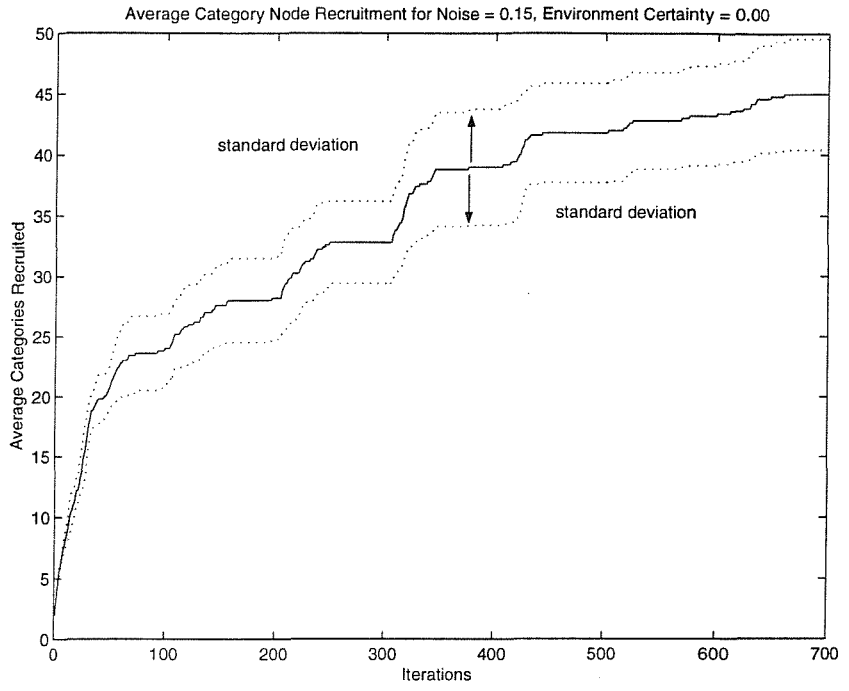

Figure 18: Noise constant at 0.15 and certainty held constant; averaged over 10 runs

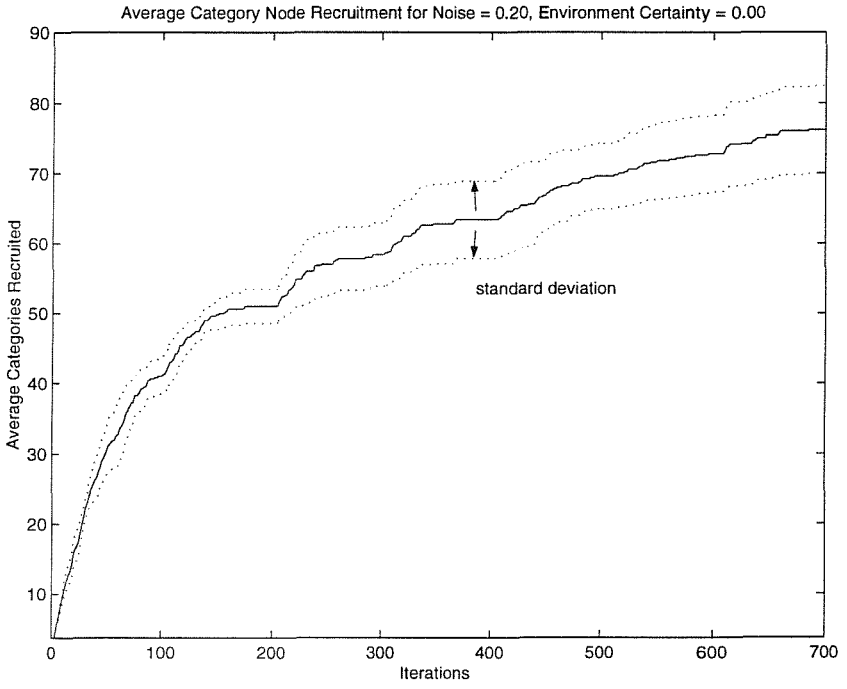

Figure 19: Noise constant at 0.20 and certainty held constant; averaged over 10 runs

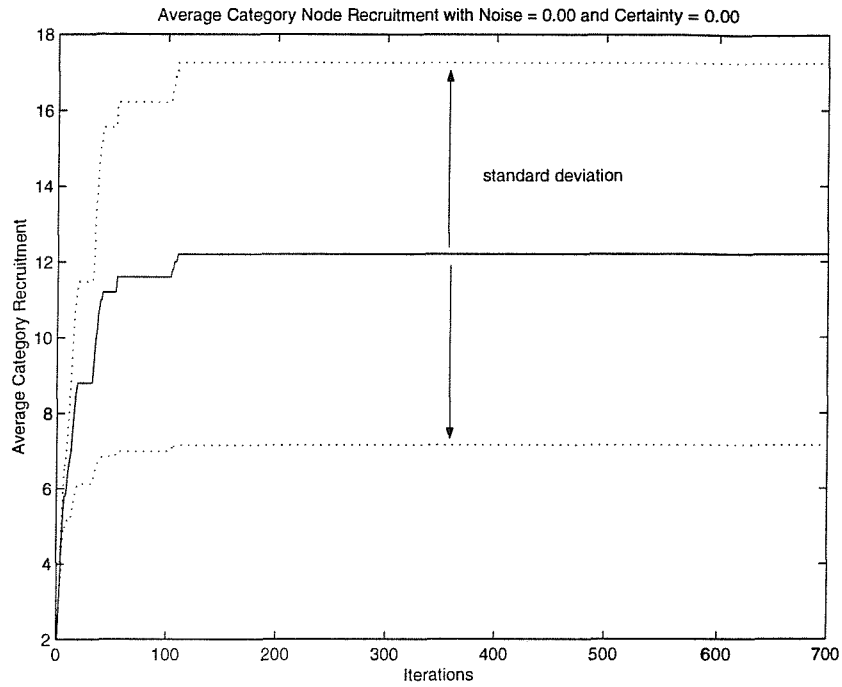

Figure 20: Certainty = 0.0 with No Sensor Noise; averaged over 10 runs

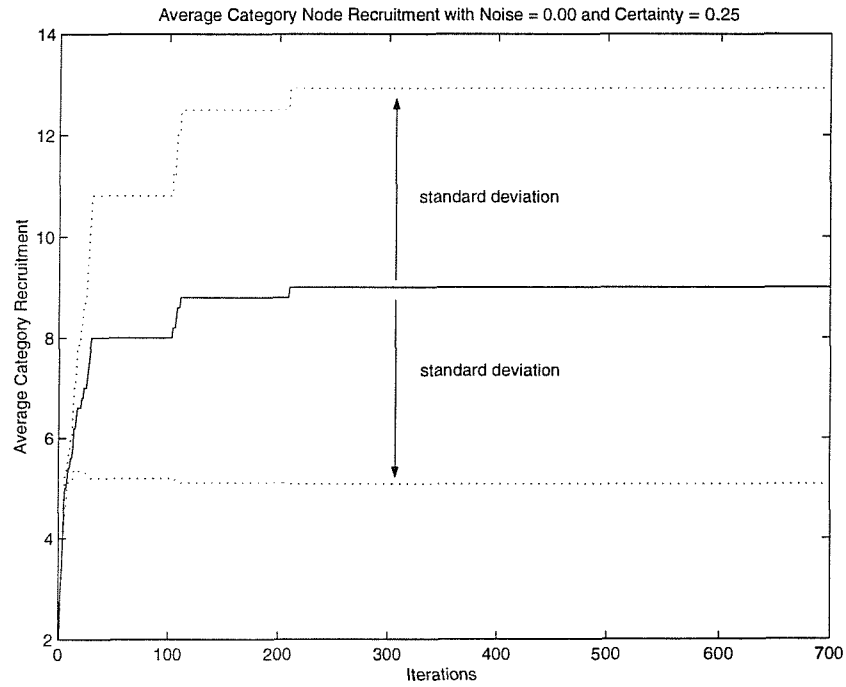

Figure 21: Certainty = 0.25 with No Sensor Noise; averaged over 10 runs

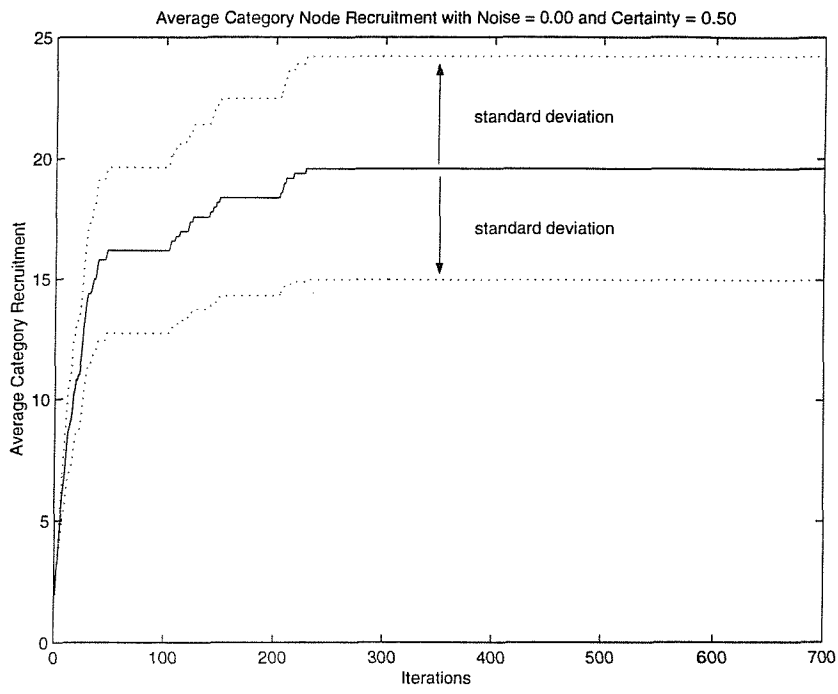

Figure 22: Certainty = 0.50 with No Sensor Noise; averaged over 10 runs

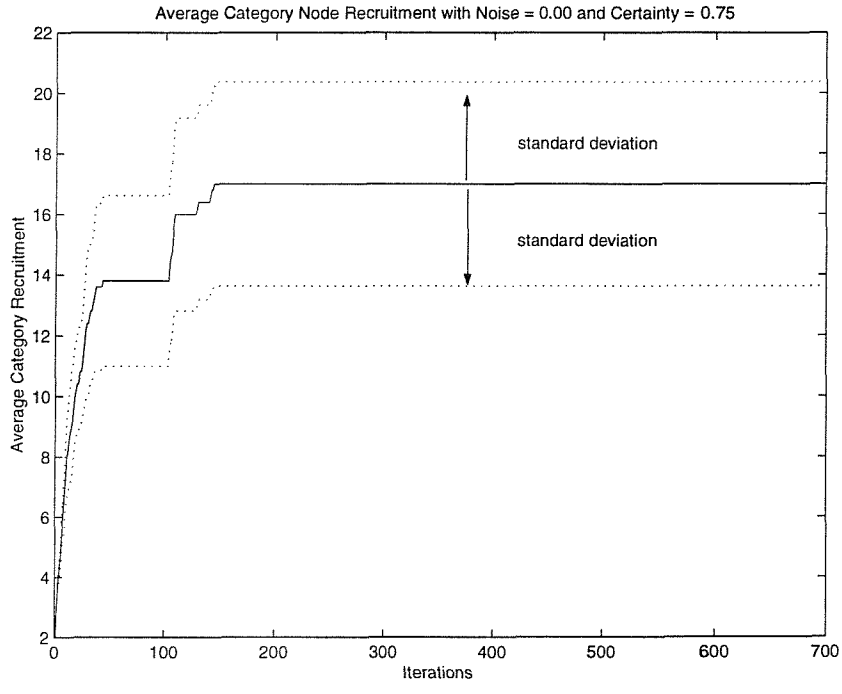

Figure 23: Certainty = 0.75 with No Sensor Noise; averaged over 10 runs

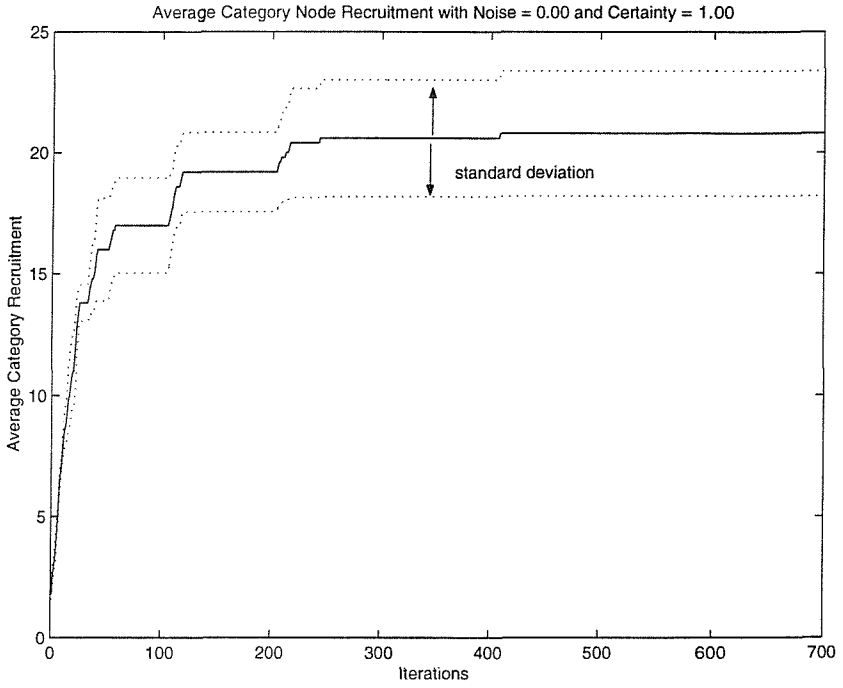

Figure 24: Certainty = 1.00 with No Sensor Noise; averaged over 10 runs

7 MLP and Local Network-based Agent: Simulation Results

7.1 Introduction

The local and MLP networks, with a minimal control architecture were tested. The full control architecture supports plasticity, novelty detection and the use of Q-learning paramters as analogs of modulatory mechanisms in naturally occuring adaptive neural systems. Here, the MLP and local networks were tested *with* an outcome-devaluation mechanism (which reduces reward when the agent is in a "satiated" state) to prevent simple, greedy strategies from emerging. However, no other improvements were made to the agent architecture to control learning and attention/vigilance.

