A MULTISTRATEGY APPROACH TO LEARNING CONTROL FROM HUMAN SKILLS

Claude Sammut

School of Computer Science and Engineering
University of New South Wales

SYDNEY AUSTRALIA 2052

claude@cse.unsw.edu.au
http://www.cse.unsw.edu.au/~claude

*A*bstract: Behavioural cloning seeks to build models of human skill by using observations of the performance of a task as input to a machine learning program. Experiments in this area have met with some degree of success, but the methods employed to date have their limitations, including brittleness and opaqueness of the control rules. Some of these limitations are due to the simple situation-action model of control generally used and some or due to restrictions imposed by the language used to represent the rules. Rather than using a situation-action model of control, it is often more useful to have a goal-oriented model in which a set of rules is used to establish goals and the control rules instigate actions to achieve those goals. Furthermore, the learning systems presently used are not capable of representing high-level concepts that would allow controllers to be simpler and more robust.

Keywords: Artificial Intelligence, Flight Control, Knowledge Acquisition, Knowledge-based Control, Machine Learning, Real-time AI

1. MOTIVATION

Behavioural cloning seeks to build models of human skill by using observations of the performance of a task as input to a machine learning program. Experiments in this area have met with some degree of success. Michie et al (1990) first demonstrated the method by recording the traces of human operators of a pole balancing simulation. The traces were input to a machine learning program which produced rules capable of controlling the pole and cart. A demonstration on a larger scale was achieved by Sammut et al (1992) who built a control system for an aircraft in a flight simulator by recording the actions of pilots during flight. The automatic controller was capable of take-off, climbing, turning and landing. Urbančič and Bratko (1994) used a similar approach to build controllers for the highly non-linear container crane problem. However, the methods employed to date have their limitations, including brittleness and opaqueness of the control rules (Urbančič et al, 1996). Some of these limitations are due to the simple situation-action model of control generally used and some or due to restrictions imposed by the language used to represent the rules.

To establish the framework for the following discussion, the flight control problem is first presented. A simple flight simulator, as can be easily found on a workstation, is instrumented so that it records the state of the simulation every time the "pilot" performs an action such as moving the joy

stick, increasing or decreasing the throttle or flaps, etc. The action is treated as a label, or class value, for each record. Thus, the records may be used as examples of what action to perform in a given situation.

Decision tree induction programs such as Quinlan's C4.5 (Quinlan, 1993) can process the examples to produce rules capable of controlling the aircraft. These are "situatioin-action" rules since they simply react to the current state of the aircraft by responding with the action that, according the data, was most typical for that state.

More than one decision tree is needed to control a complex system such as an aircraft. In Sammut et al's experiments, each control action (elevators, ailerons, throttle, flaps) were controlled by seperate decision trees. Each was induced by distinguishing one action as the dependent variable and repeating the process for each action. Furthermore, the flight was divided into stages, so that different sets of decision trees were constructed for take-off, turning, approaching the runway, and landing.

Decision trees are classed as p*ropositional* learning systems because the language used to represent the learned concepts is equivalent to propositional logic. Most behavioural cloning work, to date, has been limited to the use of propositional learning tools or neural nets. The problem with these methods is that the language of the control rules is limited to describing simple conditions on the raw attributes output by the plant being controlled. However, control rules may be better expressed using high-level attributes. For example, in piloting an aircraft, it is more illuminating to talk about the plane following a particular trajectory rather than to simply give its coordinates, orientation and velocities at a particular point in time.

The situation-action model of control is also simplistic. A human pilot would normally only resort to this kind of control for low-level tasks such as routine adjustments or during emergencies where there is insufficient time to think. A more robust controller would incorporate the goal-oriented nature of most control tasks. One model of goal-oriented behaviour is one a set of rules is used to establish goals and another set of control rules instigates actions to achieve those goals.

The following section, describes current research in the use of richer languages for describing control strategies. Section 3 elaborates on the architecture of a goal-oriented control system and section 4 concludes with a discussion of future research directions.

