Learning Where To Go without Knowing Where That Is: The Acquisition of a Non-reactive Mobot Behaviour by Explicitation

Chris Thornton
Cognitive and Computing Sciences
University of Sussex
Brighton BN1 9QN
Email: Chris.Thornton@cogs.susx.ac.uk
Tel: (44)273 678856

December 14, 1994

Abstract

In the path-imitation task, one agent traces out a path through a second agent's sensory field. The second agent then has to reproduce that path exactly, i.e. move through the sequence of locations visited by the first agent. This is a non-trivial behaviour whose acquisition might be expected to involve special-purpose (i.e., strongly biased) learning machinery. However, the present paper shows this is not the case. The behaviour can be acquired using a fairly primitive learning regime provided that the agent's environment can be made to pass through a specific sequence of dynamic states.

Introduction

In reactive mobot behaviours such as wall-following and pursuit, specific stimuli evoke specific responses. As a result, the input/output profile for the behaviour shows marked correlations. The existence of these means that the behaviour can be straightforwardly acquired using any one of the wide range of reinforcement, neural-network and evolutionary methods — in fact anything capable of exploiting statistical effects. In non-reactive behaviours, responses are reactions to hidden states, i.e., states which are temporally, physically or logically inaccessible to the agents sensors. Such a behaviour's input/output profile does *not* show marked correlations and thus cannot be straightforwardly acquired using conventional techniques.

Path-imitation is a case in point. In the path-imitation task, one agent traces out a path through a second agent's sensory field. The second agent then has to reproduce that path exactly, i.e. move through the sequence of locations visited by the first agent. This is a non-trivial and clearly non-reactive task since the behaving agent's moves are 'reactions' to temporally remote (i.e., in-the-past) states of the environment. The acquisition of path-imitation behaviour might be expected to involve special-purpose (i.e., strongly biased) learning machinery. However, this turns out not to be the case.

The behaviour can be acquired using a fairly primitive learning regime provided that the behavioural environment can be made to pass through a specific sequence of *dynamic* states.

The paper breaks down into four main sections. First, I describe the acquisition experiment performed. Second, I analyze the resulting architecture of the acquisition agent. Third, I review the theoretical background and motivation for the work. (Theoretically oriented readers may prefer to read the third section first.) The fourth and final section is a discussion.

1 The acquisition experiment

The acquisition experiment involved subjecting a simulated mobot (mobile robot) to a sequence of environmental scenarios. Learning accomplished by the mobot in the earlier scenarios had an impact on the character of later scenarios. Thus, the developmental process involved an ongoing *interaction* between learner and environment. The ultimate success of the process relied on there being a tight-coupling between the 'developmental trajectory' (changes in the learner) and the 'environmental trajectory' (changes in the environment). The learner mobot was equipped with two proximity sensors (shown as dashed-lines in Figure 1). Each one of these sensed the proximity of the nearest obstructing surface along a particular ray. Proximity values were normalized in the range 0..1, with lower values indicating lower sensed proximities. All sensor inputs were noisy with noise increasing linearly with measured range. The boundary of the space, shown here as a solid line, was not sensed by the mobot.

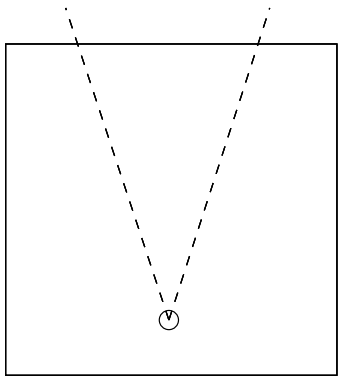

Figure 1: The learner mobot.

2 The acquisition method

For reasons that will become clear, I call the acquisition process used in this experiment 'explicitation'. It is essentially the competitive learning regime of Rumelhart and Zipser (1) with three modifications. The first modification enables the regime to operate incrementally. With this modification, competitive learning adds nodes to the network as and when they are needed. The process comes to an end when

each node is fully tuned to the inputs that it has captured.[1]

The second modification associates a significance value with each weight in the network. In standard competitive learning, each weight is equally significant in the computation of node response and nodes thus have no way to discount particular inputs. As a result there is no straightforward way for them to capture a statistical effect which does not involve all the inputs (2). In the explicitation regime, all weights for all competitive units have an associated significance value whose value reflects the accuracy of the weight as an input-value predictor. Node response is computed taking the significance values into account, and thus weights with low accuracies (significance values) are effectively discounted where appropriate.

The third modification enables the regime to operate recursively. The capture of some effects by a set of competitive nodes triggers a further round of competitive learning in which the input data are *descriptions* of the nodes themselves. For each grouping identified by this process, an internal variable is created and a link made so that the variable's value shows which of the nodes in its group is most active. New data are then derived by presenting the original data and reading off the values of the internal variables. These new data permit the re-application of the entire regime, and thus the production of another 'layer' of internal variables, and so on.

