Representations for Action Selection Learning from Real-Time Observation of Task Experts

Mark A. Wood and Joanna J. Bryson
Artificial models of natural Intelligence (AmonI)
Department of Computer Science
University of Bath, UK
{cspmaw,jjb}@cs.bath.ac.uk

Abstract

The association of perception and action is key to learning by observation in general, and to program-level task imitation in particular. The question is how to structure this information such that learning is tractable for resource-bounded agents. By introducing a combination of symbolic representation with Bayesian reasoning, we demonstrate both theoretical and empirical improvements to a general-purpose imitation system originally based on a model of infant social learning. We also show how prior task knowledge and selective attention can be rigorously incorporated via loss matrices and Automatic Relevance Determination respectively.

1 Introduction

Program-level imitation [Byrne and Russon, 1998], or the acquisition of behavioural structure from observation, is an under-researched field in AI. In robot AI, much effort has rightly been directed toward action-level imitation; the reproduction of movements involving a fine degree of motor control. Indeed, this highly complex and difficult task needs to be solved by a robot before it is able to acquire structural data. However, by using 'intelligent' virtual environments which implicitly deal with low-level actions, we can gain access to a rich class of higher-level problems. *Unreal Tournament* [Digital Extremes, 1999] is an example of such an environment, and it is popular with those few looking at this problem [Thurau *et al.*, 2004; Le Hy *et al.*, 2004]. It is also the domain of choice for the system that underpins this paper: COIL [Wood and Bryson, 2007].

In this paper we demonstrate how a formal Bayesian framework can be incorporated into a complex modular learning system. Through this we widen its potential applicability in theory, significantly improve its learning performance in practise, and add natural extensibility to the system via tried and tested Bayesian methodology. Our experiments also enable us to examine the broader issue of the combinatorial complexity of social learning. Social learning is an important mechanism for acquiring intelligent behaviour — arguably it accounts for the massive accumulation of culture and artifacts seen in humans compared to our nearest biological relatives, the other apes. Understanding the complexity characteristics of this task is key to both understanding human intelligence and harnessing the approach for AI.

1.1 System Overview

COIL is an adaption to generic imitation learning of Roy's language learning system, CELL. CELL is one of the most convincing examples to date of real-time interactive social learning. It enables a robot to learn the names for various toys using the same sorts of input as an infant. Both CELL and COIL are detailed elsewhere [Roy, 1999; Wood and Bryson, 2007]; here we give a skeletal overview of COIL, to clarify those parts most relevant to the extentions described later.

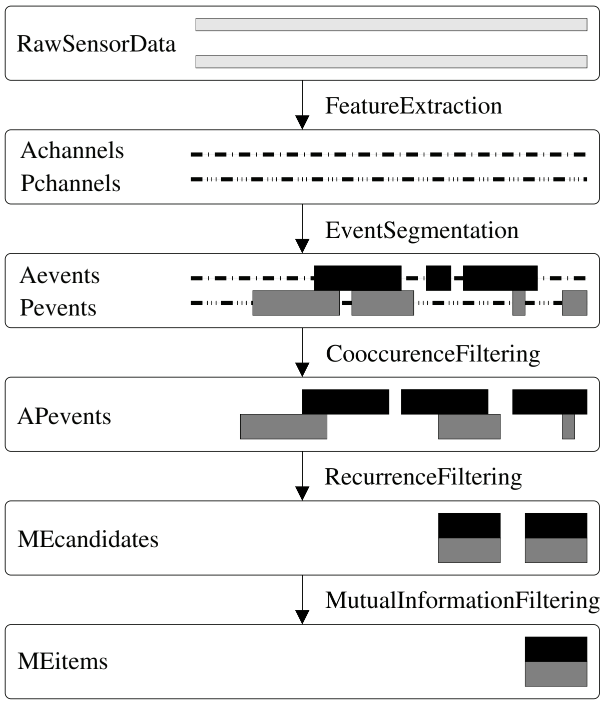


Figure 1: The inputs and outputs of each stage of COIL [Wood and Bryson, 2007].