2. MULTISTRATEGY LEARNING AND INDUCTIVE LOGIC PROGRAMMING

Srinivasan and Camacho (in press) describe a method of learning high-level features for control. Their task is to learn the relationship between the roll angle and turn radius when turning an aircraft in a flight simulator. They collected data from a number of turns performed by a human "pilot". These data were input to an inductive logic programming system, Prolog (Muggleton, 1995). Inductive logic programming (ILP) systems are able to take advantage of background knowledge provided prior to learning. In this case, the background knowledge includes concepts describing geometric objects such as circles and a regression program that is able to estimate parameters in linear relationships. Thus, Srinivasan and Camacho show that it is possible to recognise the data from a turn as following a roughly circular trajectory and the regression program finds the coefficients of the linear equation which approximates the relationship between the roll angle and the radius of the turn.

In our work, we employ a new inductive logic programming system, called i*Prolog,* which generalises the approach taken by Srinivasan in his extension to Progol. i*Prolog* is an interpreter for the programming language, Prolog, that has been extended by the addition of a variety of machine learning tools, including decision trees, regression trees, back propagation and other algorithms whose output is equivalent to propositional logic. Like other ILP systems, i*Prolog* also provides tools for learning first-order representations that are equivalent to Prolog programs.

ILP systems encode background knowledge as Prolog programs. Thus, if a human experts understands that certain raw attributes measured from a plant can be combined into a more useful higher-level feature, this combination can be expressed in Prolog and the ILP system uses the program to "pre-process" the data so that the learner can incorporate the high-level features into its concept description. Since the ILP system also generates Prolog programs, the learned concepts can become background knowledge for future learning.

Of all the learning paradigms presently in use, ILP provides the most effective means of making use of background knowledge. The first-order language also allows the learner to represent concepts that are much more complex than anything that can be learned by a propositional system. However, these advantages come at a price. A richer language almost always implies a more expensive search for the target concept. ILP systems are also excellent for learning

concepts that can be learned symbolically, but are not so capable when the concepts are numeric in nature.

To overcome these limitations, iProlog incorporates other the other kinds of learning, mentioned previously. Although, these algorithms only generate propositional representations, they can also be expressed as clauses in a Prolog program. Thus, if it is decided that a neural net, or regression, is best suited for processing the numerical data, these can be provided as background knowledge for the ILP system.

Srinivasan augmented Progol's background knowledge with a regression algorithm and geometrical descriptions of trajectories. The result was a concept such as:

```
roll_angle(Radius, Angle) :-
        pos(P1, T1), pos(P2, T2), pos(P3, T3),
        before(T1, T2), before (T2, T3),
        circle(P1, P2, P3, _, _, Radius),
        linear(Angle, Radius, 0.043, -19.442).
```

where P*1* is the position of the aircraft at time T*1* and so on. The circle predicate recognises that P*1,* P*2* and P*3* fit a circle of radius, R*adius* and regression finds a linear approximation for the relationship between R*adius* and A*ngle* which is:

$$Angle = 0.043 \times Radius - 19.442$$

The '_' arguments for circle are "don't cares" which indicate that, for this problem, the centre of the circle is irrelevant.

Thus, ILP provides a framework for linking different types of data fitting algorithms to achieve results that, individually, none are capable of. i*Prolog* is a generalisation of Srinivasan's approach that facilitates the addition of complex background knowledge, including other kinds of learning algorithms (Sammut, in press).

The use of high-level features is particularly important in control systems. It is clear that pilots do not simply react, instantaneously, to a situation. They have concepts such as climbs, turns, glide-slopes, etc. None of these concepts can be adequately captured by learning systems that do not take advantage of background knowledge. High-level features may be pre-programmed, but flexible use of background knowledge allows the learning system to search for the most appropriate features.

High-level features permit structuring of a problem in a way that is conducive to a better solution. Another form of structuring is to move away from a simple situation-action model of control to a goal-oriented one. We have already noted that the flight control problem was decomposed into different stages and within each stage, actions had their own controllers. A further decomposition is to separate the learning of goals from the learning of actions to achieve those goals.

3. GOAL-ORIENTED BEHAVIOURAL CLONING

Behavioural clones often lack robustness because they adopt a situation-action model of control. The problem is that if the state space is very large then a large number of control rules may be required to cover the space. This problem is compounded if the representation language is also limited, as described in the previous section. Bain and Sammut (in press) describe a goal-oriented approach to building control rules.