2.1 Initial environment

In the initial scenario, the environment was empty except for the learner itself and a second simulated mobot which moved through the learner's sensor field in a figure-of-eight pattern. The facilitator's trajectory crossed the learner's proximity rays in four different places, see Figure 2. The learning

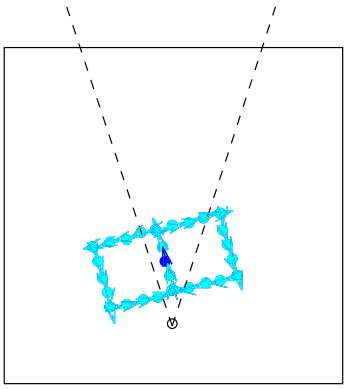

Figure 2: Environment-1.

mechanism was given access to four variables: two of these were pure input variables and were connected to the learner's proximity sensors. The other two operated as hybrid input/output variables and were connected to the motors for the learner's drive system. In the initial environment, the motor variables remained uninstantiated by either the environment or the learner itself. Thus the learner remained entirely static.

[1] A node is considered to be fully tuned when its significance weights (see next paragraph) satisfy either an upper or a lower threshold.

The proximity inputs tended to show different patterns depending whether the facilitator's motion was crossing the right proximity ray or the left. With the facilitator on the left, the left (first) proximity input would oscillate between a high (close) value and a medium (far) value while the right proximity input would show a low, relatively noisy value.[2] With the facilitator on the right the complementary situation would hold.

Application of the learning regime to these input data produced the initial network shown in Figure 3. In this figure the four input variables are represented as four rectangles in the lower part of the figure.

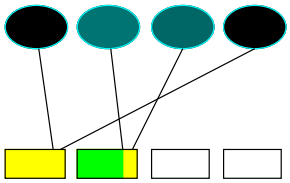

Figure 3: Competitive nodes in net-1.

The competitive nodes are shown as circles in the upper part. The shading of the circles represents the nodes activity level (darker = more active) but in this figure the levels are randomly chosen. Only connections with a high significance value are shown. Connections which intersect an input variable further to the right represent weights with higher values. Connections which intersect further to the left represent lower values. The shading division within the variables shows the current instantiation of the variable. All instantiations were normalized in the range 0..1.

From the figure we see that just four competitive nodes have been created and each one of these has a single, significant weight. Note how the weights have captured all four combinations of near/far, left/right proximity values. Thus there is exactly one node for each prototypical input pattern. Application of competitive learning to (descriptions of) these nodes divides them into two similarity groups and thus creates the first two variables in a new layer. This layer is extended in two ways: first, by raising-up those variables from the original layer which have no connections (i.e., the two motor outputs) and, second, by adding a memory buffer of four variables. The instantiations of these (in any given cycle) are obtained by copying over the previous instantiations of the main variables. The combination of variables and nodes now assembled is called 'subnet-1', see Figure 4. The original input variables form subnet-1's input variables. The internal variables created are subnet-1's output variables. The assembly comprising subnet-1's leftmost two input variables, its four nodes and its leftmost two output variables effectively provides a noise filter. As they pass through the net, the proximity inputs are recoded in terms of four, prototypical proximity values. Thus the leftmost two output variables for subnet-1 behave as prototypical-proximity detectors.

The input data are now recoded using the instantiations of subnet-1's output variables. Application of competitive learning to the recoded data leads to the creation of six more nodes. The input variables for these nodes are subnet-1's output variables. They are shown in a layer above those variables, see Figure 5. Note how all these nodes have connections to the proximity-detector units and to their partners in the memory buffer. Recall that the partner variables hold the instantiations which existed in the main variables in the previous time cycle. The main variables are the prototypical-proximity detectors. Thus these nodes have captured 1-cycle changes in the perceived prototypical proximity of the facilitator. They effectively provide unilateral (i.e., side-specific) detectors for mobot-approach and mobot-retreat.

[2] Noise on the proximity inputs varied inversely with the measured proximity. Thus higher readings (closer proximities) were more reliable than lower readings.

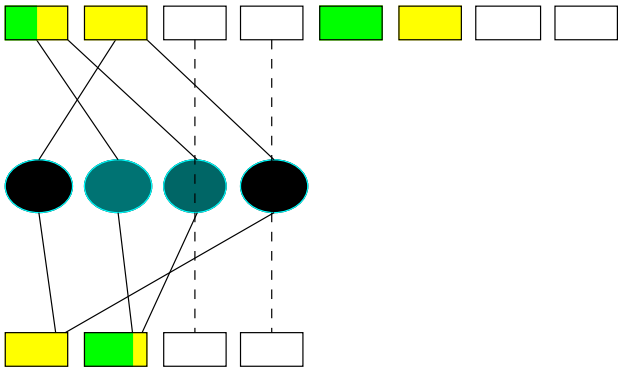

Figure 4: The structure of subnet-1.