It was decided that to evaluate the local and MLP networks with their various configurations, three experiments would be designed, and the agents tested in each environment with varying parameters. This helps establish an empirical basis for the metalearning theory (see (Doya, 1999) and (Joyce, 2001b) for details). The three experiment parameters are given in Table 1. The environments are characterised as 'easy', 'moderate' and 'hard', according to the perceived difficulty, or 'harshness', of the environment.

In each environment, the agent was given the goal parameters as follows:

- The passive decay rate for energy was $d_E = 0.1$
- The action costs for PRESS and RELEASE were $C_{PRESS} = C_{RELEASE} = 0.2$
- After initial experiments with the localised (RBF/ART-like) network, vigilance and basis-function radius were both set 0.2 (other values caused the network to behave unpredictably and the results were far worse than the MLP tests)
- After initial tests, the MLP proved to be most successful with 5 hidden units. This strongly suggests that the agent is 'assigning' hidden units to regions of the stimuli space in a more localised fashion. Other numbers of hidden units were tested (3 and 7) but the resulting performance was more erratic than with 5 units. That is, the gradual increase in performance expected after the start of the simulations was slower to be generated.
- The action-selection temperature was kept constant at 0.01.

The agents energy decays comparatively slowly, but redundant actions are quite expensive; recall, action costs are *subtracted* from the energy level.

To obtain trends, the parameter space for the reinforcement learning component was explored exhaustively. Each trial consisted of setting η and γ at varying levels and testing in the environment for 8000 iterations. The parameters were set at a value between 0.15 and 0.95, at increments of 0.10, resulting in 9 values for both η and γ which combined to give 81 combinations. Each combination of parameters was repeated five times (totalling 405 experiments), and the results presented are averages of these five trials.

Environment	Resources	Refresh	Danger	Neutral
Hard	25	50	5	15
Moderate	50	100	10	25
Easy	100	100	20	40

Table 1: Environments

During each trial, the mean drive level (arousal) and mean punishments over each cycle (the period of time between refreshes) was measured. This was to enable interpretation of the per-cycle performance. The regularity of environmental dynamics were expected to coerce the agent into producing a routine activity, and then progressively refining this over subsequent cycles. For each parameter combination, there are 5 'profiles', giving the per cycle performances which are averaged to obtain an overall mean per cycle performance for each parameter combination. While providing useful profiling data, a further summary is produced to enable an overall assessment of performance for further investigations. Taking each of the 81 averaged profiles (one for each parameter combination) the overall mean drive/arousal and punishments are derived to represent a 'lifetime performance' statistic. This then enables a parameter surface to be explored. The 'lifetime' summary data and tables can be found in sections 7.2, 8 and 9 below.

7.2 Easy Environent

The results from this environment are not particularly indicative of trends because the agent could spend a majority of its time with the button switched in the 'on' state and then only learn a real routine when the resources reach neutral and then danger levels. However, they provide some evidence that the agent is learning.

7.3 Local Network

Figure 25 and Figure 26 show the values of η and γ against the *overall* (that is, for the whole 8000 iterations) drive level (arousal) and punishments repsectively. This is taken as an indicative measure of lifetime performance and will be biased by early stages of the simulation when the agent attempts to familiarise itself with the environment. These results are averages of the 5 trials. Note that the contours are set at the lowest level achieved, where punishments should ideally be close to 0 as should drive/arousal level. Despite the punishment surface appearing irregular, the range of values over all the experiments was between $[0, -0.016]$.

Despite low variation in the punishments, the drive/arousal contours (Figure 25) reveal that regions of the parameter space are successful in achieving an overall performance, while others are not. To expose this fully, Figure 27 shows the η and γ combinations which maintained overall drive levels beneath 0.05 (exceptionally low, but see note above concerning the

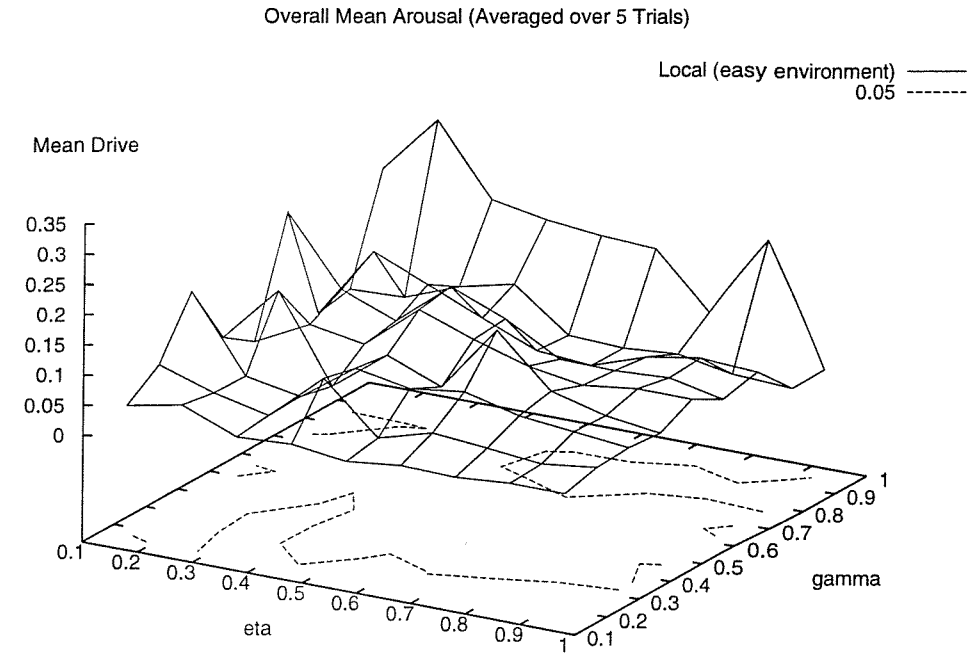

Figure 25: Local Net in Easy Environment: Overall Performance

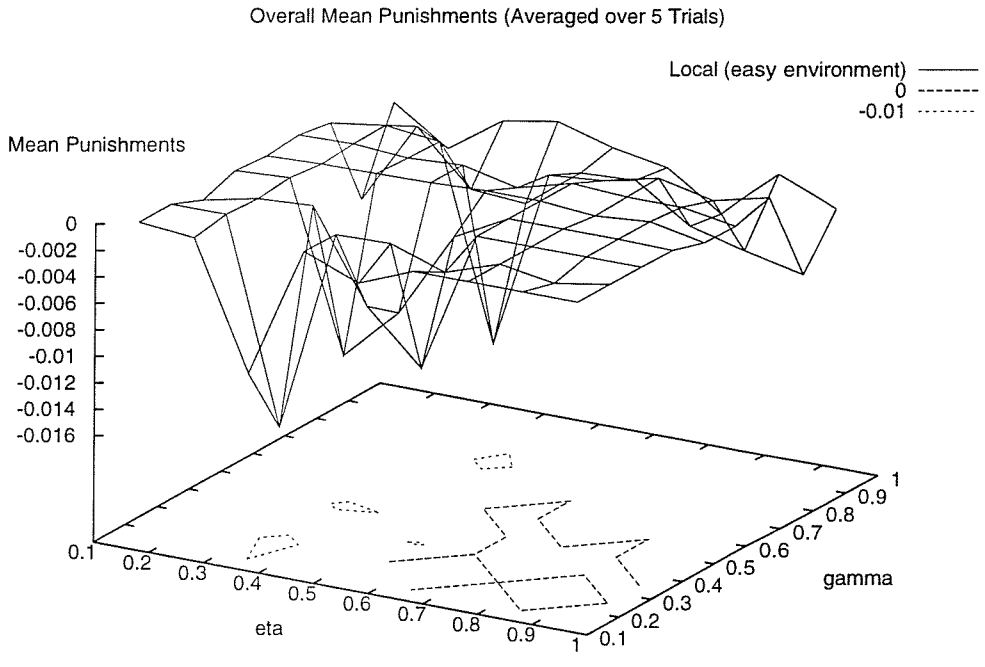

Figure 26: Local net in Easy Environment: Overall Punishments

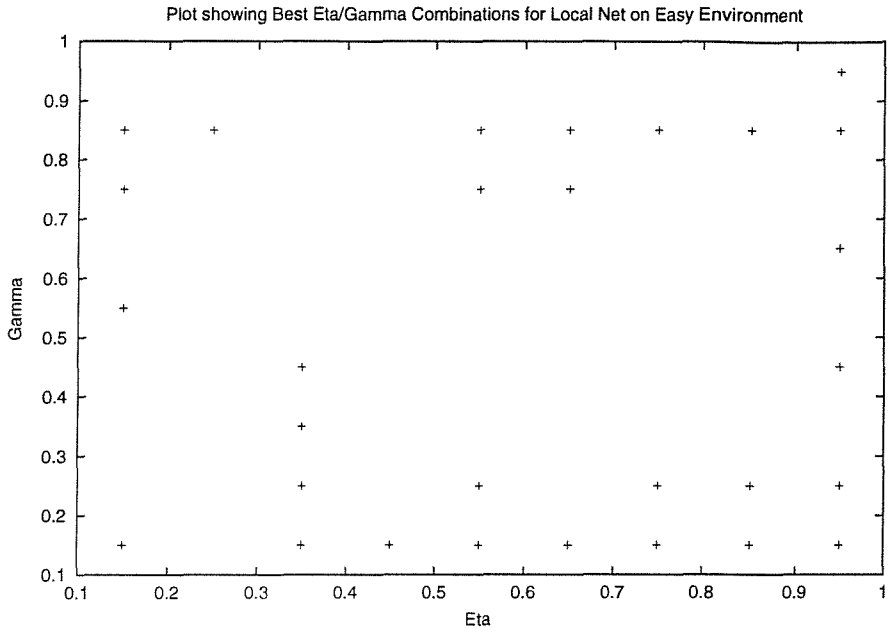

Figure 27: Local net in Easy Environment: Parameter Combinations Achieving Mean Arousal of < 0.05

easy environment). From this contour, the summary data was examined which revealed that when $\eta = 0.95$ and $\gamma = 0.95$ achieved the best result from the candidates which achieved mean arousal of 0.05 or less. In effect, almost any value from those indicated with a cross in Figure 27 would perform well.

More is revealed if the experiment averages are examined for $\eta = 0.95$ $\gamma = 0.95$. Figure 28 shows the mean drive/arousal and mean punishments over the 8000 iterations, averaged over 5 trials. Note that after initial periods of activity, the agent quickly establishes a 0 punishment, 0 arousal routine. However, this result is not surprising given that the agent is free to map the entire stimuli space or only regions which recurr. It is easy for the agent to find a few states which (by fortuitous exploration in early stages) lead to highly rewarding actions and contingencies, causing the agent to repeat actions which means its internal state become 'locked' into a small region of the stimuli space and the associated responses are constantly rewarded. The 5 trial average aims to prevent such activity, but evidently, with sufficiently high η and γ the agent is able to exploit such early regularity and never truly explore the stimuli space (e.g. for very low energies and corresponding drive values close to 1).

7.4 MLP Network

The results for the MLP agent were more defined. Figures 29 and 30 show the drive/arousal and punishments respectively. The descriptive statistics used are the same as for the local network.

