

Learning about Objects and Activities

David Hogg¹, Anthony G. Cohn¹, Vincent Devin², Derek Magee¹,
Chris Needham¹, and Paulo Santos³

¹ School of Computing, University of Leeds

² France Telecom R&D, Meylan, France

³ Centro Universatario da FEI, Sao Paulo, Brazil

Introduction

Statistical learning has become a central theme in computational studies of intelligence, with important but largely independent developments in perceptual tasks like object recognition and in conceptual reasoning (Mitchell (1997)). Typically, the former involves clustering and regression over quantitative representations based on real linear spaces (\mathcal{R}^n), whereas the latter involves symbolic procedures over qualitative representations that emphasise the relationships between objects (e.g. *next-to*, *occurs-before*, *overlaps*, *similar-shape*). Integrating learning across these two levels of abstraction presents a number of challenges, particularly in relating quantitative and qualitative representations.

We propose a system for integrating standard approaches to perceptual and conceptual learning within the simple domain of table-top games (full details will appear in Needham *et al.* (2005)). Our target scenario is of an intelligent agent being introduced into an unfamiliar environment from which it must learn to distinguish between the game-objects (e.g. playing cards, die) and also learn how to play the game - corresponding to the perceptual and conceptual levels respectively. In an off-line *learning phase*, the agent passively observes two players engaged in playing the game from a single camera looking down onto the table (the *training data*). In a subsequent *execution phase*, the synthetic agent substitutes for one of the players.

A variation of the familiar game of snap will be used by way of illustration - we call this 'GSnap'. The game is for two players who are each dealt cards containing a variety of simple shapes in different colours. In repeated plays, each player lays a card on the table simultaneously. If the cards have the same shape and colour, a player says the word same. Otherwise, if only the colour or the shape are the same, they say the word "colour" or "shape" as appropriate. If the table is empty, a player says the word "play".

The System

Both learning and execution phases start with an identical attentional process in which the raw visual data is segmented into objects using a generic tracker (Magee (2004)) and the auditory waveform is segmented into sounds by thresholding. These correspond to the objects (e.g. playing cards) and spoken utterances (e.g. "same") of the game.

Many table-top games are inherently cyclic, consisting of a series of plays. We capture the necessary information from each cycle by focusing attention on the stationary periods between the movements of objects over the table-top (in GSnap this happens each time cards are played onto the table).

The learning phase now occurs in two stages (Figure 1). In the first stage, feature measurements taken from the visual objects (when stationary) are clustered to produce attribute categories of colour, shape and position (Needham *et al.* (2005)). Similarly, a fixed number of sound categories are produced by clustering a spectral analysis of the segmented sounds.

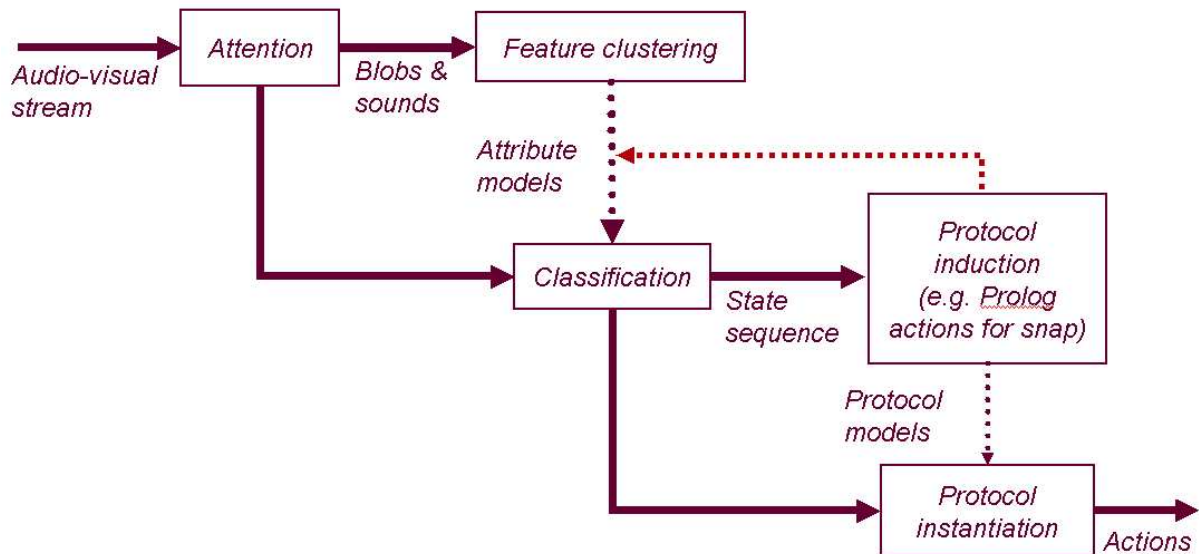


Figure 1: The two stages of the learning phase, followed by the execution phase.