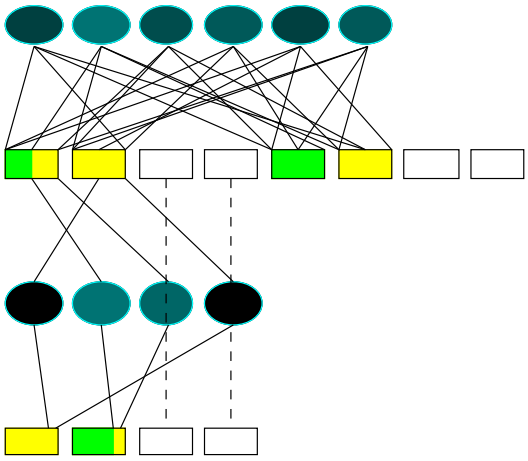

Figure 5: Competitive nodes in subnet-2.

The first node, for example, detects the situation in which the facilitator mobot is moving away from the learner on the left hand side.

When competitive learning is applied to the descriptions of the competitive nodes in subnet-2, two main groupings are recovered. In one group we have the two nodes which detect facilitator motion away (on left and right). In the other group we have the two nodes which detect facilitator motion towards. The two internal variables generated thus form *bilateral* approach and retreat detectors. Each one measures the degree to which the facilitator is moving towards, or away from the learner on *either* side. This subnet appears in the bottom, right corner of Figure 6, which shows the sequence of subnets produced up to this point.

As we will see, these subnet-2 output variables will turn out to play a crucial role in the production of path-imitation behaviour. Their values encode the relative motion of the facilitator and can thus be straightforwardly used to drive the motors of the learner during path replication.

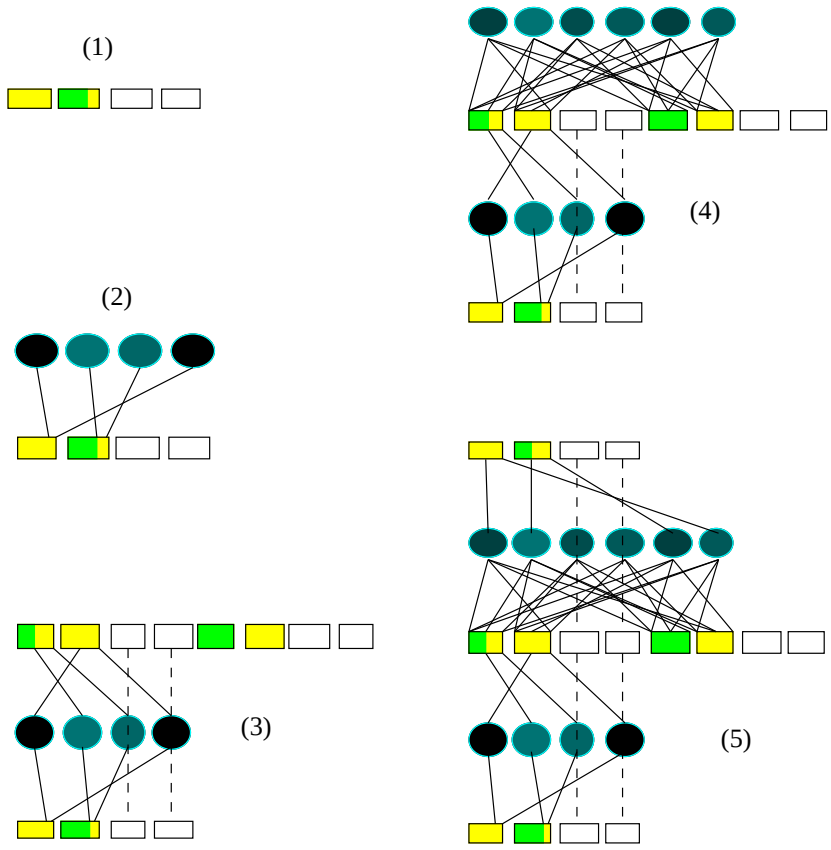

Figure 6: Subnet construction sequence.