Note that the contour for mean arousal/drive at 0.15 isolates only a few selected regions

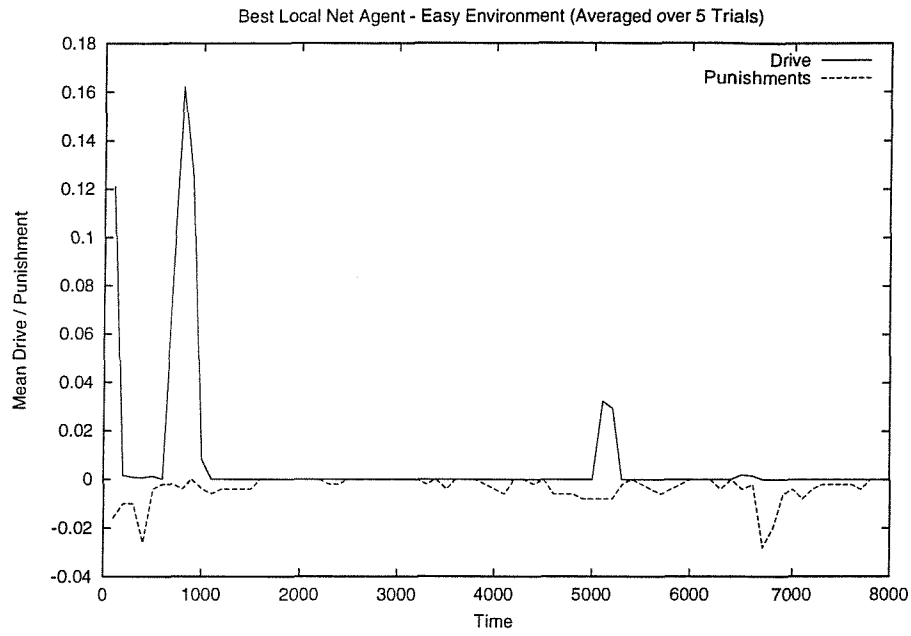

Figure 28: Best Local Net in Easy Environment: $\eta = 0.95$ $\gamma = 0.95$

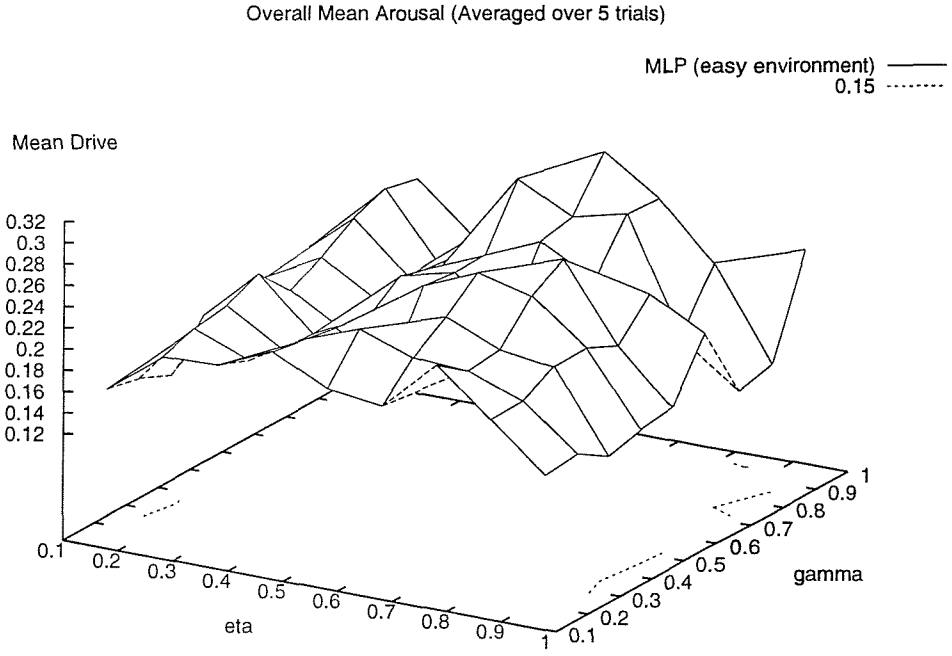

Figure 29: MLP in Easy Environment: Overall Performance

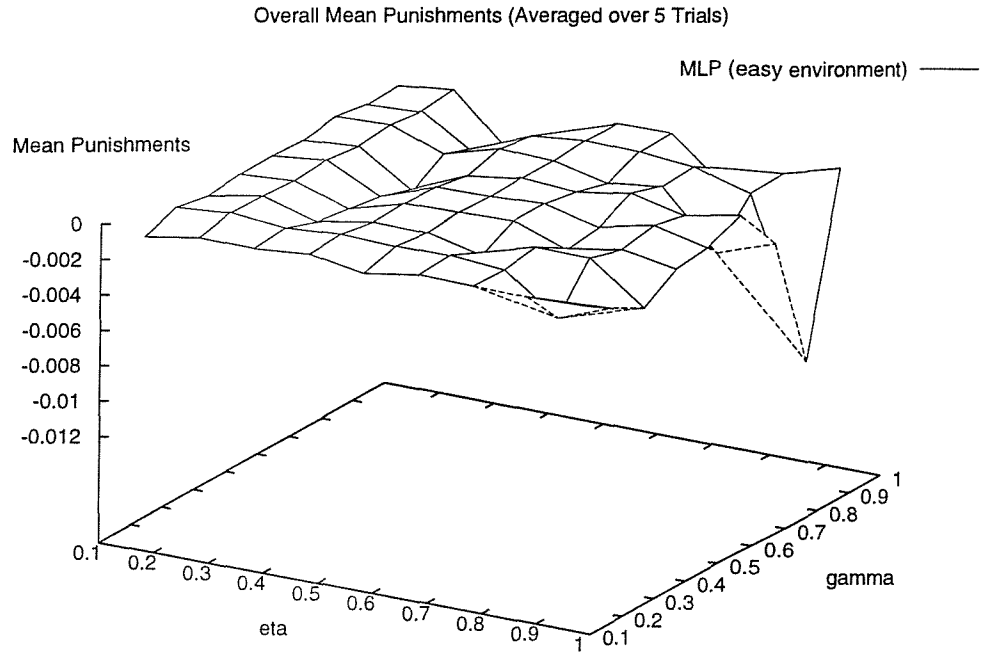

Figure 30: MLP in Easy Environment: Overall Punishments

of the parameter space, notably, high η and two regions of γ. The best performance occured at $\eta = 0.95$ and $\gamma = 0.75$ although almost all values of η resulted in mean arousal/drive of between $[0.12, 0.15]$. Interestingly, Figure 30 shows that the best (lowest) overall arousal/drive levels occurred when the mean punishments were worst, but the magnitude of these results suggests that this is unlikely to represent a significant trend because the punishment means are so close to zero.

Finally, the best agent is shown in Figure 31. Note that after initially using the punishments to learn a routine, the agent settles on a marginally higher rate of arousal and practically eliminates punishments.

8 Hard Environment

The difficulty of this environment is revealed by the results obtained. Firstly, the agent has only a short period of 'grace' when resources are delivered with a guarantee of no shock (punishment). However, if the agent settles into a greedy routine quickly, it is soon given punishments as the environment has so few resources and the neutral and danger levels are high relative to the number of rewards and the refresh rate. In addition, the agent must learn that rewards are refreshed infrequently (every 50 iterations) meaning that it must make use the NO-OP action if

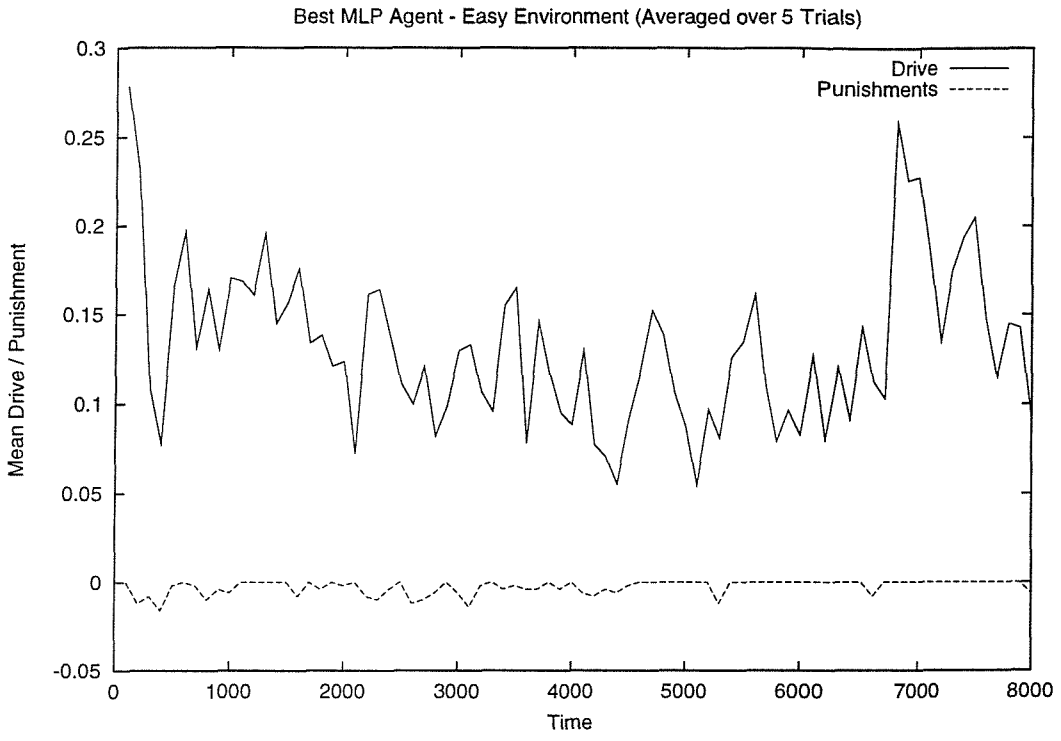

Figure 31: Best MLP in Easy Environment: $\eta = 0.95\ \gamma = 0.75$

it is to avoid repetitive punishment and not drain its internal energy quickly taking meaningless actions.

8.1 Local Network

Figures 32 (performance as drive minimisation) and 33 (related punishments) show the same data, respectively, as for Figures 25 and 26.

It is necessary to factor in the punishments when assessing the performance, because of the complexity of the contours shown on both diagrams. Note the regions of the parameter space separated by the mean arousal levels of 0.1 and 0.2 respectively. To simplify this, Figure 34 shows a planar representation of the contour, with crosses indicating the η and γ combinations which maintained a mean drive level of < 0.1. Note the apparent trade-off between high η and low γ. This suggests some kind of reciprocating relationship between these parameters in this kind of environment. Also, note the 'ridge' in Figure 32. Parameters which perform best for goal attainment (those in the < 0.1 contour region) fall in the region with worst punishment (see Figure 33). This suggests a necessary amount of punishment must occur to enable learning. To confirm this, the best agent's performance profile can be examined. Over the entire region of the parameter space for the contour showing < 0.1, the mean drive/arousal varied between only $[0.06, 0.08]$ suggesting that any combination of the parameters in Figure 34 would suffice.

