Automatically Constructing Control Systems by Observing Human Behaviour

Claude Sammut

School of Computer Science and Engineering
University of New South Wales
Sydney, Australia
claude@spectrum.cs.unsw.oz.au

Abstract

We describe experiments to devise machine learning methods for the construction of control systems by observing how humans perform control tasks. The present technique uses a propositional learning system to discover rules for flying an aircraft in a flight simulation program. We discuss the problems encountered and present them as a challenge for researchers in Inductive Logic Programming. Overcoming these problems will require ILP methods that go beyond our current knowledge, including induction over noisy numeric domains, dealing with time and causality and complex predicate invention.

1. Learning Control Rules

Almost all applications of inductive learning, so far, have been in classification tasks such as medical diagnosis. For example, medical records of patients' symptoms and accompanying diagnoses made by physicians are entered into an induction program which constructs rules that will automatically diagnose new patients on the basis of the previous data. The output is a classification. We are interested in automatically building control rules that output an action. That is, when a state of a dynamic system arises that requires some corrective action, the rules should be able to recognise the state and output the appropriate action. Just as diagnostic rules can be learned by observing a physician at work, we should be able to learn how to control a system by watching a human operator at work. In this case, the data provided to the induction program are logs of the actions taken by the operator in response to changes in the system.

In a preliminary study (Sammut, Hurst, Kedzier and Michie, 1992), we have been able to synthesise rules for flying an aircraft in a flight simulator. The rules are able to make the plane take off, fly to a specified height and distance from the runway, turn around and land safely on the runway. While control systems have been the subject of much research in machine learning in recent years, we know of few attempts to learn control rules by observing human behaviour. Michie, Bain and Hayes-Michie (1990) used an induction program to learn rules for balancing a pole (in simulation) and earlier

work by Donaldson (1960), Widrow and Smith (1964) and Chambers and Michie (1969) demonstrated the feasibility of learning by imitation, also for pole-balancing. To our knowledge, the autopilot described here is the most complex control system constructed by machine learning methods. However, there are still many research issues to be investigated and they are the subject of this paper. The main problems we discuss are listed below.

- The difference between learning classifications and learning actions is that the learning algorithm must recognise that actions are performed in response to, and result in, changes in the system being controlled. Classification algorithms only deal with static data and do not have to cope with temporal and causal relations.
- In our preliminary study we were able to demonstrate the feasibility of learning a specific control task. The next challenge is to build a generalised method that can learn basic skills that can be used in a variety of tasks. These skills become building blocks that can be assembled into a complete new controller to meet the demands of a specified task.
- One of the limitations we have encountered with existing learning algorithms is that they can only use the primitive attributes supplied in the data. This results in control rules that cannot be understood by a human expert. Constructive induction (or predicate invention) may be necessary to build higher-level attributes that simplify the rules.

We believe it is important that machine learning research should be directed towards acquiring control knowledge since this will give us a way of describing human subcognitive skills and it will result in useful engineering tools.

One of the outstanding problems our research addresses is that subcognitive skills are inaccessible to introspection. For example, if you are asked by what method you ride a bicycle, you will not be able to provide an adequate answer because that skill has been learned and is executed at a subconscious level. By monitoring the performance of a subcognitive skill, we are able to construct a functional description of that skill in the form of symbolic rules. This not only reveals the nature of the skill but also may be used as an aid to training since the student can be explicitly shown what he or she is doing.

Learning control rules by induction provides a new way of building complex control systems quickly and easily. For example, the need in aerospace for pilots to control airplanes close to the margin of instability is putting increasing pressure on present techniques both of pilot training and of flight automation. We claim that it will be possible to build a pilot's assistant using inductive methods. A control engineer is only able to supply automated modules, such as autolanders, provided that envisaged meteorological or other conditions are not too abnormal. There are specialised manoeuvres that the pilot would be relieved to see encapsulated into an automated sub-task, but which cannot, for reasons of complexity and unpredictability, be tackled with standard control-theoretic tools. Yet they can be tackled, often at the expense of effectiveness or safety, by a trained pilot's skills that have been acquired by practice but

which the pilot cannot explain. Control engineers and programmers, much as they might wish to, at present have no way to capture these procedures so as to solve the flight automation problem. In this context, the industry requires a convenient, and not too expensive, means of automatically constructing models of individual piloting skills.