2.2 The second environment

In the second environment the facilitator behaves in much the same way as it did in the initial environment — now executing an S-shaped trajectory in the area of space in front of the learner (see Figure 7) — but the learner's motor variables are now instantiated in each cycle. The aim of this is to ensure that the learner tracks the facilitator, thus keeping it 'inside' the learner's sensor rays. This direct instantiation process is, of course, an explicit teaching stimulus provided by the environment. The character of the learning scenario thus shifts from unsupervised to supervised. Application of competitive learning to the new, four-variable input data adds two nodes to subnet-1. These capture the two main patterns of motor-variable instantiation and the proximity inputs they are associated with; i.e., motor outputs for a leftwards move associated with a high proximity input on the left sensor and motor outputs for a rightwards move associated with a high proximity input on the right sensor. The learning process is carried forward no further than this due to the presence of the memory buffer in the output layer for subnet-1 (see below).

As a result of this supervised learning phase, the learner acquires the tendency to fill-in motor-variable instantiations in a particular way. This, in effect, enables the learner to track an object moving across its sensory field. Thus when a third environment is presented in which the facilitator 'demonstrates' a

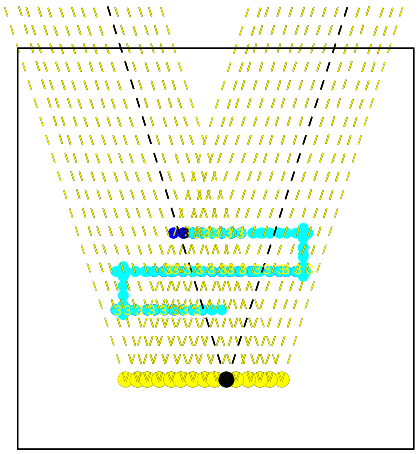

Figure 7: Environment-2.

path, the learner responds by tracking the facilitator across the space, see the upper-left box in Figure 8. As the learner tracks the facilitator across the space, the output variables in subnet-2 are instantiated with values which reflect the relative motion of the facilitator. The bilateral approach/retreat detectors encode the left/right motion of the facilitator. The raised motor variables encode the forwards/backwards motion.

By storing the sequence of instantiations produced in subnet-2's output variables, we thus acquire a sequence of relative-motion descriptions which can be used to regenerate the path executed by the facilitator. Feeding this program (after suitable post-processing) into the learner's motor system, we effectively obtain the desired path-imitation, see the lower two boxes in Figure 8 . In the lower-right box, the facilitator's path is shown using a light dashed line. The learner's imitation of it is shown using a heavier, dashed line.

3 The architecture: a guided tour

In Figure 9 we see the final network architecture produced by the learning process. The various shaded areas correspond to functional components. In the lower, left part of the architecture we have the 'Prototypical proximities detection subsystem'. Recall that this essentially serves to clean-up the relatively noisy proximity inputs. This cleaning-up process is actually an essential part of the overall acquisition process since it paves the way for the development of the unilateral motion detectors which emerge in subnet-2. It is only thanks to the lack of noise in the proximity measures (produced in subnet-1's output variables) that subnet-2's node are able to capture the relevant approach/retreat patterns. In the right-hand part of subnet-1 we have the motor-control subsystem. This is perhaps the simplest component in the entire architecture. The nodes have captured the two main reflexes introduced through the supervisory process of environment-2, namely, move right when right-hand proximity input is high, move left when left-hand proximity input is high.[3]

The column-shaped central region embodies the raised motor-variables. This subsystem emerges due

[3] These effects are not represented clearly since negative motor-values are involved in one of the cases.

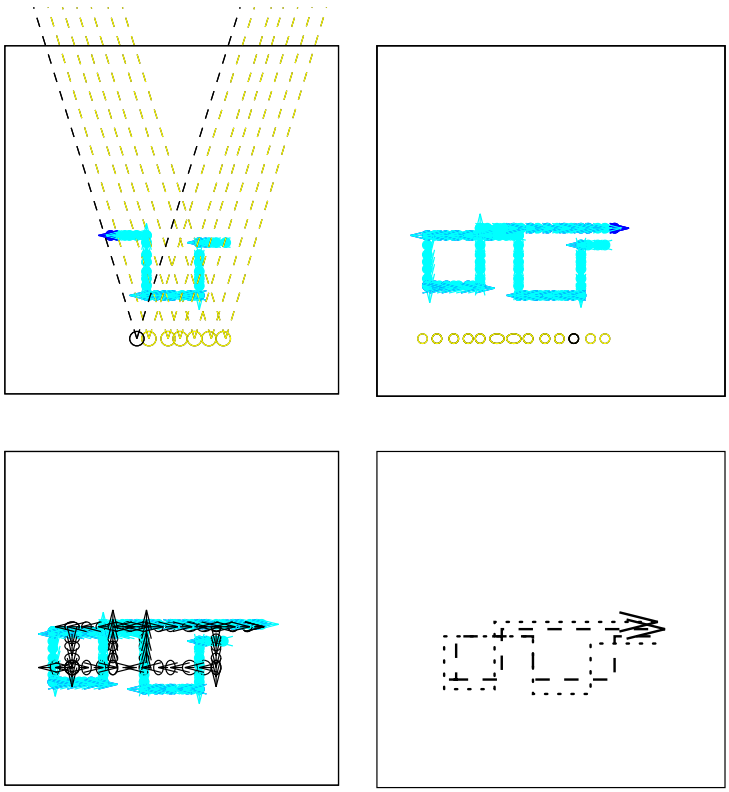

Figure 8: Immitation of a simple path.