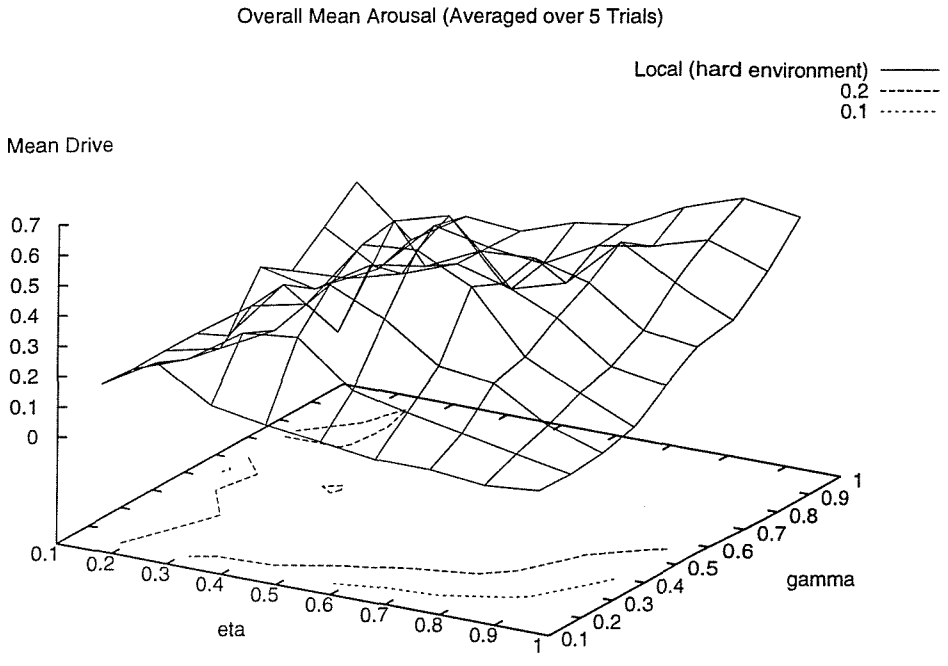

Figure 32: Local Net in Hard Environment: Overall Performance

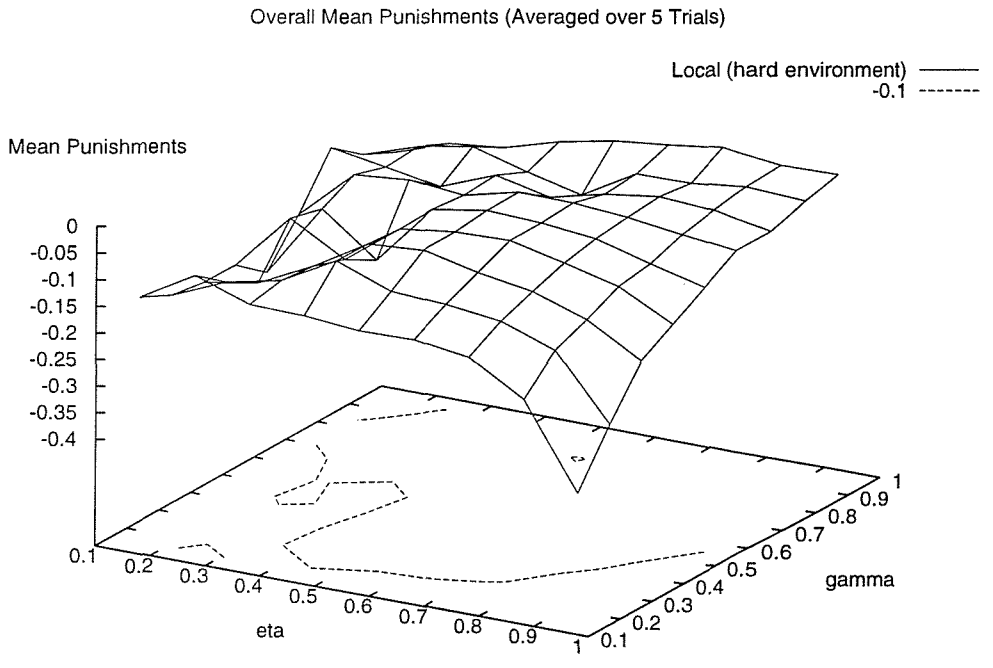

Figure 33: Local Net in Hard Environment: Overall Punishments

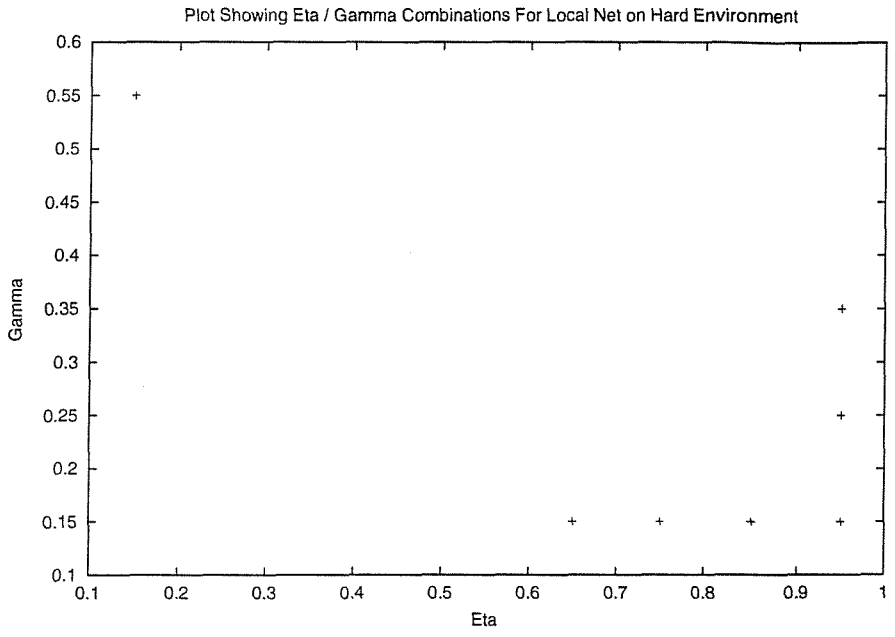

Figure 34: Local Net in Hard Environment: Parameter Combinations Achieving Mean Arousal of < 0.10

One note on the critical nature of the high η, low γ relationship; if for high η, $\gamma > 0.45$ then the performance of the agent's goal attainment degrades from around 0.08 to 0.5 for $\gamma = 0.95$ (the strong upwards slope at the right-hand edge of Figure 32).

Figure 35 shows the profile over time of the best local network agent in the hard environment with $\eta = 0.85$ and $\gamma = 0.15$. Note how initially (in early cycles of the environment) the agent learns the value of avoiding neutral / danger indications in the environment by trial and error. Eventually, it settles on a compromise of punishments and goal attainment. With punishments settled at around -0.2, this indicates that around 10 punishments are received each cycle in order that goal attainment is maintained.

8.2 MLP Network

Results for the MLP network are shown in Figures 36 and 37. They are quite easy to interpret; $\eta \geq 0.85$ and $\gamma \geq 0.75$ provide the best results (as evidenced by the contour 0.15 in Figure 36). The punishments similarly fit a decreasing trend as γ and η both approach 1. This contrasts with the local net, where there appeared to be a reciprocating relationship between γ and η (see Figure 34).

The best performance was found to be at $\eta = 0.95$ and $\gamma = 0.85$. This agent was examined and its 5 trial average profile is shown in Figure 38.