While our experiments have been primarily concerned with flight automation, inductive methods can be applied to a wide range of related problems. For example, an anaesthetist can be seen as controlling a patient in an operating theatre in much the same way as a pilot controls an aircraft. The anaesthetist monitors the patient's condition just as a pilot monitors the aircraft's instruments. The anaesthetist changes dosages of drugs and gases to alter the state of a system (the patient) in the same way that a pilot alters thrust and attitude to control the state of a system (the aircraft). A flight plan can be divided into stages where different control strategies are required, eg. take-off, straight and level flight, landing, etc. So too, the administration of anaesthetics can be divided into stages: putting the patient to sleep, maintaining a steady state during the operation and revival after the procedure has been completed.

In the next section, we will describe our preliminary experiments using a decision tree induction program. While we were able to meet our initial goals, we believe that we are reaching the limits of the descriptive power of propositional learning algorithms and will have to a first-order system. Unfortunately, no existing Inductive Logic Programming algorithm is suitable for use in control applications. Section 3 describes some of the problems that we face and section 4 suggests a number of avenues of research for ILP.

2. Preliminary Study

This section provides a brief description of our preliminary study into constructing rules for an autopilot by logging the flights of human pilots. The reader is referred to (Sammut, Hurst, Kedzier and Michie, 1992) for more detail. The source code to a flight simulation program was made available to us by Silicon Graphics Incorporated (SGI). Our task was to log actions taken by 'pilots' during a number of 'flights' on the simulator. These logs were then used to construct, by induction, a set of rules that could fly the aircraft through the same flight plan that the pilots flew. The results presented below are derived from the logs of three subjects who each 'flew' 30 times. We will refer to the performance of a control action as an 'event'. During a flight, up to 1,000 events can be recorded. With three pilots and 30 flights each the complete data set consists of about 90,000 events. An autopilot has been constructed for each of the three subjects. Each pilot is treated separately because different pilots can fly the same flight plan in different ways.

The central control mechanism of the simulator is a loop that interrogates the aircraft controls and updates the state of the simulation according to a set of equations of motion. Before repeating the loop, the instruments in the display are updated. The display update has been modified so that when the pilot performs a control action by

moving the mouse or changing the thrust or flaps settings, the action and the state of the simulation are written to a log file. The data recorded are:

on_ground	boolean:	is the plane on the ground?
g_limit	boolean:	have we exceeded the plane's g limit
wing_stall	boolean:	has the plane stalled?
twist	integer:	0 to 360° (in tenths of a degree, anti-clockwise)
elevation	integer:	0 to 360° (in tenths of a degree, anti-clockwise)
azimuth	integer:	0 to 360° (in tenths of a degree, anti-clockwise)
roll_speed	integer:	0 to 360° (in tenths of a degree per second)
elevation_speed	integer:	0 to 360° (in tenths of a degree per second)
azimuth_speed	integer:	0 to 360° (in tenths of a degree per second)
airspeed	integer:	(in knots)
climbspeed	integer:	(feet per second)
E/W distance	real:	E/W distance from centre of runway (in feet)
altitude	real:	(in feet)
N/S distance	real:	N/S distance from northern end of runway (in feet)
fuel	integer:	(in pounds)
rollers	real:	±4.3
elevator	real:	±3.0
rudder	real:	not used
thrust	integer:	0 to 100%
flaps	integer:	0°, 10° or 20°
spoilers	integer:	not relevant for a Cessna

Most of the attributes of an event are numeric, including real numbers, sub-ranges and circular measures. Since there can be an enormous amount of variation in the way pilots fly, the data are very noisy. Note also that the output value of induction is a control setting such as the position of the flaps, rollers or elevator. Thus, the output values are also required to be numeric.