In the second stage of learning, objects and sounds from the training data are first classified according to the learnt categories, using the nearest-neighbour algorithm (Webb (1999)). The result is a time series of qualitative states, each describing the objects on the table-top and any utterance made during a stationary period. The *qualitative states* are represented as sets of logical atoms. Figure 2 shows a series of five qualitative states from a game of GSnap. Each contains a ‘state’ predicate giving the texture, colour and position categories assigned to each visible object, an ‘action’ predicate giving the category of the spoken utterance, and ‘time’ and ‘successor’ predicates defining a sequential ordering of the time-stamps associated with each qualitative state.

Logical induction is now used to generalise the relationship between utterances (‘action’ predicate), and the attributes of objects appearing on the table in the current and previous qualitative states (‘state’ and ‘successor’ predicates). This generalisation is used in the execution phase to generate appropriate utterances in place of one of the players.

For induction, we use the Progol inductive logic programming system (Muggleton and Firth (2001)) to induce a simple Prolog program H which when combined with the ‘state’ and ‘successor’ atoms (from the training data) B , entails (most of) the ‘action’ atoms E .

$$B \wedge H \models E$$

A Prolog program consists of an ordered set of logical formulae of the form $p, q, \dots, r \Rightarrow s$ (known as Horn clauses, and represented in Prolog as $s :- p, q, \dots, r.$). To constrain the search for H , the predicates that can appear in the head and tail of Horn clauses (i.e. to the left and right of the implication respectively) must be specified in advance. Progol’s task is therefore to configure the predicates and variables within a set of Horn clauses obeying this constraint. The resulting program should both be simple and deal correctly with the training data meaning

that the appropriate actions are generated in each situation. These criteria are captured in a cost function that combines (with other factors) the number of examples dealt with correctly, with the negated length of the Horn clause (number of literals). Progol adds as many Horn clauses to the program as necessary to deal with all of the training data.

From a game of GSnap lasting for 50 plays (illustrated in Figure 2), Progol induced the following set of rules (re-ordered from the most to the least specific):

- (1) `action(utt4,t65).`
- (2) `action(utt5,t198).`
- (3) `action(utt1,t237).`
- (4) `action(utt6,A) :- state([[B,C,D],[B,C,E]],A).`
- (5) `action(utt10,A) :- state([[B,C,D],[E,C,F]],A).`
- (6) `action(utt1,A) :- state([[B,C,D],[B,E,F]],A).`
- (7) `action(utt4,A) :- state([[B,C,D],[E,F,G]],A).`
- (8) `action(utt9,A) :- state([],A).`
- (9) `action(utt8,A) :- state([],A).`
- (10) `action(utt2,A) :- state([],A).`
- (11) `action(utt3,A) :- state([],A).`
- (12) `action(utt7,A) :- successor(B,A), state([[C,D,E],[F,G,H]],B).`

The mapping from constants to utterances was as follows:

utt6 = “same”, utt10 = “colour”, utt1 = “shape”, utt4 = “nothing”
 utt2, utt3, utt5, utt7, utt8, utt9 = “play”.

Clauses 1-3 and 12 are the result of isolated ‘noise’ events. These clauses do not cause problems in the execution phase. Clauses 4-7 implement the essential logic of GSnap through judicious use of variable identity. Clauses 8-11 implement the same conceptual action (to utter the sound “play”). They arise because the “play” sounds from the training data have been clustered into six separate categories. Noticing the identical form of the right-hand sides of these clauses, we are able to identify automatically the six sounds as equivalent and simplify the program accordingly. In principle it should be possible to update the sound category model to merge the six categories into one.

In the execution phase, our synthetic agent substitutes for the voice of one of the players - both original players then remain silent. As in the second phase of learning, the attention and classification procedures produce a stream of qualitative states. The learnt Prolog program is then used to query the action that should be taken following each play. In our experiments on a variety of games (including paper-scissors-stone), the logic of the game is normally obtained exactly, with errors arising only through misclassification of game-objects during execution - this can arise from changes in lighting for example.

Conclusions

The proposed integration of perceptual and conceptual modes of learning has provided grounding for the logical terms appearing in induced activities. More unexpectedly, we have found that object and sound categories may themselves be refined by their emergent role within these induced activities. The notion of a time series of qualitative states has provided a workable interface between quantitative and qualitative modes of learning and execution for a range of simple table-top games.