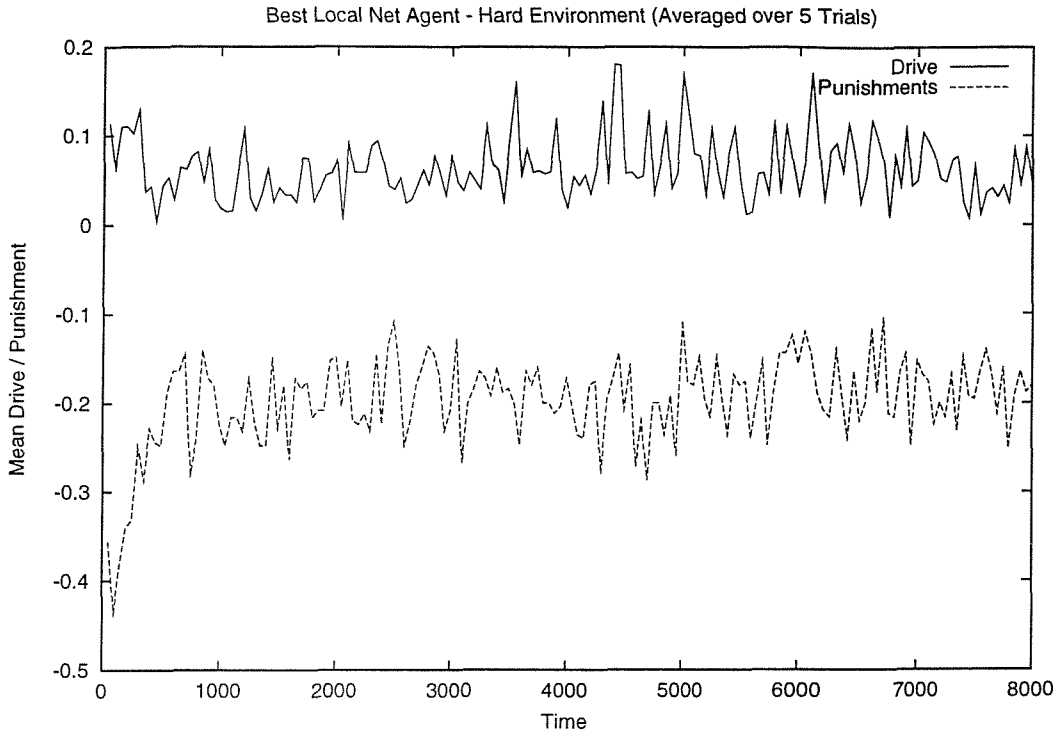

Figure 35: Best Local Net in Hard Environment: $\eta = 0.85$ $\gamma = 0.15$

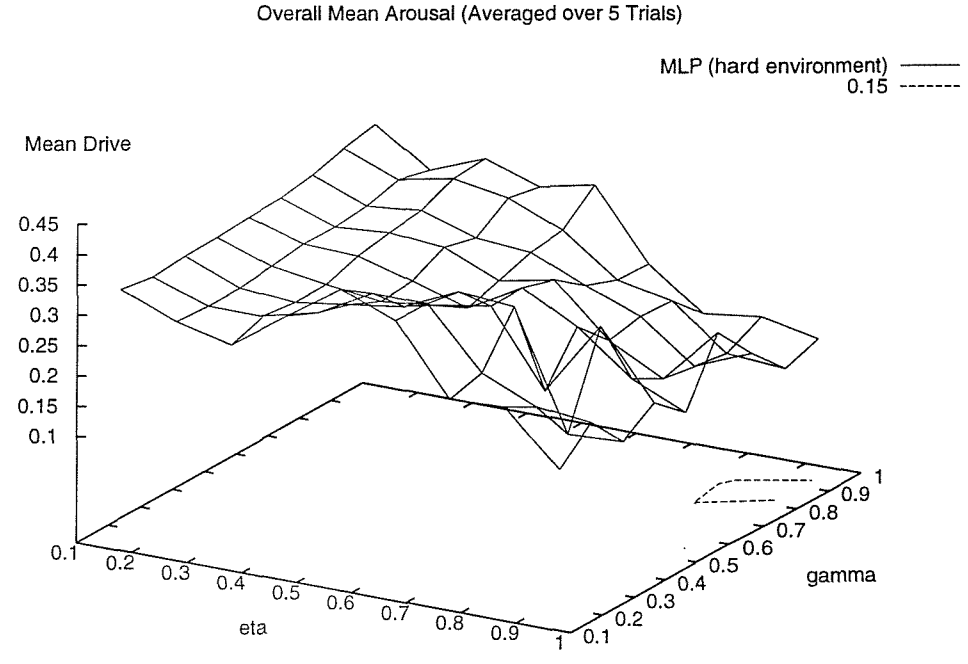

Figure 36: MLP Net in Hard Environment: Overall Performance

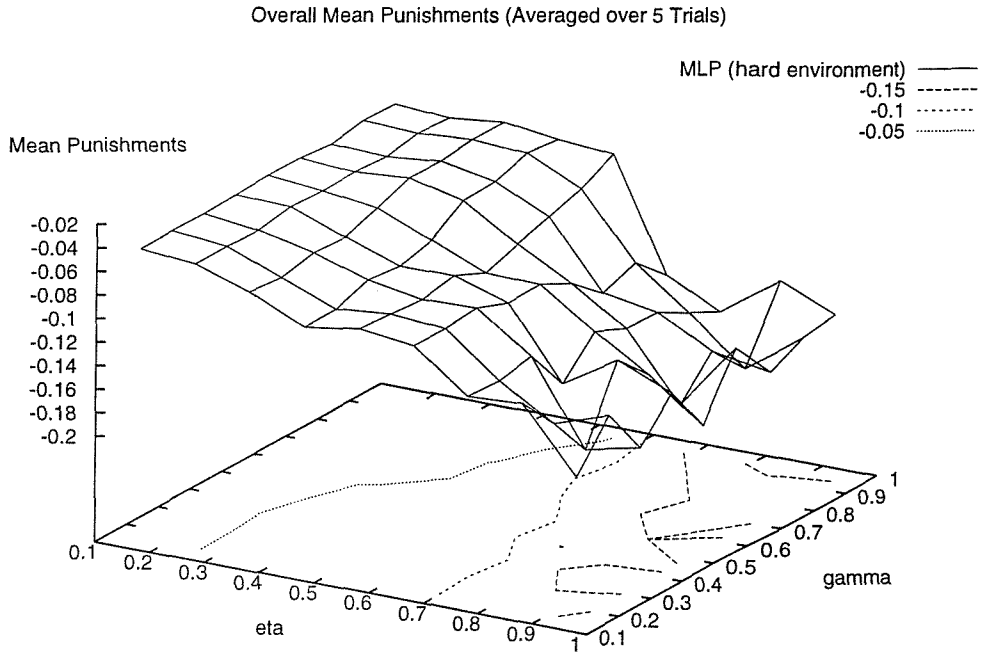

Figure 37: MLP Net in Hard Environment: Overall Punishments

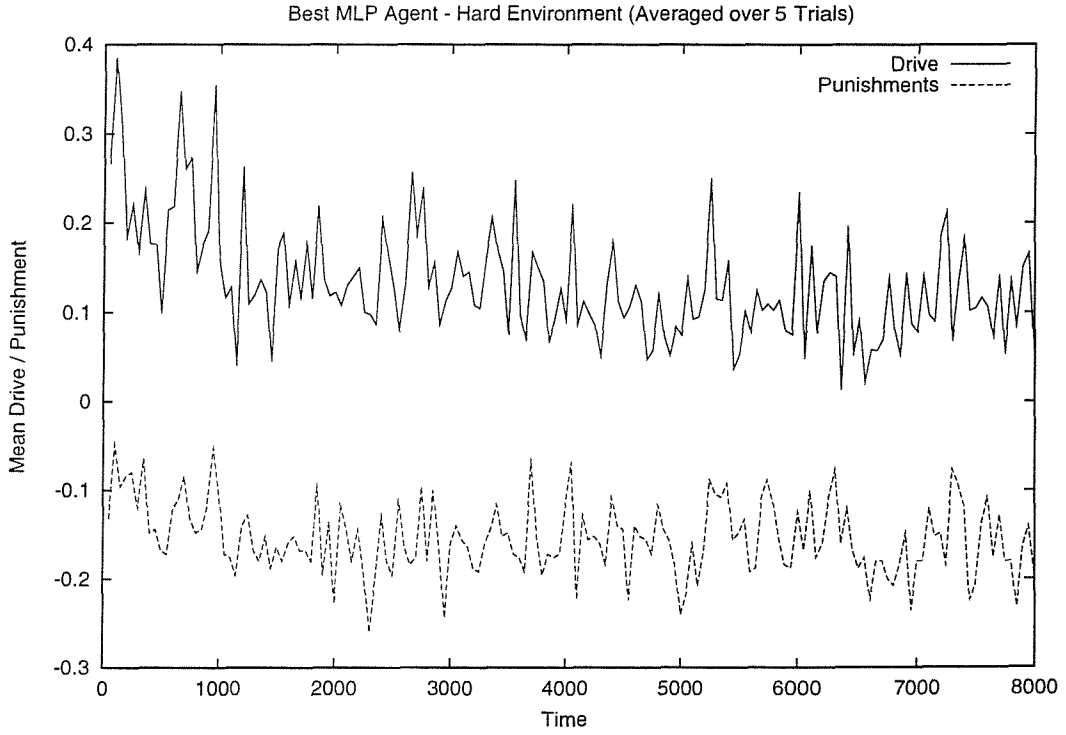

Figure 38: Best MLP Net in Hard Environment: $\eta = 0.95$ $\gamma = 0.85$

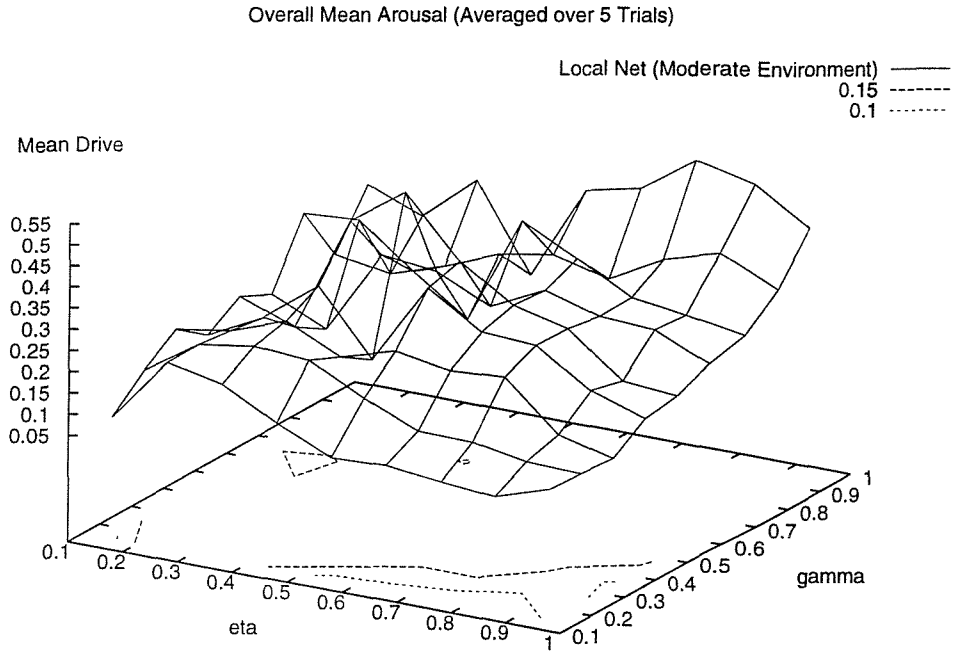

Figure 39: Local Net in Moderate Environment: Overall Performance

9 Moderate Environment

After testing the agents on two extreme environments (easy and hard) it was decided to look at the parameter space for a 'moderate' environmet; one deemed to be fair but not easy.

9.1 Local Network

Figures 39 and 40 show the parameter space in the usual way. It is interesting to note that the same trend as for the local network in the hard environment is displayed; the local network's best goal attainment is when the agent accepts punishments as part of the routine of maintaining its goal.

Examining the contours of Figures 39 and 40 also demonstrates that (unlike the MLP network) their is no clear winning combination of parameters. At best, there are ranges where the performance and punishments are similar.

9.2 MLP Network

As before, the overall lifetime performances are given in Figures 41 and 42. The best agent performance is easy to isolate; the contours in Figure 41 indicate strong trends in the performance / parameter space. A marginally lower drive is found for $\eta = 0.95$ and $\gamma = 0.75$. The

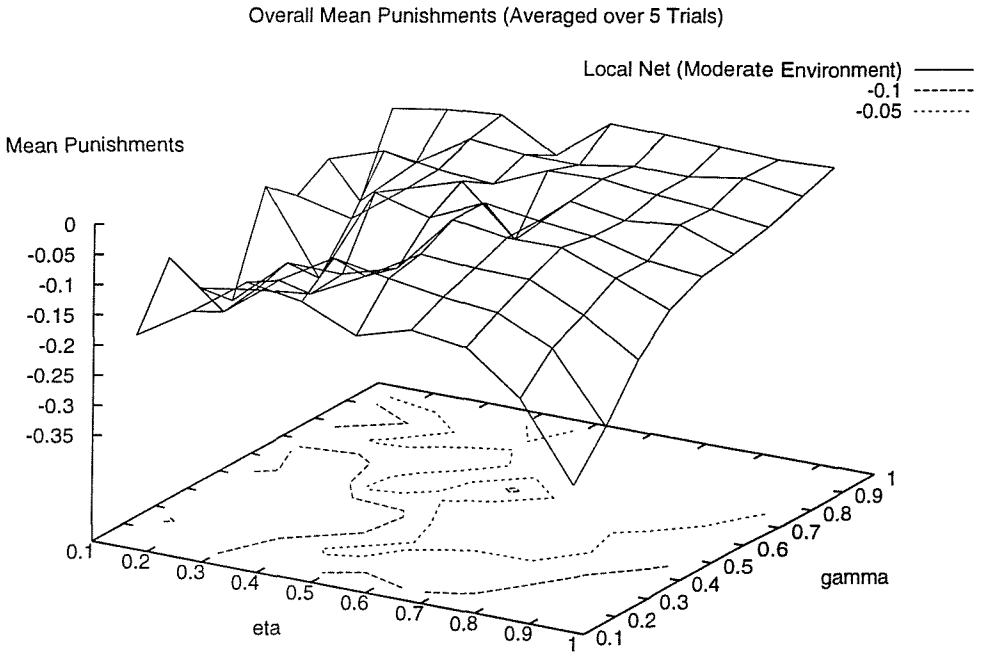

Figure 40: Local Net in Moderate Environment: Overall Punishments

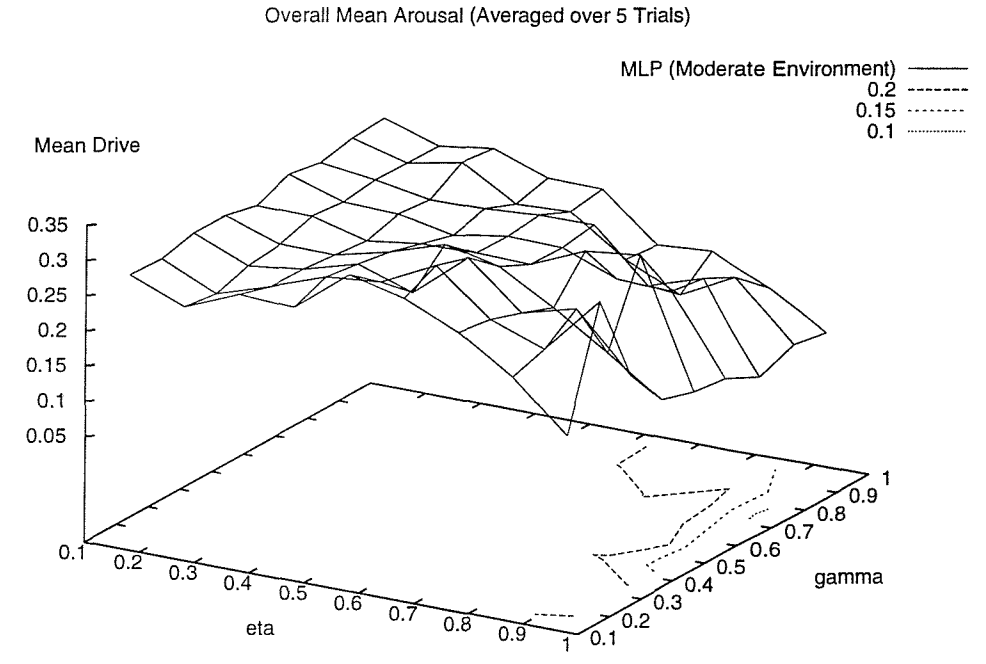

Figure 41: MLP Net in Moderate Environment: Overall Performance

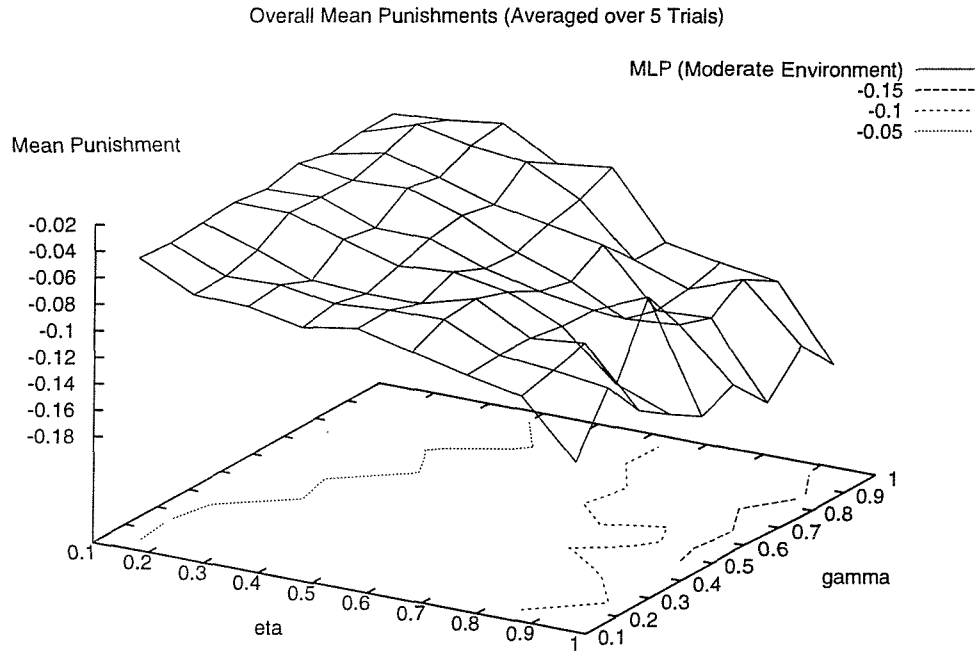

Figure 42: MLP Net in Moderate Environment: Overall Punishments

profile for this agent is shown in Figure 43. From this, the clear trend is similar for the hard environment. The agent initially has high drive/arousal and low punishments, but over time, trades-off punishments to reduce drive/arousal.

9.3 Trends and Experimental Conclusions

As stated, *no* parameter adaptation occured during each of the experiments; η and γ where set to one combination of the values stated above. The raw results, averaged over 5 runs, were analysed for the local network using the Hebbian interpretation of the Q-learning rule, and the MLP using the Bellman residual as the target vector. From the analyses, it was possible to establish the following empirical trends:

- For the MLP networks, the best performance was always with high η, suggesting the agents need to constantly adapt the responses to maintain goals. However, the role of γ seems to be that in very hard environments, this value should be higher than for moderate and easy environments. In latter cases, the variability of γ is high for easy environments and was more localised in the higher range (towards 1) for the moderate environment.