At the start of a flight, the aircraft is pointing North, down the runway. The subject is required to fly a well-defined flight plan that consists of the following manoeuvres: take off and fly to an altitude of 2,000 feet; level out and fly to a distance of 32,000 feet from the starting point; turn right to a compass heading of approximately 330°; at a North/South distance of 42,000 feet; turn left to head back towards the runway; line up on the runway and descend; land on the runway.

The data from each flight were segmented into the stages listed above. For each stage we construct four separate decision trees for the elevator, rollers, thrust and flaps. The rudder is not used. A program filters the flight logs generating four input files for the induction program. The attributes of a training example are the flight parameters of the simulator, listed above. The dependent variable or class value is the attribute describing a control action. The reason for segmenting the data is that each stage requires a different manoeuvre. By combining all the data from all stages, we would be expecting the induction program to construct seven sets of rules for controlling the aircraft in each of the seven stages. This makes the program's task more difficult than is necessary since we have already defined the sub-tasks and have told the human subjects what they are. It is reasonable that the learning program should have the same information as the pilots.

For the preliminary study, we used the decision tree induction program C4.5 (Quinlan, 1987). To test the induced rules, the original autopilot code in the simulator is replaced by the rules. A post-processor converts C4.5's decision trees into if-statements in C so that they can be incorporated into the flight simulator easily. Hand-crafted C code determines which stage the flight has reached and decides when to change stages. The appropriate rules for each stage are then selected in a switch statement. Each stage has four, independent if-statements, one for each action.

We demonstrate how these rules operate by describing the controllers for the first stage. The critical rule at take-off is the elevator rule:

```
elevation > 4 : level_pitch
elevation <= 4 :
|   airspeed <= 0 : level_pitch
|   airspeed > 0 : pitch_up_5
```

This states that as thrust is applied and the elevation is level, pull back on the stick until the elevation increases to 4°. Because the controls take some time to respond, the final elevation usually reaches 11°, which is close to the values obtained by the pilot. `pitch_up_5` indicates a large elevator action, whereas, `pitch_up_1` would indicate a gentle elevator action. The other significant control at this stage is flaps:

```
elevation <= 6 : full_flaps
elevation > 6 : no_flaps
```

Once the aircraft has reached an elevation angle of 6°, the flaps are raised.

The rules we have synthesised are successful in the sense that the plane follows the flight plan just as the human trainer would and lands safely on the runway. Because induction over a large set of data has an averaging effect, the autopilot actually flies more smoothly than the trainer. Figure 1 shows a profile of the trainer's flight, plotting the E/W distance travelled as a function of the N/S distance away from the runway. Each point represents an action being taken by the pilot. This flight can be compared with the autopilot's flight shown in Figure 2. A similar comparison can be made between the altitude profiles for the trainer and the autopilot, shown in figures 3 and 4,