The learning part of COIL has five constituent stages of processing (see Figure 1). It is designed to function embedded in an *imitator* agent observing a conspecific *expert* agent executing a given task in a shared environment. The input to the first stage consists of the incoming raw data from the imitator's sensors. During **Feature Extraction**, these data are directed into separate *channels* which are one of two types: Action channels receive data pertaining to the actions taken by the expert, and Perception channels receive data pertaining (but not identical) to the perceptual state of the expert. Once in channels the data are segmented into *A*- and *P-events* respectively, and then further into *A*- and *P-subevents* according to a set of pre-programmed triggers; this is **Event Segmentation**. The **Co-Occurrence Filter** then simply binds co-occurring A- and P-events together as *AP-events* and shunts them to a queue called STM (Short Term Memory). Whenever a new AP-event arrives in STM, it is compared with each of those already there in turn. The **Recurrence Filter** scans for co-occurring A- and P-subevent pairs and, if any are found, binds them as a Motivation-Expectation or *M-E Candidate* and stores them in MTM (Mid Term Memory). Finally, the **Mutual Information Filter** calculates a mutual information score for each M-E Candidate using a somewhat complex algorithm not described here (see [Wood and Bryson, 2007] for details). Those candidates whose scores exceed a pre-determined threshold are saved in LTM (Long Term Memory as *M-E Items*.

The M-E Items themselves represent observed perception-action pairs (though note the original agent may not have had this model [Bryson and Wood, 2005]), and can consequently be used to create a reactive imitative behaviour. The imitator's sensors define a perception space which it will move through as the task surroundings change. M-E Items can be used to create maps from regions of this space onto the imitator's action repertoire. In practise, the imitator searches the perception chunks stored in LTM for those that match its current perceptual state. If matches are found, the highest priority match is selected (ranked by mutual information score), and then the associated action chunk is returned and executed.

1.2 System Shortcomings

Although COIL has had some previously-reported success in performing imitation tasks, we have discovered a number of flaws which prevent it from scaling to more difficult problems.

Representations

COIL's primary weakness results from trying to represent general action and perception with the same system CELL uses for speech and vision. CELL receives continuous microphone data, later converted into discrete phonemes during Event Segmentation. COIL receives continuous action data which is parsed into discrete action segments. The crucial difference is between *the spaces in which these discrete objects lie*. We omit the details of Roy's metric here (see [Roy, 1999, p. 106]), but intuitively both phonemes and shapes can have infinite variation and can be mapped into a continuous space using histograms where the notion of 'nearness' is relatively well-defined. In contrast, the limited set of executable actions[1] required for an imitation task lie in a discrete space where 'nearness' is not only hard to define but not a particularly useful concept. For example, how far is `jump` from `turn_right`, and what use would that information be anyway? Perceptual channels are similarly unrelated since we are not simply learning to respond to different shapes, but are interested in different proprioceptive / physical positioning as well as more thoroughly categorised 'visual' stimuli (e.g. identifying an object *vs.* an agent). These comparisons point toward using a more thoroughly discrete representation for both actions and perceptions throughout COIL.

An obvious solution might appear to be to reduce the level of abstraction at which the learning and action selection occurs. For example, by descending into finer-grained muscle and motor position spaces we would reclaim continuity. There are two problems here: firstly, both of these spaces have unusual discontinuities which lead to the problems of inverse kinematics which ensnare traditional robotics. Secondly, the complexity inherent in the remaining imitation (and subsequent action selection) process demands a reduced space of more generic operators. Thus rather than learning 180 different `turn-right` commands, each varying by only one degree of turn arc, we simply learn to *initiate* `turn-right`, and then check for stopping criteria. Note that this solution is more robust to uncertain motion and sensing, assuming the stopping or homing criteria are perceivable within some range of degrees during the turn.