- The local networks marginally outperform the MLP networks, but this is most likely due to the closer-to-exhaustive mapping of the stimuli space, enabling greater flexibility in

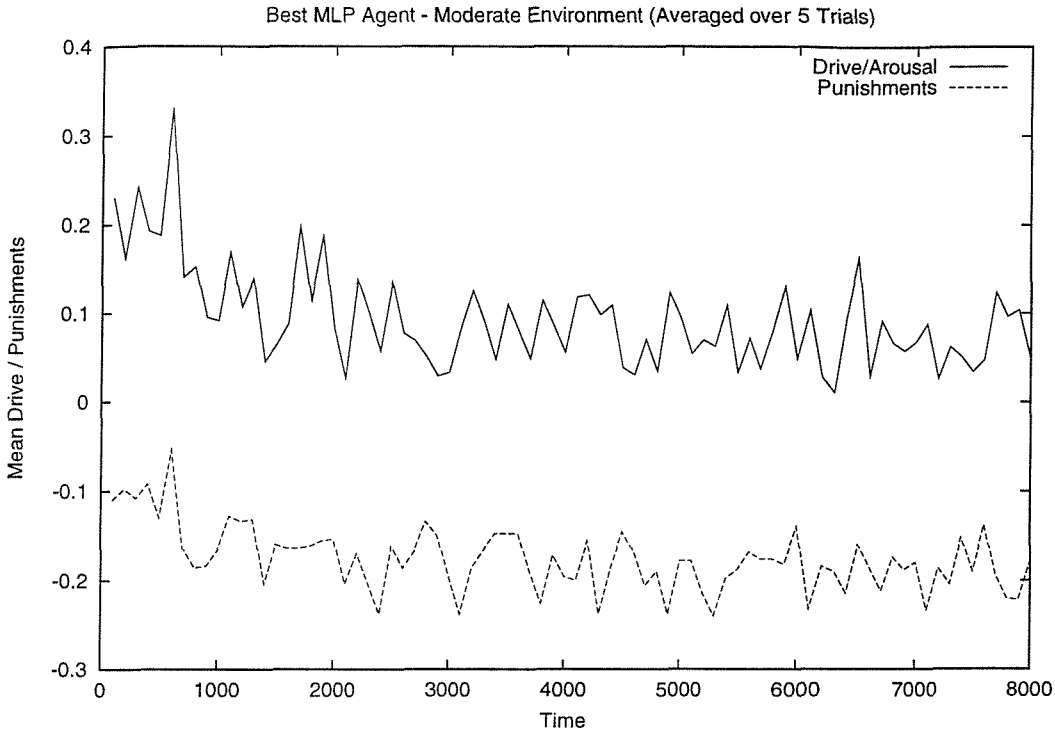

Figure 43: Best MLP Net in Medium Environment: $\eta = 0.95$ $\gamma = 0.75$

constructing appropriate maps from stimuli to desirable actions/outcomes.

- the reliance on high η suggests that *all* networks are constantly adjusting (and needing to adjust) the response associated to internal state. While not unsurprising, this suggests that a more appropriate method might be to encourage exploration while reducing learning rates (both η and γ in situations which are 'no win' cf. the hard environment and when the agent reaches high arousal/drive in the absence of rewards without accompanying punishments (e.g. when the environment is in the 'danger' state). This is in part supported by the small regions (isolated by contours) of low η and γ in the local networks – see Figures 25, 32 and 39.

- For the local networks, an important trend (opposite to the MLPs) is that the balance between γ and η appears more critical. In Figures 29, 36 and 41 for MLPs, it is almost always the case that the contours suggest that both parameters should be high. The contours are relatively tidy and suggest a strong decreasing performance gradient toward the high η, high γ state. However, for local networks, the higher values of γ affect the punishments for approximately similar overall goal attainment (e.g. the overall drive/arousal is similar). This is especially true for the moderate environment (see Figures 39 and 40) where punishment clearly trades off against performance as η and γ vary. This suggests

that future rewards balanced against learning the current outcome are critical and that too high γ skews this considerably. For example, in Figure 39 the mean drive/arousal level varies from around 0.05 to around 0.5 for high constant η and variable γ.

While these results are not intended to be conclusive, they aided in deciding how to proceed in building a control architecture for agents. The above empirical conclusions are carried forward into this design process, because the theoretical proposition of (Doya, 1999) remains under-constrained.

For MLPs the trend appears to be to set both parameters high. This suggests that the experience of novel situations is overwriting existing, learned patterns. Whereas for the local network, the relationship between η and γ is more complex, suggesting a requirement to balance the two parameters. The purpose of testing was to see if there is any empirical support for Doya's proposal that it might be possible to better understand learning, in situated agents, via cognitive and neuro-modulatory mechanisms.

There was also little support for an overall superior technique – the MLP and the local network perform in different ways (e.g. in how they partition the input space) and while the local network might be marginally superior, engineering constraints will most likely decide on a choice of network design. In further work (to build a more robust architecture for agents which included parts of Doya's meta-learning theory) the local network was chosen, more because of its transparency and separation from perceptual machinery.

10 Simulation Experimental Data

10.1 Introduction

For completeness, this section reproduces the summary lifetime performances for the agent experiments with MLP and local networks. Some of the conclusions drawn from the graphs of section 7 are easier to justfiy with reference to the actual tabular data.

10.2 Easy Environment (Local Net)

Eta	Gamma	Mean Drive	Mean Punishments
0.15	0.15	0.043345	-0.000150
0.15	0.25	0.081728	-0.000125
0.15	0.35	0.173435	-0.001150
0.15	0.45	0.068620	-0.000225
0.15	0.55	0.032470	-0.000450
0.15	0.65	0.217761	-0.000200
0.15	0.75	0.020986	-0.000625
0.15	0.85	0.031936	-0.007750
0.15	0.95	0.202670	-0.001750

Eta	Gamma	Mean Drive	Mean Punishments
0.25	0.15	0.061050	-0.000550
0.25	0.25	0.052256	-0.000100
0.25	0.35	0.050965	-0.000250
0.25	0.45	0.163204	0.000000
0.25	0.55	0.077429	-0.000275
0.25	0.65	0.106325	-0.000100
0.25	0.75	0.140557	-0.000050
0.25	0.85	0.035208	-0.001450
0.25	0.95	0.300342	-0.004450

Eta	Gamma	Mean Drive	Mean Punishments
0.35	0.15	0.025131	-0.010025
0.35	0.25	0.031660	-0.015425
0.35	0.35	0.033828	0.000000
0.35	0.45	0.021781	-0.012675
0.35	0.55	0.061790	-0.000250
0.35	0.65	0.071164	-0.000400
0.35	0.75	0.102405	-0.000500
0.35	0.85	0.069725	-0.005450
0.35	0.95	0.185060	-0.001575

Eta	Gamma	Mean Drive	Mean Punishments
0.45	0.15	0.029943	-0.000025
0.45	0.25	0.109850	-0.000100
0.45	0.35	0.097126	-0.006850
0.45	0.45	0.089576	-0.008675
0.45	0.55	0.135960	-0.000175
0.45	0.65	0.144864	-0.000150
0.45	0.75	0.063800	-0.015000
0.45	0.85	0.091169	-0.005725
0.45	0.95	0.168791	-0.000850

Eta	Gamma	Mean Drive	Mean Punishments
0.55	0.15	0.018711	-0.001675
0.55	0.25	0.027655	0.000000
0.55	0.35	0.080444	-0.010750
0.55	0.45	0.054039	-0.002175
0.55	0.55	0.103462	0.000000
0.55	0.65	0.108725	-0.001100
0.55	0.75	0.027015	-0.001475
0.55	0.85	0.023462	-0.002625
0.55	0.95	0.159676	-0.002050

Eta	Gamma	Mean Drive	Mean Punishments
0.65	0.15	0.028873	0.000000
0.65	0.25	0.053022	-0.001400
0.65	0.35	0.092280	0.000000
0.65	0.45	0.164895	0.000000
0.65	0.55	0.076475	0.000000
0.65	0.65	0.058994	0.000000
0.65	0.75	0.019809	-0.001125
0.65	0.85	0.019344	-0.002425
0.65	0.95	0.155142	-0.002775

Eta	Gamma	Mean Drive	Mean Punishments
0.75	0.15	0.026574	0.000000
0.75	0.25	0.045085	0.000000
0.75	0.35	0.069560	0.000000
0.75	0.45	0.081375	0.000000
0.75	0.55	0.062143	-0.000350
0.75	0.65	0.057457	-0.000250
0.75	0.75	0.050849	-0.000150
0.75	0.85	0.026810	-0.005100
0.75	0.95	0.066591	-0.005525

Eta	Gamma	Mean Drive	Mean Punishments
0.85	0.15	0.033910	0.000000
0.85	0.25	0.036582	-0.000675
0.85	0.35	0.060265	0.000000
0.85	0.45	0.057650	0.000000
0.85	0.55	0.073652	0.000000
0.85	0.65	0.062415	-0.000250
0.85	0.75	0.066116	-0.001075
0.85	0.85	0.010885	-0.006175
0.85	0.95	0.203861	-0.001725

Eta	Gamma	Mean Drive	Mean Punishments
0.95	0.15	0.033713	0.000000
0.95	0.25	0.047744	0.000000
0.95	0.35	0.050804	0.000000
0.95	0.45	0.045821	-0.000050
0.95	0.55	0.073099	-0.000825
0.95	0.65	0.044039	-0.000900
0.95	0.75	0.060502	-0.000075
0.95	0.85	0.005230	-0.007200
0.95	0.95	0.007141	-0.003575

10.3 Easy Environment (MLP Net)

Eta	Gamma	Mean Drive	Mean Punishments
0.15	0.15	0.158375	-0.000975
0.15	0.25	0.152290	-0.000275
0.15	0.35	0.138027	-0.000625
0.15	0.45	0.177715	-0.000400
0.15	0.55	0.151051	-0.000350
0.15	0.65	0.181574	-0.000475
0.15	0.75	0.184614	-0.000125
0.15	0.85	0.184643	-0.000450
0.15	0.95	0.162045	-0.000375

Eta	Gamma	Mean Drive	Mean Punishments
0.25	0.15	0.198665	-0.000425
0.25	0.25	0.208230	0.000000
0.25	0.35	0.213857	-0.000050
0.25	0.45	0.227172	-0.000200
0.25	0.55	0.195080	-0.000025
0.25	0.65	0.209078	-0.000225
0.25	0.75	0.229555	0.000000
0.25	0.85	0.240644	-0.000250
0.25	0.95	0.233645	-0.000025

Eta	Gamma	Mean Drive	Mean Punishments
0.35	0.15	0.200151	-0.000475
0.35	0.25	0.188228	-0.000300
0.35	0.35	0.183003	-0.000950
0.35	0.45	0.179454	-0.001225
0.35	0.55	0.165865	-0.001500
0.35	0.65	0.178697	-0.002200
0.35	0.75	0.184085	-0.001100
0.35	0.85	0.176942	-0.002175
0.35	0.95	0.189885	-0.002850

Eta	Gamma	Mean Drive	Mean Punishments
0.45	0.15	0.222557	-0.000200
0.45	0.25	0.217394	-0.000100
0.45	0.35	0.211587	-0.000400
0.45	0.45	0.205249	-0.000325
0.45	0.55	0.227542	-0.000425
0.45	0.65	0.207384	-0.000775
0.45	0.75	0.214403	-0.000225
0.45	0.85	0.227366	-0.000500
0.45	0.95	0.195229	-0.002100

Eta	Gamma	Mean Drive	Mean Punishments
0.55	0.15	0.197308	-0.000725
0.55	0.25	0.236928	-0.000175
0.55	0.35	0.184709	-0.000425
0.55	0.45	0.243865	-0.000250
0.55	0.55	0.240301	-0.000350
0.55	0.65	0.239774	-0.000325
0.55	0.75	0.295744	-0.000325
0.55	0.85	0.218035	-0.000175
0.55	0.95	0.226636	-0.000225

Eta	Gamma	Mean Drive	Mean Punishments
0.65	0.15	0.190710	-0.000225
0.65	0.25	0.219463	-0.000125
0.65	0.35	0.241383	-0.001125
0.65	0.45	0.266578	-0.000175
0.65	0.55	0.200843	-0.000200
0.65	0.65	0.262327	-0.000525
0.65	0.75	0.268547	-0.001025
0.65	0.85	0.314075	-0.000250
0.65	0.95	0.247319	-0.000175

Eta	Gamma	Mean Drive	Mean Punishments
0.75	0.15	0.238574	-0.000275
0.75	0.25	0.216365	-0.000350
0.75	0.35	0.219717	-0.000100
0.75	0.45	0.254460	-0.001225
0.75	0.55	0.271601	-0.000950
0.75	0.65	0.235853	-0.000025
0.75	0.75	0.281634	-0.000625
0.75	0.85	0.279211	-0.000500
0.75	0.95	0.172323	-0.002625

Eta	Gamma	Mean Drive	Mean Punishments
0.85	0.15	0.197073	-0.000325
0.85	0.25	0.198303	-0.002525
0.85	0.35	0.213210	-0.000100
0.85	0.45	0.214114	-0.000650
0.85	0.55	0.199523	-0.000275
0.85	0.65	0.232294	-0.000700
0.85	0.75	0.175173	-0.003850
0.85	0.85	0.227885	-0.001425
0.85	0.95	0.143445	-0.001250

Eta	Gamma	Mean Drive	Mean Punishments
0.95	0.15	0.153758	-0.000225
0.95	0.25	0.157511	-0.001400
0.95	0.35	0.138634	-0.002400
0.95	0.45	0.147048	-0.001150
0.95	0.55	0.151372	-0.000925
0.95	0.65	0.204587	-0.000125
0.95	0.75	0.132562	-0.002700
0.95	0.85	0.141980	-0.010350
0.95	0.95	0.232562	-0.000375

10.4 Hard Environment (Local Net)

Eta	Gamma	Mean Drive	Mean Punishments
0.15	0.15	0.161510	-0.140200
0.15	0.25	0.151970	-0.170400
0.15	0.35	0.155088	-0.179400
0.15	0.45	0.154029	-0.180075
0.15	0.55	0.088825	-0.226900
0.15	0.65	0.257579	-0.150875
0.15	0.75	0.187156	-0.058925
0.15	0.85	0.272144	-0.105050
0.15	0.95	0.362453	-0.134200