to the fact that, throughout the initial development, the motor-variables are uninstantiated and thus remain unconnected from any node. However, the raising-up ('exheriting') process is vital for the final outcome. Following the supervisory process of environment-2, the motor-variables are filled-in by the learner thus enabling tracking moves to be executed. The values that are filled-in are of course accessible at the output layer of subnet-2. Since the learner mimics the left/right motion of the facilitator exactly, these values serve to encode that motion. Taken together with the bilateral motion detectors, these raised motor-variables yield the full encoding of the facilitators relative motion which provides the basis for the imitation behaviour. Note how action *on* the environment and cognizing *of* the environment is wound together here in a seamless interaction.

4 The theoretical background

I now turn attention to the theoretical motivation for the work described above. It is widely accepted that any learning problem can be defined in terms of a (possibly notional) target, input/output mapping. The learner's goal is viewed as the mapping of any input taken from the target mapping onto its associated output. If the learner is to have any chance of achieving this goal, it must receive some sort of 'feedback', i.e., information about what the target mapping contains. In the degenerate case where the feedback provided is actually an explicit listing of the *entire* target mapping, the acquisition task

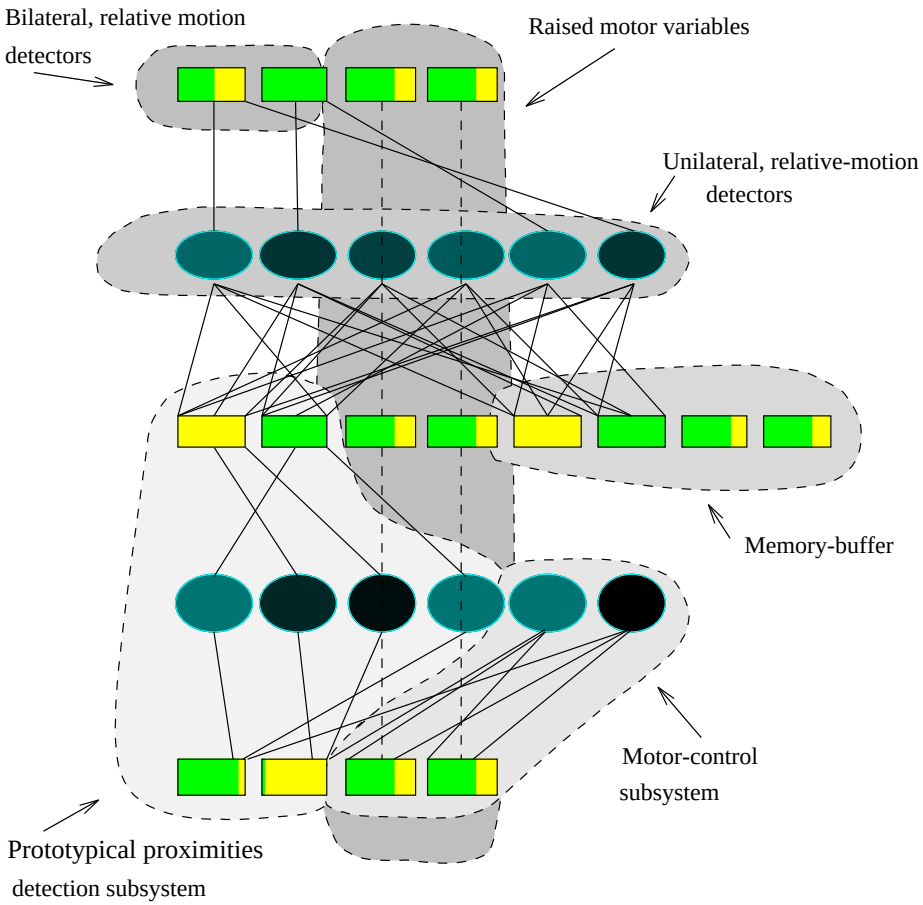

Figure 9: Final network architecture and subsystems.

is reduced to a memory task. In a non-degenerate variant of this case the learner is given access to a part of the mapping and this effectively necessitates generalization. If the access is explicit (i.e., if input/output examples are presented) we have the familiar supervised learning scenario. If access is only obtained implicitly, i.e., via some reward function, then the scenario is a form of reinforcement learning.[4]