Algorithms

The performance of COIL is further affected by certain characteristics inherited from the CELL algorithms. Firstly, the Recurrence Filter is designed to search for events recurring only within a short history of observations. This is appropriate for aspects of infant learning that assume frequent repetition, e.g. of word/object pairs, but task learning in general may have arbitrarily long gaps between recurring perception/action pairs. The problem with simply increasing the size of the Recurrence Filter window is that every arriving AP-event is compared to all those already present, which clearly results in combinatorial problems as the size of the queue increases. The Mutual Information Filter has a similar scaling problem. It has cubic complexity in the number of elements in MTM (which in general could grow without bound) and exponential in the number of monitored channels (which grows with the complexity of the task domain). Additionally, the probabilities used in the calculation of the mutual information are frequentist (as opposed to Bayesian) approximations, and are consequently very sensitive to noise caused by small frequency counts (ie. rarely observed events). Roy tackles this by interpolating these probabilities with priors, but the choice of prior mass parameter required by this technique can have significant effects on the resulting probabilities, particularly if many of the events in question are infrequent. This may well be the case for our applications, so we desire a more robust method.

[1]Limited in that relatively few actions can be *initiated* at any given time.

2 Model

We therefore wish to implement a learning algorithm which can operate within the COIL framework and minimise the potential problems outlined above. Specifically, it should be:

Scalable - both in terms of memory requirements and learning complexity.

Incremental - so that all observations are used and knowledge is consolidated in a natural way.

Rigorous - having output that is interpretable and justifiable through established mathematical argument.

Robust - not prone to failure when processing unusual, unforseen or extreme events.

It would also be preferable for this algorithm to be sufficiently general-purpose to be applied to other similar learning problems in this field. The Bayesian framework would seem a good place to begin, as it allows each observed event to update prior belief in an efficient and well-defined manner [Bishop, 1995, p. 17]. However, there are many algorithms which make use of it, so the question becomes which one to use in order to obtain the desired posterior beliefs. We chose a multi-layer perceptron (MLP) architecture which, given a certain set of constraints, provides Bayesian posterior probabilities as its outputs. We describe this specific configuration in the next section.

2.1 Network Architecture

The parts of COIL most prone to the kinds of problems described above are the Recurrence and Mutual Information filters. To replace these, we therefore require that the new algorithm receive perception-action data from the Co-occurrence filter and output a behavioural map. In MLP terms, this map looks like a classifier network, which receives perceptual data and assigns it to the appropriate action class. To allow an MLP to learn such a classification, we must translate the observed perception-action pairs into an appropriate set of training examples. Each example should consist of a set of input variables and a set of target variables. In this case, the input variables should correspond to the observed perceptual state of the expert, and the target variables should correspond to the observed action. The question is, what encoding to use for these variable sets?

As explained above, we are assuming for now that perceptual categories (such as `item_left` and `no_item_visible`) have no implicit relationship to each other. They do not lie in a metric space and so cannot be represented by ordinal variables. We therefore use a purely categorical *1-of-c* encoding for the input [Bishop, 1995, p. 300]. Suppose a given Perception channel has n possible symbolic states. Then symbol i can be represented by a vector of n bits where only the i^{th} bit is set ($1 <= i <= n$). If there are m Perception channels then the concatenation of m such vectors produces the complete required binary input vector of length $n_1 + ... + n_m$. If there are k observable actions, then this is equivalent to a classification problem with k target classes, and we can create one output node for each class. We have already stated that it is desirable for these outputs to have a Bayesian interpretation as posterior probabilities of action class membership for a given perceptual state. This is achievable using a softmax activation function for the network output units [Bridle, 1990] and minimising a cross-entropy error function for the network weights [Bishop, 1995, p. 237]. After some empirical testing, we chose to include three hidden units in the network, although the results were not particularly sensitive to this choice. We are currently looking into Bayesian model selection techniques for selecting the number of hidden units (see Section 4). Given the above node structure, the network we used was fully connected with a simple feedforward structure, as shown in Figure 2.