Eta	Gamma	Mean Drive	Mean Punishments
0.25	0.15	0.265949	-0.080525
0.25	0.25	0.219324	-0.125800
0.25	0.35	0.196530	-0.160650
0.25	0.45	0.279355	-0.072925
0.25	0.55	0.291605	-0.088175
0.25	0.65	0.218666	-0.057075
0.25	0.75	0.195761	-0.075950
0.25	0.85	0.168852	-0.070875
0.25	0.95	0.203492	-0.099200

Eta	Gamma	Mean Drive	Mean Punishments
0.35	0.15	0.159890	-0.115275
0.35	0.25	0.339826	-0.103350
0.35	0.35	0.289410	-0.109750
0.35	0.45	0.318753	-0.130450
0.35	0.55	0.168042	-0.165275
0.35	0.65	0.331277	-0.047375
0.35	0.75	0.242328	-0.094475
0.35	0.85	0.342433	-0.052525
0.35	0.95	0.314714	-0.087475

Eta	Gamma	Mean Drive	Mean Punishments
0.45	0.15	0.128762	-0.117075
0.45	0.25	0.359380	-0.047050
0.45	0.35	0.474347	-0.048775
0.45	0.45	0.548682	-0.054000
0.45	0.55	0.565994	-0.051525
0.45	0.65	0.359310	-0.058025
0.45	0.75	0.309597	-0.053500
0.45	0.85	0.306276	-0.080575
0.45	0.95	0.300856	-0.057000

Eta	Gamma	Mean Drive	Mean Punishments
0.55	0.15	0.106719	-0.126300
0.55	0.25	0.217977	-0.084850
0.55	0.35	0.393536	-0.041700
0.55	0.45	0.562757	-0.041350
0.55	0.55	0.618048	-0.034150
0.55	0.65	0.443583	-0.032725
0.55	0.75	0.260920	-0.075500
0.55	0.85	0.286496	-0.104400
0.55	0.95	0.362973	-0.036000

Eta	Gamma	Mean Drive	Mean Punishments
0.65	0.15	0.081216	-0.123725
0.65	0.25	0.177355	-0.091125
0.65	0.35	0.272050	-0.064575
0.65	0.45	0.476863	-0.037875
0.65	0.55	0.428766	-0.038200
0.65	0.65	0.455420	-0.033250
0.65	0.75	0.313659	-0.050075
0.65	0.85	0.379610	-0.045150
0.65	0.95	0.389919	-0.023925

Eta	Gamma	Mean Drive	Mean Punishments
0.75	0.15	0.080857	-0.138425
0.75	0.25	0.143230	-0.103375
0.75	0.35	0.250022	-0.080650
0.75	0.45	0.278869	-0.060325
0.75	0.55	0.350015	-0.048300
0.75	0.65	0.408010	-0.041325
0.75	0.75	0.483988	-0.035225
0.75	0.85	0.412216	-0.043525
0.75	0.95	0.479643	-0.010125

Eta	Gamma	Mean Drive	Mean Punishments
0.85	0.15	0.063171	-0.197950
0.85	0.25	0.109473	-0.138650
0.85	0.35	0.145657	-0.114150
0.85	0.45	0.194262	-0.084800
0.85	0.55	0.223698	-0.075400
0.85	0.65	0.284739	-0.054625
0.85	0.75	0.383637	-0.041925
0.85	0.85	0.464653	-0.031850
0.85	0.95	0.546246	-0.024325

Eta	Gamma	Mean Drive	Mean Punishments
0.95	0.15	0.080386	-0.354750
0.95	0.25	0.076439	-0.258400
0.95	0.35	0.088766	-0.172225
0.95	0.45	0.116121	-0.136800
0.95	0.55	0.194538	-0.100525
0.95	0.65	0.263079	-0.065800
0.95	0.75	0.293380	-0.063075
0.95	0.85	0.399257	-0.038900
0.95	0.95	0.519793	-0.022950

10.5 Hard Environment (MLP Net)

Eta	Gamma	Mean Drive	Mean Punishments
0.15	0.15	0.335268	-0.043925
0.15	0.25	0.327023	-0.043825
0.15	0.35	0.332342	-0.044000
0.15	0.45	0.334091	-0.042875
0.15	0.55	0.337581	-0.041550
0.15	0.65	0.339407	-0.041825
0.15	0.75	0.347705	-0.044975
0.15	0.85	0.362180	-0.039900
0.15	0.95	0.372919	-0.040775

Eta	Gamma	Mean Drive	Mean Punishments
0.25	0.15	0.298925	-0.048075
0.25	0.25	0.295573	-0.047950
0.25	0.35	0.296460	-0.049500
0.25	0.45	0.302792	-0.044475
0.25	0.55	0.310358	-0.045775
0.25	0.65	0.309038	-0.045075
0.25	0.75	0.313997	-0.047025
0.25	0.85	0.327825	-0.041950
0.25	0.95	0.313743	-0.044200

Eta	Gamma	Mean Drive	Mean Punishments
0.35	0.15	0.277755	-0.062900
0.35	0.25	0.295470	-0.065850
0.35	0.35	0.288781	-0.059600
0.35	0.45	0.289258	-0.061675
0.35	0.55	0.307250	-0.051900
0.35	0.65	0.316495	-0.041325
0.35	0.75	0.323235	-0.042350
0.35	0.85	0.345739	-0.038800
0.35	0.95	0.349719	-0.038850

Eta	Gamma	Mean Drive	Mean Punishments
0.45	0.15	0.341336	-0.084400
0.45	0.25	0.317831	-0.083825
0.45	0.35	0.302112	-0.074150
0.45	0.45	0.278233	-0.077350
0.45	0.55	0.282394	-0.070625
0.45	0.65	0.308644	-0.052050
0.45	0.75	0.295103	-0.050675
0.45	0.85	0.331314	-0.041800
0.45	0.95	0.320374	-0.045025

Eta	Gamma	Mean Drive	Mean Punishments
0.55	0.15	0.402497	-0.077225
0.55	0.25	0.366873	-0.079550
0.55	0.35	0.315385	-0.082000
0.55	0.45	0.287943	-0.076875
0.55	0.55	0.254410	-0.089400
0.55	0.65	0.270736	-0.080150
0.55	0.75	0.288276	-0.072000
0.55	0.85	0.295149	-0.063025
0.55	0.95	0.339645	-0.048825

Eta	Gamma	Mean Drive	Mean Punishments
0.65	0.15	0.367870	-0.082650
0.65	0.25	0.365886	-0.083300
0.65	0.35	0.356469	-0.080675
0.65	0.45	0.303700	-0.091125
0.65	0.55	0.304852	-0.090500
0.65	0.65	0.288232	-0.104675
0.65	0.75	0.249474	-0.127875
0.65	0.85	0.229790	-0.117150
0.65	0.95	0.226971	-0.142500

Eta	Gamma	Mean Drive	Mean Punishments
0.75	0.15	0.254663	-0.117025
0.75	0.25	0.268852	-0.118675
0.75	0.35	0.348753	-0.113075
0.75	0.45	0.181157	-0.151275
0.75	0.55	0.256542	-0.121775
0.75	0.65	0.204855	-0.134350
0.75	0.75	0.215617	-0.135575
0.75	0.85	0.212224	-0.150725
0.75	0.95	0.161119	-0.165050

Eta	Gamma	Mean Drive	Mean Punishments
0.85	0.15	0.258602	-0.114300
0.85	0.25	0.230051	-0.146425
0.85	0.35	0.154752	-0.183475
0.85	0.45	0.303164	-0.122475
0.85	0.55	0.188479	-0.149725
0.85	0.65	0.159257	-0.190075
0.85	0.75	0.149849	-0.159750
0.85	0.85	0.141476	-0.189275
0.85	0.95	0.172940	-0.129575

Eta	Gamma	Mean Drive	Mean Punishments
0.95	0.15	0.171779	-0.168325
0.95	0.25	0.220243	-0.130800
0.95	0.35	0.159746	-0.173475
0.95	0.45	0.194691	-0.141250
0.95	0.55	0.149490	-0.184600
0.95	0.65	0.250968	-0.133525
0.95	0.75	0.187679	-0.168350
0.95	0.85	0.133540	-0.153775
0.95	0.95	0.153252	-0.149600

10.6 Moderate Environment (Local Net)

Eta	Gamma	Mean Drive	Mean Punishments
0.15	0.15	0.084103	-0.189575
0.15	0.25	0.152079	-0.090675
0.15	0.35	0.206224	-0.169775
0.15	0.45	0.151406	-0.219275
0.15	0.55	0.201154	-0.059775
0.15	0.65	0.163783	-0.104700
0.15	0.75	0.315227	-0.073050
0.15	0.85	0.176237	-0.171850
0.15	0.95	0.300683	-0.048325

Eta	Gamma	Mean Drive	Mean Punishments
0.25	0.15	0.236099	-0.132950
0.25	0.25	0.236738	-0.162800
0.25	0.35	0.227450	-0.155150
0.25	0.45	0.216267	-0.140525
0.25	0.55	0.152024	-0.194875
0.25	0.65	0.105212	-0.125625
0.25	0.75	0.324559	-0.042950
0.25	0.85	0.153444	-0.091625
0.25	0.95	0.249039	-0.032725

Eta	Gamma	Mean Drive	Mean Punishments
0.35	0.15	0.209131	-0.067650
0.35	0.25	0.256794	-0.094250
0.35	0.35	0.273450	-0.146475
0.35	0.45	0.315240	-0.142275
0.35	0.55	0.429820	-0.036400
0.35	0.65	0.305853	-0.058875
0.35	0.75	0.224819	-0.056150
0.35	0.85	0.196383	-0.036150
0.35	0.95	0.359582	-0.025250

Eta	Gamma	Mean Drive	Mean Punishments
0.45	0.15	0.141573	-0.083875
0.45	0.25	0.247326	-0.040625
0.45	0.35	0.213825	-0.099750
0.45	0.45	0.165876	-0.115150
0.45	0.55	0.520093	-0.061025
0.45	0.65	0.291071	-0.029875
0.45	0.75	0.136062	-0.064150
0.45	0.85	0.248516	-0.044175
0.45	0.95	0.157766	-0.074750

Eta	Gamma	Mean Drive	Mean Punishments
0.55	0.15	0.084229	-0.123100
0.55	0.25	0.182247	-0.057225
0.55	0.35	0.251080	-0.045775
0.55	0.45	0.358332	-0.018150
0.55	0.55	0.381544	-0.020275
0.55	0.65	0.230947	-0.109775
0.55	0.75	0.393454	-0.024750
0.55	0.85	0.271836	-0.044375
0.55	0.95	0.382445	-0.005800

Eta	Gamma	Mean Drive	Mean Punishments
0.65	0.15	0.089856	-0.096575
0.65	0.25	0.129290	-0.071025
0.65	0.35	0.229475	-0.052550
0.65	0.45	0.278201	-0.031725
0.65	0.55	0.299631	-0.032550
0.65	0.65	0.293848	-0.029200
0.65	0.75	0.328741	-0.021375
0.65	0.85	0.237497	-0.029275
0.65	0.95	0.411692	-0.007100

Eta	Gamma	Mean Drive	Mean Punishments
0.75	0.15	0.075359	-0.108825
0.75	0.25	0.127317	-0.080425
0.75	0.35	0.238172	-0.054100
0.75	0.45	0.271138	-0.030525
0.75	0.55	0.269083	-0.040925
0.75	0.65	0.255615	-0.043700
0.75	0.75	0.258751	-0.031050
0.75	0.85	0.308711	-0.030000
0.75	0.95	0.502500	-0.009825

Eta	Gamma	Mean Drive	Mean Punishments
0.85	0.15	0.064330	-0.176700
0.85	0.25	0.105268	-0.121975
0.85	0.35	0.127270	-0.096000
0.85	0.45	0.188284	-0.063750
0.85	0.55	0.169668	-0.064100
0.85	0.65	0.248534	-0.034525
0.85	0.75	0.242013	-0.037575
0.85	0.85	0.347513	-0.015050
0.85	0.95	0.471379	-0.013250

Eta	Gamma	Mean Drive	Mean Punishments
0.95	0.15	0.106114	-0.304125
0.95	0.25	0.101742	-0.235050
0.95	0.35	0.094845	-0.152625
0.95	0.45	0.147107	-0.100250
0.95	0.55	0.159222	-0.073600
0.95	0.65	0.193398	-0.065750
0.95	0.75	0.218947	-0.049650
0.95	0.85	0.285720	-0.026675
0.95	0.95	0.391298	-0.008100

10.7 Moderate Environment (MLP Net)

Eta	Gamma	Mean Drive	Mean Punishments
0.15	0.15	0.273188	-0.048375
0.15	0.25	0.271255	-0.050325
0.15	0.35	0.283172	-0.047950
0.15	0.45	0.283264	-0.046300
0.15	0.55	0.272180	-0.049650
0.15	0.65	0.291030	-0.043475
0.15	0.75	0.282733	-0.050050
0.15	0.85	0.292352	-0.046725
0.15	0.95	0.295662	-0.045975