The data from a flight are input to a learning program whose task is to identify the effects of control actions on the state variables. Another program is used to learn to predict the desired values of the state variables during different stages of a flight. To control the aircraft, we employ simple goal regression where we use the rules learned in the second stage to predict the goal values of the state variables and then use the rules from the first stage of learning to select the actions that will achieve those goal values.

An example of an "effects" rule is shown below:

Elevators = -0.28 → ElevationSpeed = 3

Elevators = -0.19 → ElevationSpeed = 1

Elevators = 0.0 → ElevationSpeed = 0

Elevators = 0.9 → ElevationSpeed = -1

This describes the effect of an elevator action on the elevation speed of the aircraft. A "goal" rules may be as follows:

Distance > -4007 → GoalElevation = 0

Height > 1998 → GoalElevation = 20

Height > 1918 → GoalElevation = 40

Height > 67 → GoalElevation = 100

Distance <= -4153 → GoalElevation = 40

else → GoalElevation = 20

At present, the goals represented in this system are still very simple. In this example, they indicate that at some point in the flight, given by the altitude or distance from the runway, the elevation of the aircraft should be as shown.

The control algorithm uses "goal" rules to determine the correct settings for the goal variables. If there is a difference between the goal and the current value then an effects rule is selected to achieve the desired goal value. The selection is based in the direction and magnitude of the desired change.

Presently, the choice of goal and effects variables is done by hand. A goal of future research is to have this selection done automatically. However, the present method results in more robust and more transparent control rules than those obtained following the situation action-model.

4. CONCLUSION

Research in behavioural cloning has demonstrated that it is possible to build control rules by applying machine learning methods to the behavioural traces of human operators. However, to become a practical tool for control or for instruction, behavioural cloning must address the issues of robustness and readability of the control rules.

In this paper, it has been suggested that these problems can be overcome by employing learning methods that provide greater structuring over the domain. Structure can be imposed by breaking the learning task down to learning goals and learning actions to achieve goals. Further structuring can be achieved by constructing high-level features that permit more compact representations.

Research in goal-directed learning and in ILP suggest that these aims are achievable. However, there are several problems that must be overcome. These are mainly due to an over-dependence in human assistance.

Presently, goal and effects variables must be provided before hand. When these are known, then clearly, it is sensible to take advantage of such knowledge. However, when the distinction is not known, it is important that the learning system be capable of distinguishing these variables for itself.

ILP systems that provide the ability to use background knowledge generally require the user to impose some restriction on which background knowledge should be tried. In principle, the ILP system can search the space of possibilities for itself, but this is very expensive without user provided constraints. Again, if such constraints are known, then they should be used, but if they are not, more intelligent search is required to make the application of background knowledge less onerous for the user.

REFERENCES

Bain, M., and Sammut, C. (in press). A framework for behavioural cloning. In S. Muggleton, K. Furakawa, & D. Michie (Eds.), M*achine Intelligence 15*. Oxford University Press.

Michie, D., Bain, M., and Hayes-Michie, J. E. (1990). Cognitive models from subcognitive skills. In M. Grimble, S. McGhee, & P. Mowforth (Eds.), K*nowledge-base Systems in Industrial Control*. Peter Peregrinus.

Muggleton, S. (1995). Inverse Entailment and Progol. N*ew generation Computing,* 1**3,** 245-286.

Quinlan, J. R. (1993). C*4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Sammut, C., Hurst, S., Kedzier, D., and Michie, D. (1992). Learning to Fly. In D. Sleeman & P. Edwards (Ed.), P*roceedings of the Ninth International Conference on Machine Learning,* Aberdeen: Morgan Kaufmann.

Sammut, C. (in press). Using background knowledge to build multistrategy learners. M*achine Learning Journal.*

Srinivasan, A and Camacho, R. Numerical Reasoning in ILP. In S. Muggleton, K. Furukawa & D. Michie (Ed.), M*achine Intelligence 15*. Oxford University Press. Forthcoming.

Urbančič, T., and Bratko, I. (1994). Reconstructing Human Skill with Machine Learning. In A. Cohn (Ed.), P*roceedings of the 11th European Conference on Artificial Intelligence,* John Wiley & Sons.

Urbančič, T., Bratko, I., and Sammut, C. (1996). Learning models of control skills: phenomena, results and problems. In 1*3th IFAC World Congress,* San Francisco.