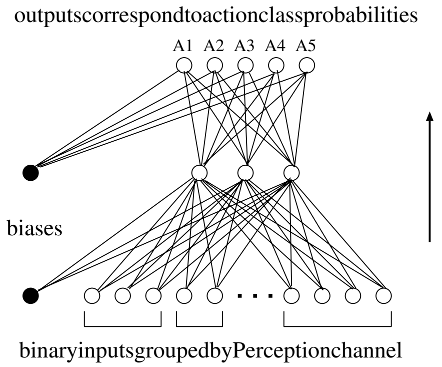

outputscorrespondtoactionclassprobabilities
A1
A2
A3
A4
A5
biases
binaryinputsgroupedbyPerceptionchannel

Figure 2: Diagram of MLP architecture. The binary input vector is a concatenation of 1-of-c encoded symbols for each Perception channel. There are three hidden units with softmax activation to the outputs, which consequently correspond to posterior probabilities of action class membership. Arrow shows direction of forward propagation; biases are shown explicitly for clarity.

The network training scheme uses Bayesian regularisation with separate hyperparameters for each of four weight groups: first-layer weights, first-layer biases, second-layer weights and second-layer biases [Bishop, 1995, p. 408]. Training was by back-propagation for up to 100 cycles of scaled conjugate gradient search (fewer if convergence occurred beforehand), followed by re-estimation of the hyperparameters using the *evidence approximation* technique [MacKay, 1992b]. This cycle of re-estimation and re-training was carried out 8 times. The test data for the network consisted of querying all possible combinations of perceptual state, to evaluate the most probable action assigned to that state by the classifier. Finally, these posterior probabilities were marginalised according to the observed data [MacKay, 1992a]. Although this last step does not affect the most probable action class, it can have significant effects if loss matrices are added (see Section 3.3 below).

Empirical evidence showing the increased *learning* performance of the new algorithm can be found in the next section. Before we examine this however, we review the theoretical criteria set out at the beginning of this section and ask if they are satisfied.

Scalable - as far as learning complexity is concerned, network training time increases only linearly with the number of observed events, as compared to the combinatorics of the original algorithms (see Section 1.2). Also, the MLP is a function which requires storage equal to the number of network weights as opposed to a potentially boundless number of stored M-E Items.

Incremental - due to this increase in efficiency and the belief accumulation property of the Bayesian framework, every observation can be taken into account and consolidated with prior knowledge.

Rigorous - the fact that we can interpret the MLP outputs as posterior probabilities is a well-proven property of this type of network and totally independent of the domain in which we're working.

Robust - the parametric re-estimation carried out after each network training cycle serves to minimise any problems caused by local minima relating to, say, weight initialisation.

3 Experiments

To evaluate our new model we tested it against the same data collected for the original COIL experiments. These data were gathered in *Unreal Tournament* (UT), a commercially released, multi-player 'First Person Shooter' (game) [Digital Extremes, 1999]. As the term suggests, the user has an agent's-eye view of the game and direct, real-time control of an avatar's actions. UT also supports remote control of agents by sending commands to the game server over a network. This provides a framework for allowing external programs to direct an agent's actions. As such, UT provides a viable platform for testing strong AI, since humans and AI can compete and interact in a real-time, spatially situated domain. The game server sends two categories of sensor data back to the client. The first is synchronous: at regular intervals the client is informed of the agent's status (e.g. health, ammo, current weapon, etc). The second is asynchronous: for example whenever a wall is bumped, a footstep is heard or damage is taken.

These data when viewed as a whole are a highly dynamic, high-dimensional mixture of categorical and continuous variables, akin to sensory data acquired in the real world. Other similarities include physics simulation, such as collisions, gravity, and sound and light transmission. AI agents' (*bots'*) sensor data are incomplete in the sense that only a reduced subset of the game variables are observable; the bots have limited virtual sensors. For example, the imitator cannot know the health state of the expert, although this may well affect the expert's choices. The environment contains many independent features, each of which could be represented in a number ways. Thus the problem of assimilating the behaviour of another agent via observation is far from simple. The first three stages of COIL each serve to reduce the complexity of this problem (see Section 1.1) so that it arrives at the inputs of our new algorithm in a tractable state.