Figure 1: Cross-range Profile for Trainer

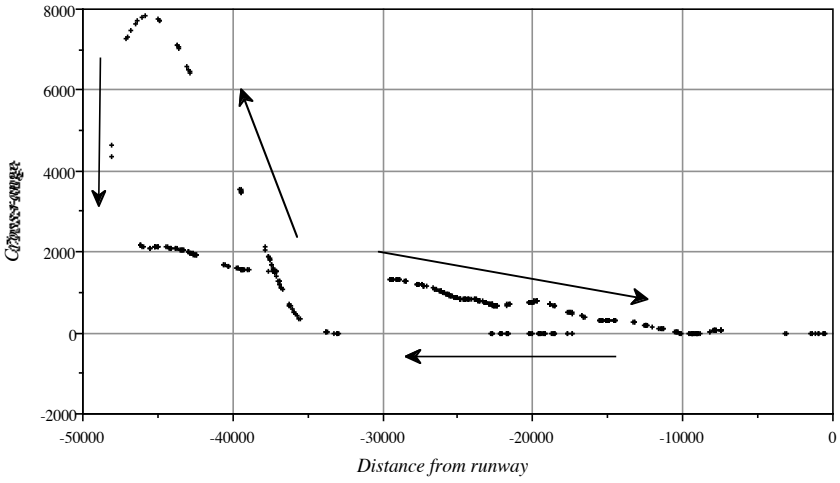
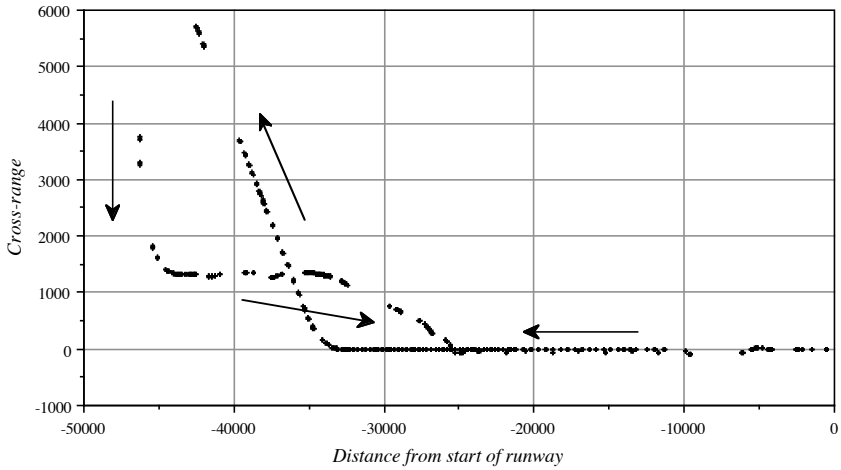


Figure 2: Cross-range Profile for Autopilot



respectively.

It is important to note that this study was not merely an exercise in collecting data and applying an existing program. The output of an induction program is only as good as its input. Before obtaining a successful outcome it was necessary to learn how best to collect and analyse the data and while the study has demonstrated the possibility of building control systems by induction, it has also revealed many problems still to be solved. We discuss these problems in the next section and our proposed solutions after that.

3. Problems

While our work as been concentrated on the domain of flight automation because it provides us with a complex, realistic task that ensures our methods remain practical, we are mindful of the importance of keeping our methods sufficiently general that they can be used in other domains. Thus, while the issues we discuss below are described in