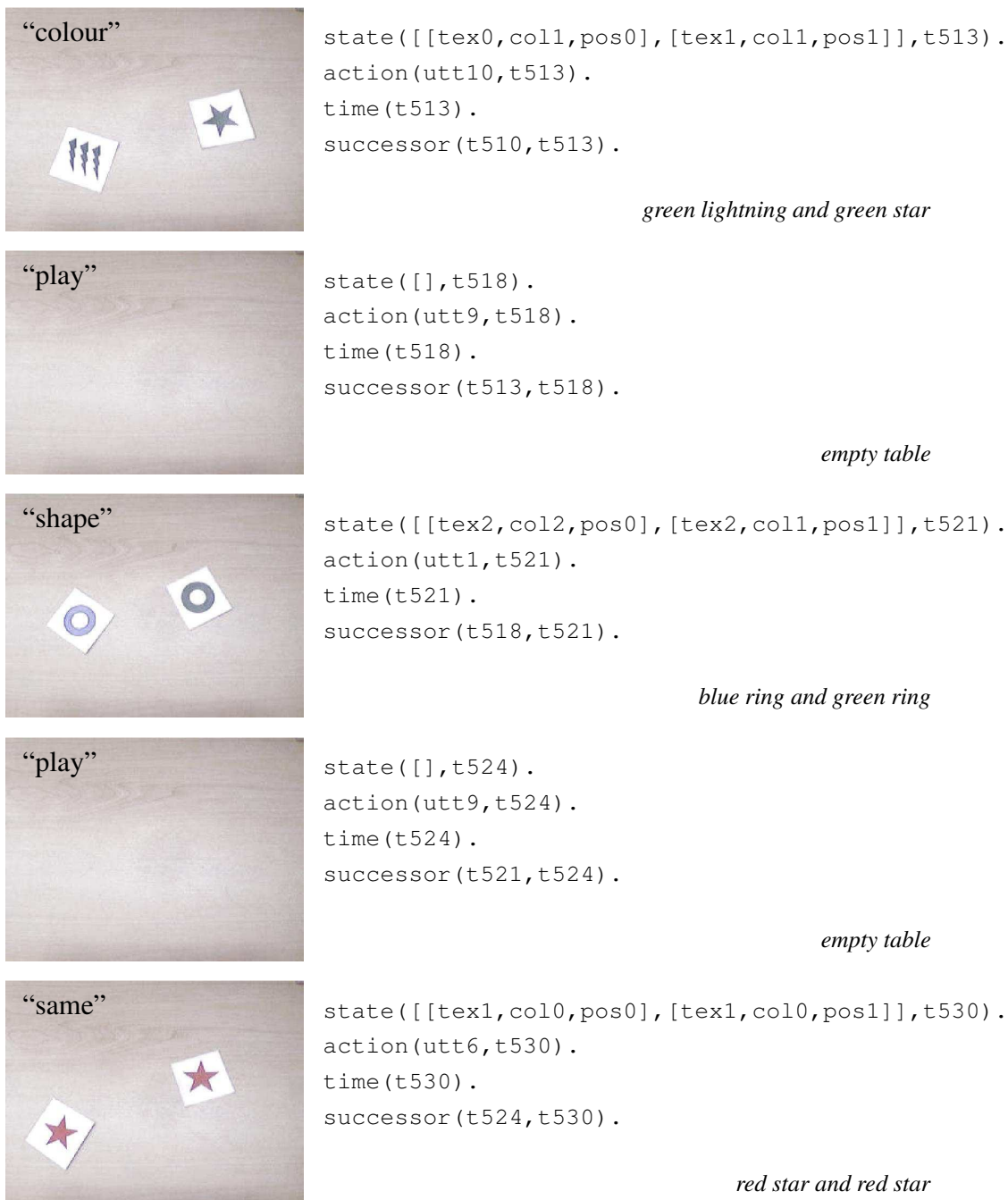


Figure 2: A series of qualitative states from the game of GSnap.

References

- Magee D.R. Tracking multiple vehicles using foreground, background and motion models. *Image and Vision Computing*, **Vol 22**, 143–155 (2004).
- Mitchell T. *Machine Learning*. McGraw-Hill, New York; London (1997).
- Muggleton S. and Firth J. CProlog4.4: a tutorial introduction. S. Dzeroski and N. Lavrac, editors, *Relational Data Mining*, 160–188. Springer-Verlag (2001).
- Needham C.J., Santos P.E., Magee D.R., Devin V., Hogg D.C. and Cohn A.G. Protocols from perceptual obseravtions. *Artificial Intelligence*, to appear (2005).
- Webb A. *Statistical Pattern Recognition*. Arnold, London (1999).