In short, UT agents deal with real-world temporally-bounded cognitive constraints, not least the combinatorial complexity of learning, which makes them ideal test subjects for our research.

3.1 Task 1

The first task involved collecting *health vials*, one of many so-called 'pickups' available in UT, from various locations within a game map. The data was originally received and processed by a COIL system embedded in an AI-controlled bot, programmed to observe from within the environment a human-controlled bot carrying out the task. We used three different 'tactics' during our demonstrations:

CW Tend to turn clockwise if no vials are visible.

ACW Tend to turn anticlockwise.

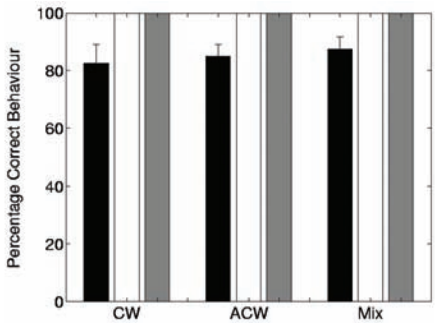
Mix No fixed tendency.

Ten trials for each tactic were carried out, for a total of 30 trials each lasting approximately 60 seconds. During the original experimental runs, the data arriving in channels (ie. post Feature Extraction) were saved prior to further processing. This allowed us to compare the new learning algorithms on the same data sets. The MLP replaces only the recurrence and mutual information filtering stages of COIL, with the first three stages remaining unchanged. For further performance comparison, we also fed this data into a decision tree algorithm, C4.5 [Quinlan, 1992].

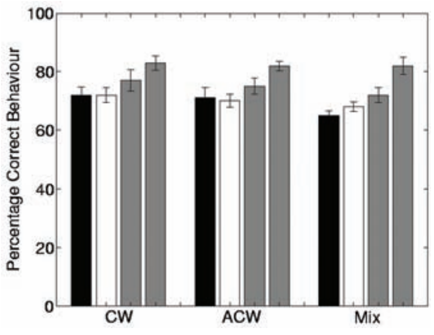
Our performance metric derives from our representation of behaviour as a mapping from discrete regions of perceptual space to discrete actions. We defined the behaviour on which we based our demonstrations as the 'ideal' map from perception to action for this task. The proportion of the learned behaviour which matches the ideal allowed us to assign a 'percentage correct behaviour' score to each trial. The results comparing the three techniques are shown in Figure 3(a). As can be seen from the figure, the MLP (grey bars) generated universally perfect behaviour for this task, correcting all errors made by COIL's native algorithms (black bars). Interestingly though, C4.5 (white bars) also performs a perfect classification

3.2 Task 2

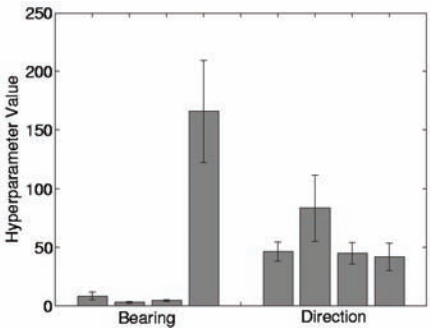
The second task required the expert to locate and destroy enemy bots in an environment which also contained an equal number of friendly bots. Each trial lasted as long as it took for this task to be completed; typically around 60 seconds. Tactics CW, ACW and Mix were used analogously to Task 1, again with ten trials each for a total of 30. All algorithms and training methods remained the same for this task as for the previous one. Results are summarised in Figure 3(b). The MLP (left-hand grey bar in each group) provides a small but not signifcant (t-test, $p = 0.05$) increase in performance from both COIL (black bars) and C4.5 (white bar), which for this task performs no better than COIL. Upon inspection of the data it is clear that for a majority of the trials, the associations that would be necessary to form a fully correct behaviour are never observed. Specifically, most of the misclassifications are made for turning toward an enemy; in the absence of such associations the imitator tended to adopt the dominant turning direction observed and consequently err in either the `enemy_left` or `enemy_right` state. The other common mistake was to fire before the enemy was centred in sights; both were made less by the network than the other