Given this formalization of learning, a task analysis can be derived by enumerating the ways in which the learner's output guesses may be *justified* by the feedback source [3]. The feedback is, of course, ultimately derived from the target mapping. Thus guesses that are justified by the feedback must, at a deeper level, be justified by the contents of the target mapping. The guess that y is the correct output for some input x, i.e., that the probability of y is 1 is therefore *only* justified if, in the target mapping, it is the case that

(1) $P(y) = 1$, i.e., the unconditional probability of seeing output y is 1, or if

[4] Typically, reinforcement learning scenarios impose additional constraints on the availability of rewards.

(2) $P(y|x') = 1$, where x' is either the current input or some part of it, or if

(3) $P(y|g(x)) = 1$, where g is some arbitrary function and x is the current input.

This taxonomy is derived simply by enumerating the possible syntactic forms for the justification. It is invariant with respect to the probability value selected (we do not have to use $p = 1$). It is also *exhaustive* since there is no alternative way of characterizing the conditional or unconditional probability of y being the right output for input x. The interesting consequence of this is that any learning method which produces justified output guesses, must exploit (i.e., use in the generation of guesses) some combination of these three forms of justification.

The cash value of this becomes clear when we consider the ways in which each of the three forms can be exploited. Exploiting a justification in this context means 'finding' the probability in a given distribution. Thus the complexity of exploiting a particular justification is related to the size of the relevant distribution. This prompts us to split the justification forms up according to whether the relevant distribution is *finite*. In particular, we must distinguish between what I call the **direct** forms $P(y)$ and $P(y|x)$ which are associated with finite probability distributions, and the **indirect** form $P(y|g(x))$ which is associated with an infinite one. The distribution $P(y|g(x))$ is infinite due to the infinite number of choices to be made regarding g, which is defined as any computable function.

This analysis of justification sources leads directly to a fundamental insight concerning the complexity of learning problems. Problems which involve exploiting either of the two direct forms involve the equivalent of sampling a finite distribution while problems which involve exploiting the indirect form involve the equivalent of sampling an infinite distribution. Other things being equal, the task of exploiting direct forms thus has a *lower* theoretical complexity than the task of exploiting the indirect form. Note that this is a qualitative distinction similar to the one between polynomial and exponential time complexity. However, it has a firm, mathematical basis in the enumeration of syntactic forms for probability values.

4.1 Statistical v. relational problems

It is important to note that values of the function g (which I call the **recoding function** below) must depend on *relative* argument values. In other words, the function must compute or evaluate a relational property of its arguments. Were we to have an indirect justification in which values of the recoding function depended on the *absolute* values of its arguments, then we could specify the same justification purely in terms of those absolute values, i.e., by deleting the g and associated parentheses. Thus, all indirect-form justifications necessarily involve functions which effectively compute relationships of their arguments.

In recognition of this, I call the class of higher-complexity problems which involve exploiting indirect justifications **relational** and the class of lower-complexity, direct-justification problems **statistical** (since exploiting a direct-form justification involves sampling for a statistical effect in the form of an observed probability). If it was not obvious before, the introduction of this new terminology should drive home the point that this analysis gives a foundation to the well-known Machine Learning heuristic which states that 'learning relationships is hard' [4].

5 Example

To illustrate this notion of indirect justification, consider the following training set. This is based on two input variables (x1 and x2) and one output variable (y1). There are six training examples in all. An arrow separates the input part of the example from the output part.

```
x1  x2      y1

1   2  -->  1
2   2  -->  0
3   2  -->  1
3   1  -->  0
2   1  -->  1
1   1  -->  0
```

A variety of direct justifications are to be found in these training data. For example, we have the unconditional probability $P(y1 = 1) = 0.5$, and the conditional probability $P(y1 = 1|x1 = 2) = 0.67$. These probabilities, and in fact all the probabilities directly observed (i.e., obtainable by sampling) in these data, turn out to be close to their chance values. Indirect justifications are to be found via some function g. In the case at hand imagine that the function effectively substitutes the input variables in each training pair with a single variable whose value is just the difference between the original variables. This gives us a set of derived pairs as shown below (the value of x4 here is the difference between the values of x1 and x2).

```
Original pairs              Derived pairs (x4 = |x1-x2|)

x1  x2      y1              x4      y1

1   2  -->  1              1  -->  1
2   2  -->  0              0  -->  0
3   2  -->  1              1  -->  1
3   1  -->  0              2  -->  0
2   1  -->  1              1  -->  1
1   1  -->  0              0  -->  0
```

Note how the recoding has produced data in which we observe a number of extreme probabilities relating to the output variable y1, namely P(y1=0|x4=0)=1, P(y1=1|x4=1)=1 and P(p1=0|x4=2)=1. The recoding thus provides us with indirect justification for predicting y1=0 with a probability of 1, if the difference between the input variables is 1. It also provides us with indirect justification for predicting y1=1 with a probability of 1, if the difference between the input variables is either 2 or 0. In short, we have indirect justification for the output rule 'y1=1 if x4=1; otherwise y1=0'.