Eta	Gamma	Mean Drive	Mean Punishments
0.25	0.15	0.242470	-0.068275
0.25	0.25	0.236251	-0.067850
0.25	0.35	0.251340	-0.061200
0.25	0.45	0.256728	-0.046350
0.25	0.55	0.271297	-0.048225
0.25	0.65	0.266590	-0.048650
0.25	0.75	0.272952	-0.041375
0.25	0.85	0.271882	-0.033300
0.25	0.95	0.270995	-0.043375

Eta	Gamma	Mean Drive	Mean Punishments
0.35	0.15	0.274911	-0.069550
0.35	0.25	0.261213	-0.066625
0.35	0.35	0.260285	-0.076675
0.35	0.45	0.255892	-0.059150
0.35	0.55	0.251034	-0.063575
0.35	0.65	0.264118	-0.046775
0.35	0.75	0.259463	-0.052350
0.35	0.85	0.287025	-0.041675
0.35	0.95	0.281455	-0.037925

Eta	Gamma	Mean Drive	Mean Punishments
0.45	0.15	0.270329	-0.077650
0.45	0.25	0.288927	-0.073325
0.45	0.35	0.282282	-0.079075
0.45	0.45	0.274630	-0.071375
0.45	0.55	0.261735	-0.064250
0.45	0.65	0.248809	-0.060000
0.45	0.75	0.260668	-0.064975
0.45	0.85	0.242404	-0.046225
0.45	0.95	0.254584	-0.060800

Eta	Gamma	Mean Drive	Mean Punishments
0.55	0.15	0.331515	-0.070950
0.55	0.25	0.296144	-0.070200
0.55	0.35	0.256294	-0.078925
0.55	0.45	0.300324	-0.068425
0.55	0.55	0.262601	-0.079025
0.55	0.65	0.261267	-0.080375
0.55	0.75	0.266599	-0.072325
0.55	0.85	0.246314	-0.066500
0.55	0.95	0.253722	-0.055600

Eta	Gamma	Mean Drive	Mean Punishments
0.65	0.15	0.312595	-0.080875
0.65	0.25	0.334592	-0.069875
0.65	0.35	0.320217	-0.067475
0.65	0.45	0.282496	-0.075600
0.65	0.55	0.261043	-0.088625
0.65	0.65	0.260109	-0.094525
0.65	0.75	0.267517	-0.079825
0.65	0.85	0.193497	-0.105200
0.65	0.95	0.188624	-0.104725

Eta	Gamma	Mean Drive	Mean Punishments
0.75	0.15	0.278154	-0.090675
0.75	0.25	0.271257	-0.088750
0.75	0.35	0.255664	-0.089650
0.75	0.45	0.238830	-0.093800
0.75	0.55	0.294758	-0.094875
0.75	0.65	0.230104	-0.115100
0.75	0.75	0.194804	-0.122025
0.75	0.85	0.156686	-0.118875
0.75	0.95	0.194546	-0.112525

Eta	Gamma	Mean Drive	Mean Punishments
0.85	0.15	0.229063	-0.098850
0.85	0.25	0.243749	-0.092650
0.85	0.35	0.276738	-0.085350
0.85	0.45	0.188700	-0.129875
0.85	0.55	0.303756	-0.077575
0.85	0.65	0.226284	-0.111475
0.85	0.75	0.219789	-0.119500
0.85	0.85	0.196605	-0.104450
0.85	0.95	0.155973	-0.118875

Eta	Gamma	Mean Drive	Mean Punishments
0.95	0.15	0.160683	-0.140775
0.95	0.25	0.326494	-0.097975
0.95	0.35	0.196699	-0.128875
0.95	0.45	0.136524	-0.144900
0.95	0.55	0.122574	-0.159550
0.95	0.65	0.116809	-0.149275
0.95	0.75	0.093874	-0.176050
0.95	0.85	0.114764	-0.145750
0.95	0.95	0.106629	-0.173675

References

Ackley, D. H. and M. L. Littman (1991). Generalisation and scaling in reinforcement learning. In *Advances in Neural Information Processing Systems*, Volume 2, pp. 550–557.

Agre, P. (1997). *Computation and Human Experience*. Cambridge University Press.

Agre, P. E. (1988). *The Dynamic Structure of Everyday Life*. Ph.D. thesis, Massachusetts Institute of Technology, Deparment of Electrical Engineering and Computer Science.

Arbib, M. A. (1989). *The Mataphorical Brain 2: Neural Networks and Beyond*. John Wiley and Son.

Arbib, M. A. (Ed.) (1995a). *The Handbook of Brain Theory and Neural Networks*. The MIT Press, Cambridge, MA.

Arbib, M. A. (1995b). Levels and styles of analysis. See Arbib (1995a), pp. 11–17.

Arkin, R. C. (1998). *Behaviour-Based Robotics*. The MIT Press.

Ballard, D. H. (1997). *An Introduction to Natural Computation*. The MIT Press: Cambridge, MA.

Barto, A. G. and R. S. Sutton (1981). Landmark learning: an illustration of associative search. *Biological Cybernetics 42*, 1–8.

Barto, A. G., R. S. Sutton, and C. W. Anderson (1983). Neuronlike adaptive elements that can solve difficult learning problems. *IEEE Transactions on Systems, Man and Cybernetics 13*(5), 834–846.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Blake, C. and C. Merz (1998). UCI repository of machine learning databases.

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs with relationships to statistical pattern recognition. In F. Fogelman Soulié and J. Herault (Eds.), *Neurocomputing*. Springer Verlag, Berlin.

Brooks, R. A. (1997). From earwigs to humans. *Robotics and Autonomous Systems 20*, 291–304.

Brown, T. H. and S. Chatterji (1995). Hebbian synaptic plasticity. See Arbib (1995a), pp. 454–458.

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

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

Catania, A. C. (1992). *Learning*. Prentice-Hall International Editions, Inc.

Chang, C. and P. Gaudiano (1998). Application of biological learning theories to mobile robot avoidance and approach behaviours. *Journal of Complex Systems 1*, 79–114.

Cohen, P. R. (1995). *Empirical Methods for Artificial Intelligence*. MIT Press.

Dennett, D. C. (1996). *Kinds of Minds*. London: Phoenix/Orion Books.

Dobie, M., R. Tansley, D. Joyce, M. Weal, P. Lewis, and W. Hall (1999). A flexible architecture for content and concept based multimedia information exploration. In *Proceedings of the Challenge of Information Retrieval*, pp. 1–12. Also available from `www.bib.ecs.soton.ac.uk/`.

Doya, K. (1999). Metalearning, neuromodulation and emotion. In *Abstracts of the 13th Toyota Conference on Affective Minds*, pp. 46–47. Also available from `http://www.erato.atr.co.jp/ doya/`.

Dreyfus, H. L. (1992). *What Computers Still Can't Do*. The MIT Press, Cambridge, MA.

Etzioni, O. (1993). Intelligence without Robots (A reply to Brooks). *Artificial Intelligence Magazine 14*(4), 7–13.

Foley, D. H. (1972). Considerations of sample and feature size. *IEEE Transactions on Information Theory 18*(5), 618–626.

Grossberg, S. (1972a). A neural theory of punishment and avoidance. i: Qualitative theory. *Mathematical Biosciences 15*, 39–67.

Grossberg, S. (1972b). A neural theory of punishment and avoidance. ii: Quantitative theory. *Mathematical Biosciences 15*, 253–285.

Grossberg, S. and D. S. Levine (1987). Neural dynamics of attentionally modulated pavlovian conditioning: Blocking, interstimulus interval, and attentive feedback. *Applied Optics 26*, 5015–5030.

Hebb, D. (1949). *The Organisation of Behaviour.* John Wiley and Sons, New York.

Hilgard, E. R. and G. H. Bower (1966). *Theories of Learning (3rd Edition).* Meredith Publishing Company.

Huhns, M. N. and M. P. Singh (Eds.) (1998). *Readings in Agents.* Morgan Kaufman Publishers, San Francisco.

Humphrys, M. (1997). *Action Selection methods using Reinforcement Learning.* Ph.D. thesis, University of Cambridge, Computer Laboratory.

Jennings, N. R., K. Sycara, and M. Wooldridge (1998). A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems 1*, 275–306.

Joyce, D. W. (2001a). Modifying situated automata theory for connectionist agents (ECSTR-IAM004). Technical report, IAM Research Group, Department of Electronics and Computer Science, University of Southampton.

Joyce, D. W. (2001b). *A Phenomenological-Connectionist Theory of Computational Agency (Volume I).* Ph.D. thesis, Department of Electronics and Computer Science, University of Southampton.

Joyce, D. W. and P. H. Lewis (1999a). Connectionist models for learning choice behaviour in reactive software agents. In *Proc. of the IASTED International Conf. on Artificial Intelligence and Soft Computing*, pp. 437–442.

Joyce, D. W. and P. H. Lewis (1999b). Modelling learning for intelligent software agents: A connectionist approach. In *Proceedings of the Second Workshop on Intelligent Virtual Agents*, pp. 143–146.

Joyce, D. W., P. H. Lewis, R. H. Tansley, M. R. Dobie, and W. Hall (2000). Semiotics and agents for integrating and navigating through media representations of concepts. In *Storage and Retrieval for Media Databases 2000, Proceedings of SPIE, Number 3972*, pp. 120–131.

Kaebling, L. P., M. L. Littman, and A. W. Moore (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research 4*, 237–285.

Kohonen, T. (1997). *Self Organizing Maps* (2 ed.). Springer-Verlag, Berlin, Heidelberg, New York.

Levine, D. S. (1991). *Introduction to Neural and Cognitive Modelling.* Lawrence-Erlbaum Associates.

Lewis, P. H., H. C. Davis, M. R. Dobie, and W. Hall (1997). Towards multimedia thesaurus support for media-based navigation. In A. W. M. Smeulders and R. Jain (Eds.), *Image Databases and Multimedia Search*, Volume 8, pp. 111–118. World Scientific.

MacLennan, B. J. and G. M. Burghardt (1994). Synthetic ethology and the evolution of cooperative communication. *Adaptive Behaviour 2*(2), 161–188.

Maturana, H. R. and F. J. Varela (Eds.) (1980). *Autopoiesis and Cognition. Boston Studies in the philosophy of science*. D Reidel Publishing Company.

Minsky, M. (1986). *The Society of Mind*. The MIT Press, Cambridge, MA.

Narendra, K. S. and M. A. L. Thathachar (1974). Learning automata – a survey. *IEEE Transactions on Systems, Man and Cybernetics 4*(4), 323–334.

Page, M. (2000). Connectionist modelling in psychology: A localist manifesto. *Behavioural and Brain Sciences 23*(4), 443–512.

Radhakrishan, P. and P. H. Lewis (1998). "Agent Like" support for content based retrieval and navigation. In *Proceedings of SPIE Storage and retrieval for image and video databases VI*, pp. 228–235.

Ruck, D. W., S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter (1990). The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *IEEE Transactions on Neural Networks 1*(4), 296–298.

Schmajuk, N. (1997). *Animal Learning and Cognition*. Cambridge University Press, Cambridge.

Skinner, B. F. (1938). *The Behaviour of Organisms*. New york, Appleton.

Spier, E. and D. McFarland (1998). Learning to do without cognition. In *From Animals to Animats 5: The Fifth International Conference on Simulation of Adaptive Behaviour (SAB98)*, pp. 38–47. The MIT Press.

Sun, R. and T. Peterson (1998). Autonomous learning of sequential tasks: Experiments and analyses. *IEEE Transactions on Neural Networks 9*(6), 1217–1234.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning 3*(1), 9–44.

Sutton, R. S. (1991). Dyna, an integrated architecture for learning, planning and reacting. In *Working notes of the 1991 AAAI Symposium*, pp. 151–155. AAAI Press.

Sutton, R. S. and A. G. Barto (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Tansley, R. H. (2000). *The Multimedia Thesaurus: Adding a Semantic Layer to Multimedia Information*. Ph.D. thesis, Department of Electronics and Computer Science, University of Southampton.

Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning 8*, 257–277.

Tesauro, G. (1995). Temporal difference learning and TD-gammon. *Communications of the ACM 38*(5), 58–68.

Usher, J. L. and J. L. McClelland (1995). On the time course of perceptual choice: A model based on principles of neural computation – PDP.CNS.95.5. Technical report, Center for the Neural Basis of Cognition and Carnegie Mellon University, Dept. of Psychology.

Varela, F. J., E. Thompson, and E. Rosch (1991). *The Embodied Mind*. The MIT Press.

Williamson, J. R. (1996). Gaussian ARTMAP: A neural network for fast incremental learning of noisy multi-dimensional maps. *Neural Networks 9*(5), 881–897.

Yuille, A. L. and D. Geiger (1995). Winner-take-all mechanisms. See Arbib (1995a), pp. 1056–1060.