(a) Task 1



(b) Task 2



(c) ARD

Figure 3: Comparative performance of the different learning algorithms over a variety of tasks. The black bars correspond to the original COIL algorithms, the white bars correspond to C4.5 and the grey bars correspond to the new MLP classifications. In the second study, the right-hand grey bar corresponds to the system moderated by a loss matrix (see text for details). The third study shows the automatically determined relative importance of two Perception channel input sets. Error bars show the standard error of the means.

algorithms. Performance is further improved, this time significantly (t-test, $p = 0.05$), by introducing a loss matrix to represent prior task knowledge (right-hand grey bars in each group); one of the extentions we go on to talk about in the next section.

3.3 Bayesian Extentions

As discussed in Section 2, the probabilistic interpretation of results possible from the network model is highly desirable. This also allows other Bayesian techniques to be applied to the network and its outputs. We now discuss two such techniques and show preliminary results.

Loss Matrices

In general decision theoretic terms, a *loss matrix* describes the relative penalties associated with misclassifying a given input [Bishop, 1995, p.27]. In this case we can describe the matrix as having elements L_{kj} representing the loss resulting from assigning an action A_j to a given perceptual state when in fact A_k should be assigned. Decisions are then made by minimising the *risk*, or equivalently by using the following discriminant to classify a given perceptual state $\mathbf{x}$:

$$\sum_{k=1}^{c} L_{kj} P(A_k|\mathbf{x}) < \sum_{k=1}^{c} L_{ki} P(A_k|\mathbf{x}) \qquad \forall\, i \neq j \tag{1}$$

where $P(A_k|\mathbf{x})$ can be obtained from the (marginalised) network output probabilities. To demonstrate this we applied a simple loss matrix to the networks generated during Task 2:

$$(L_{kj}) = \begin{pmatrix} 0 & 5 & 5 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \tag{2}$$

where A_1 is the `fire` action, A_2 is `turn_left` and A_3 is `turn_right`. This matrix specifies that 'accidentally' firing instead of correctly turning should incur five times greater a penalty than any other misclassification[2]. Informally, one would expect this to be equivalent to giving the imitator the instruction "only shoot if you're sure", prior to acting. The results of applying this matrix to the Task 2 network outputs are shown in Figure 3(b) (right-hand grey bars in each group). The improvement, as expected, is due to fewer cases of firing before the enemy is in position. Although this is a relatively simple example of the application of this technique, it does demonstrate the ease at which prior knowledge can be formally incorporated into the model, and how it could be systematically altered to test the effect on output behaviour.

Selective Attention

It is likely that for a given task, only a small subset of the full available perceptual state will be required for good performance. So far in this paper, this subset has been chosen by hand, but the MLP model can enable us to make this selection automatically, within a sound Bayesian framework. This *Automatic Relevance Determination* [Neal, 1996, p. 15] is achieved by using a different hyperprior. Instead of grouping the weights such that there are four independent regularisation hyperparameters, the weights associated with *each input* have their own hyperparameter. These coefficients vary in proportion to the *inverse* posterior variance of the weight distribution for that input. Thus if a given input has a high coefficient value, the weight distribution for that input has a low variance and the input has little bearing on the ultimate classification: the input has *low relevance*. Using a similar training and re-estimation scheme as described earlier, these hyperparameters can be used to determine the relative relevance of the different network inputs, which in this case correspond to different aspects of the environment. Thus we have a method for automatic attention selection within a broader set of chan-

[2]Note a loss matrix with zeros on the main diagonal and ones everywhere else describes a discriminant equivalent to simply choosing the class with the greatest posterior probability.

nels.