The task analysis has implications for the way in which we carry out behaviour acquisition research. We know that acquisition problems will divide up into statistical and relational types, and that the latter will tend to be harder than the former. Being able to tell the difference between statistical and relational tasks is thus an important goal. To identify a task as statistical we have to show that the underlying input/output mapping yields direct-form justifications, i.e., exhibits correlations between input and output variables. Thus, to identify a task as statistical we should attempt to analyze the correlation structure of its associated input/output mapping.

In some cases, the absolute size of the relevant mapping may make this process rather long-winded. However, the presence of strong input/output correlations can often be demonstrated using qualitative reasoning. As a general rule of thumb a task is likely to be statistical (i.e., to exhibit strong correlation structure in its input/output mapping) if it can be implemented using a reactive (i.e., reflex-based) system. If the behaviour can be implemented on the basis of reflex reactions to direct stimuli, then any given input (stimulus) must contain clear cues as to which output (response) is appropriate. If these cues are unambiguous — which they must be — then they will tend to co-occur with the relevant responses in the target mapping and thus naturally yield a strong correlation effect.

5.1 Use of alternative acquisition methods

The acquisition of path-imitation is an interesting topic for investigation since it is a simple but manifestly non-statistical task. The fact that the target behaviour has to be produced after the entire path-presentation process is finished implies that it could not possibly have a straightforward reactive implementation. And even if we leave the behavioural delay on one side, a reactive approach seems out of the question since the learner must be able to observe the path-tracing agent from any angle, meaning that the production of a path-replicating action by the learner is essentially a reaction to a 'hidden state' of the environment, i.e., a relative motion within the path's coordinate system.

Attempting to solve this acquisition problem using conventional learning methods such as C4.5 (5,6) and backpropagation (7) produces no useful results (application of 'windowing' notwithstanding). Such methods work with a single set of exemplars taken from the target. Thus, in order to present the path-imitation problem it is necessary to concatenate together all the relevant inputs from all the various environments and to map each one onto an associated 'target output', which is the entire control program for the relevant path. Given the rather baroque nature of this setup, it is perhaps not altogether surprising that conventional methods do not produce any performance on this problem.

The path-imitation experiment performed demonstrates that explicitation is, in at least one case, capable of relational learning. But its generality on relational tasks is still not fully established. On the most optimistic view, the prototype-recruitment process, which is the core of the method, will turn out to be a universal method for relational acquisition, i.e., one which is capable, given sufficient iterations, of capturing any relational effect whatsoever. On the least optimistic view, this method will turn out to be applicable to path-imitation only.

6 Discussion

In this final section I will briefly summarize some of the advantages and disadvantages of the proposed acquisition model. A principal disadvantage is the fact that the path-imitation behaviour produced by the acquisition model is actually not particularly accurate. The relative-motion encoding provided by the output variables of subnet-2 are only capable of reflecting motion towards or away from the learner. Thus in replicating the path the learner is only able to align itself on one of two different 'latitudes'. The scale of the path is also only very weakly fixed within the model. The path size is effectively derived from the number of times the learner is 'nudged' left or right during path-observation. This of course varies according to the orientation of the proximity rays. The stage-management of the path-replication process introduced fortuitous assumptions regarding the relationship between the number of nudges and the amount of space to be traversed for each nudge.

A second disadvantage involves the fairly arbitrary blocking of recursive learning following presentation

of environment-2. This is achieved in the model via a rule which states that input vectors cannot be extended once memory variables are in play. This rule preempts the lengthy task of restructuring the data structures representing the input and output buffers of the various nets. It also has the happy consequence of preventing further, environment-2 based learning which, in the present case, would not be of advantage. However, it cannot be claimed as a principled part of the model, especially in view of the fact that the selection of memory-size in the first place is essentially arbitrary.

Finally, there is the worrisome fact that the path-imitation control program is collected up and delivered in an almost entirely arbitrary way. The values of the final output variables are stored in a separate memory resource and then re-processed. The learner is then forcibly moved to the starting position for the path and the control program is 'executed' (i.e., the learner's motor-variables are repeatedly instantiated with values derived from the control program). This aspect of the model is most unsatisfactory but no obvious remedy presents itself at the present time.

Principal among the *advantages* of the model has to be the fact that it shows how a hard relational acquisition task can be accomplished using a very weakly biased, theoretically well-motivated mechanism. Although the model in its present form is not particularly plausible as a model of a neural process, no part of it would seem to be beyond a neural interpretation. The basic competitive process has a well known neural interpretation (8) and the creation and management of internal variables can almost certainly be accounted for in terms of the creation of small subnets.

One of the interesting features of the 'explicitation net' is that it exhibits two distinct types of process. Firstly we have the phase in which activation flows forwards through the networks. In this phase the activation of competitive nodes creates the instantiation of output variables and thus further inputs creating further activations etc. Following this, input values may be filled-in (e.g., in the case where a significant input for a strongly active node remains uninstantiated) and this provides the potential for a backwards propagation of activity. Thus, the architecture responds to the presentation of a new, partially complete input by effectively *assimilating* it to those *structural* and statistical properties of the environment which it has already captured.

In the learning phase, additional competitive nodes, internal variables and subnetworks are generated on the fly. These seek to account for statistical and possibly structural and relational properties of the environment that are not currently captured by the network. In this way the network seeks to *accommodate* new features of its environment. This process, in which the network oscillates between an assimilation process and an accommodation process, may have explanatory potential for the Paigetian picture of development (9). Of course, the nature of the accommodation process is well viewed in terms of the development of new recodings at various levels of description. Thus we might also seek to make a link with Karmiloff-Smith's theories of representational redescription (10,11,12).

Developmental psychologists might perceive a link between the acquisition model and the process known as 'scaffolding'. In the model, the path-imitation behaviour is acquired by subjecting the learner to a *stage-managed* sequence of environments. The general scenario would seem to be an acceptable approximation of the way in which scaffolding is thought to occur in human contexts. As Rutkowska says, 'Scaffolding involves more able humans manipulating the infant's transactions with the environment so as to foster novel abilities. ... It can be thought of as a form of supervised learning that exploits temporarily engineered emergence of function to support permanent adaptive change' (13).

Philosophers may also be able to extract something of interest. The constructivist nature of the explicitation process offers a detailed picture of the way in which sensory stimuli might provide the underpinnings for higher-level, weakly representational[5] structure. It also fills in some of the details

[5] The word 'representational' is used in the following sense: R represents property P of environment E for agent A if R constitutes a measure of property P and values of the measure play a functional role in the production of A's behaviour

regarding the way in which an environment might drive a cognizer from one cognitive level 'up' to the other. The implication may be that the hard-and-fast distinction we make between classical, symbolic accounts of cognition and connectionist, neural accounts may not be quite so uncrossable as it seems.

References

[1] Rumelhart, D. and Zipser, D. (1985). Feature discovery by competitive learning. *Cognitive Science*, No. 9 (pp. 75-112). ((N-R)).

[2] Thornton, C. (1994). Unsupervised learning with the soft-means algorithm. *Proceedings of the World Congress on Neural Networks*. Vol. IV (pp. 20-205). San Diego.

[3] Thornton, C. (Forthcoming). *Unsupervised Constructive Learning*.

[4] Dietterich, T., London, B., Clarkson, K. and Dromey, G. (1982). Learning and inductive inference. In P. Cohen and E. Feigenbaum (Eds.), *The Handbook of Artificial Intelligence: Vol III*. Los Altos: Kaufmann.

[5] Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, *1* (pp. 81-106).

[6] Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. San Mateo, California: Morgan Kaufmann.

[7] Rumelhart, D., Hinton, G. and Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, *323* (pp. 533-6).

[8] Kohonen, T. (1989). Speech recognition based on topology-preserving neural maps. In I. Aleksander (Ed.), *Neural Computing Architectures* (pp. 26-40). Cambridge, Mass.: MIT Press.

[9] Boden, M. (1979). *Piaget*. Fontana Modern Masters, Fontana Press.

[10] Karmiloff-Smith, A. (1990). Constraints on representational change: evidence from children's drawing. *Cognition*, *34* (pp. 57-83).

[11] Karmiloff-Smith, A. (1992). *Beyond modularity: a developmental perspective on Cognitive Science*. Cambridge,Ma.: MIT Press/Bradford books.

[12] Karmiloff-Smith, A. and Clark, A. (1993). What's special about the development of the human mind/brain?. *Mind and Language*, *8*, No. 4 (pp. 569-581).

[13] Rutkowska, J. (1994). Emergent functionality in human infants. In D. Cliff, P. Husbands, J. Meyer and S.W. Wilson (Eds.), *Proceedings of the Third International Conference on Simulation of Adaptive Behaviour* (SAB-94) (pp. 179-188). Brighton, UK.

but are not statistically correlated with any sensory input to A.