To test this theory, we added a Perception channel to Task 1 which identified the *absolute direction* the imitator was facing, represented by one of four 'compass' symbols. One would expect this set of inputs to have lower relevance than the existing channel relating to the relative bearing of the vials. We carried out a further ten trials under much the same conditions as Task 1. The results are summarised in Figure 3(c). As expected, the coefficient values for the inputs associated with the new channel are significantly higher on average than the inputs associated with the original channel. The exception to this is the fourth Bearing input which (bearing in mind the 1-of-c encoding) was fully determined by the state of the first three inputs. Given these inputs converged to high relevance, the fourth was correctly deemed of very low relevance. In principle, this technique could be used to prune perception space down to make local task learning more accurate and efficient. To allow this, some kind of 'relevance threshold' would have to be chosen, which may well vary from task to task. The method for making such a choice remains an open question.

4 Conclusion

We have presented a new and significantly improved version of the COIL system for program-level imitation. In doing this, we have shown that taking advantage of Bayesian reasoning, along with the large body of systems-independent research that comes with it, can allow for improvement both in the theoretical bounds of a system and its empirical performance. This may require some representational transformation, but in return unlocks an arsenal of well-established techniques that can be brought to bear on practical problems in novel ways. Also, the techniques themselves may suggest new avenues of research, and therefore provide the system with natural extensibility.

The fundamental issue of interest to us, motivating and examined in both this paper and its parent, is the combinatorial complexity of social learning. In this case study, we have compared COIL's perception-action association storage method with that of learning a function from perception to action. The former is both more memory-intensive and more search-intensive due to COIL's native representations and algorithms, but is in fact an arbitrarily flexible specification for behaviour. In practise for simple tasks, and in theory for more complex tasks, the functional approach is more efficient. However, it is likely that there will come a point where the task domain becomes too dynamic and hierarchical for behaviour to be adequately specified by a single MLP, or indeed by any 'stock' function.

Our latest work suggests two possible alternative solutions: firstly a hierarchical system of functions, monitoring local task domains at the lower level and arbitrating between task domains at the higher. This would still require an external process to exchange information between functions. Secondly, we could apply the hierarchical model to an improved association storage method. By using perceptual prioritisation and action concatenation we can carefully control the trade-off between performance and complexity while at the same time capitalising on the flexibility of this approach.

Acknowledgements

The research presented is partially funded by an EPSRC DTA Studentship, administered by the University of Bath Department of Computer Science.

References

[Bishop, 1995] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[Bridle, 1990] J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. *Neurocomputing: Algorithms, Architectures and Applications*, pages 227–236, 1990.

[Bryson and Wood, 2005] Joanna J. Bryson and Mark A. Wood. Learning discretely: Behaviour and organisation in social learning. In Yiannis Demiris, editor, *Third International Symposium on Imitation in Animals and Artifacts*, pages 30–37, Hatfield, UK, April 2005. The Society for the Study of Artificial Intelligence and the Simulation of Behaviour.

[Byrne and Russon, 1998] Richard W. Byrne and Anne E. Russon. Learning by imitation: a hierarchical approach. *Behavioral and Brain Sciences*, 16(3), 1998.

[Digital Extremes, 1999] Digital Extremes. *Unreal Tournament*, 1999. Epic Games, Inc.

[Le Hy *et al.*, 2004] Ronan Le Hy, Anthony Arrigoni, Pierre Bessière, and Olivier Lebeltel. Teaching bayesian behaviours to video game characters. *Robotics and Autonomous Systems*, 47:177–185, 2004.

[MacKay, 1992a] David J. C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(5):720–736, 1992.

[MacKay, 1992b] David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

[Neal, 1996] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer, August 1996.

[Quinlan, 1992] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, October 1992.

[Roy, 1999] Deb Kumar Roy. *Learning from Sights and Sounds: A Computational Model*. PhD thesis, MIT, Media Laboratory, September 1999.

[Thurau *et al.*, 2004] Christian Thurau, Christian Bauckhage, and Gerhard Sagerer. Learning human-like movement behavior for computer games. In *Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior (SAB'04)*, 2004.

[Wood and Bryson, 2007] Mark A. Wood and Joanna J. Bryson. Skill acquisition through program-level imitation in a real-time domain. *IEEE Transaction on Systems, Man and Cybernetics, Part B: Cybernetics*, 2007. In press.