Figure 3: Altitude Profile for Trainer

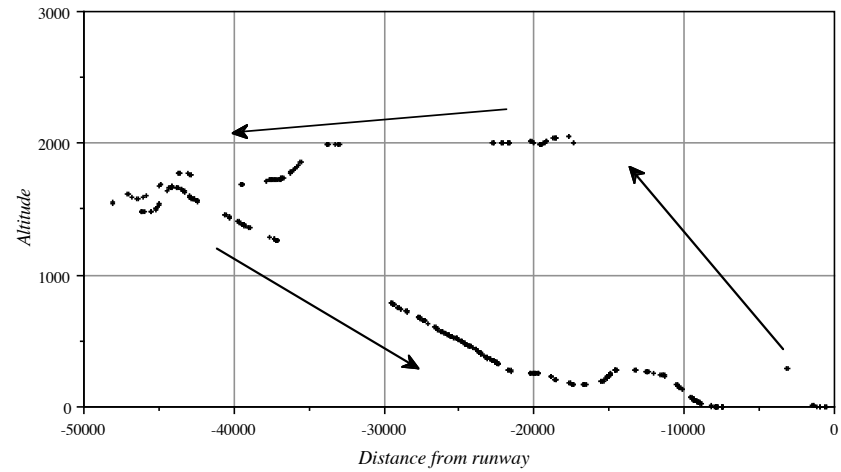
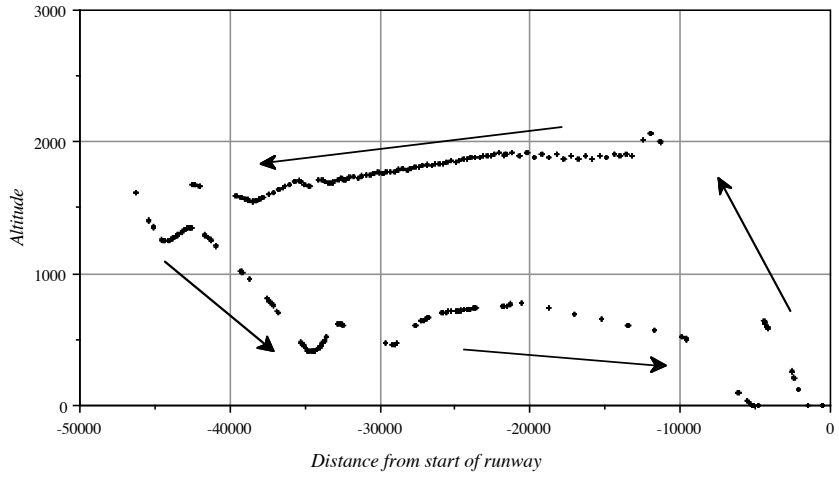


Figure 4: Altitude Profile for Autopilot



terms of the flight simulator, the reader should keep in mind that similar problems exist in almost all control tasks. The challenge posed in this paper is, "Can Inductive Logic Programming methods be useful in solving these problems and how must they be extended in order to handle control tasks"?

3.1 Causality

For our preliminary study, we used an induction program that was designed for learning classifications, not control actions. This program, and others of its kind, produced flight rules such as:

```
if speed > 130 knots
then thrust = 100%
```

This accurately summarises the data since high speed is usually due to high thrust. Unfortunately, it is not a useful control rule since it tells us how we got to a high speed but not what to do now that we're there. The induction algorithm has no way of recognising a causal relationship between thrust and speed.

There has been some work in trying to extract causal relations from data (Bratko, Muggleton & Varsek, 1991) but this can be quite difficult when there are many variables and actions involved. However, the task becomes easier when the program is provided with background knowledge about causality.

3.2 What does the pilot see?

We do not record the same information that the pilot uses. For example, when landing, a pilot usually chooses an 'aiming point' on the runway and stays on a steady glide path by maintaining a direction toward the aiming point. The aiming point and deviation from it are not recorded since these are attributes chosen by the pilot and are not part of the instrumentation of the aircraft. As a result, rules become more complicated than they should be because they must effectively perform a coordinate transformation between the recorded parameters and the pilot's parameters. To avoid this problem, the induction

program would have to reconstruct the information that the pilot is using from the data and generate new attributes that better reflect the pilot's goals.

3.3 Learning General Skills

The present method only learns how to fly a specific flight plan and so is very limited. We should be able to learn basic skills that can be used as building blocks for a complete flight. So instead of learning a flight plan, the system should learn rules for sub-tasks such as, 'turn left through X degrees' or'climb to altitude H'. Once building blocks have been learned, it should be possible to use well-known planning algorithms to put them together into a complete flight plan. To make 'reusable' rules requires an induction system that can generate concepts in a first-order language (ie. with variables and relations).

3.4 Brittleness

When strong disturbances are allowed in the simulation, the airplane can be sent a considerable distance off-course. If there were no training data for such a situation, the rules in the autopilot will be meaningless and will result in nonsensical behaviour. A human pilot would use a deep model to guess the appropriate action to take but our present system cannot even recognise the limits of its knowledge, let alone reason about is predicament.

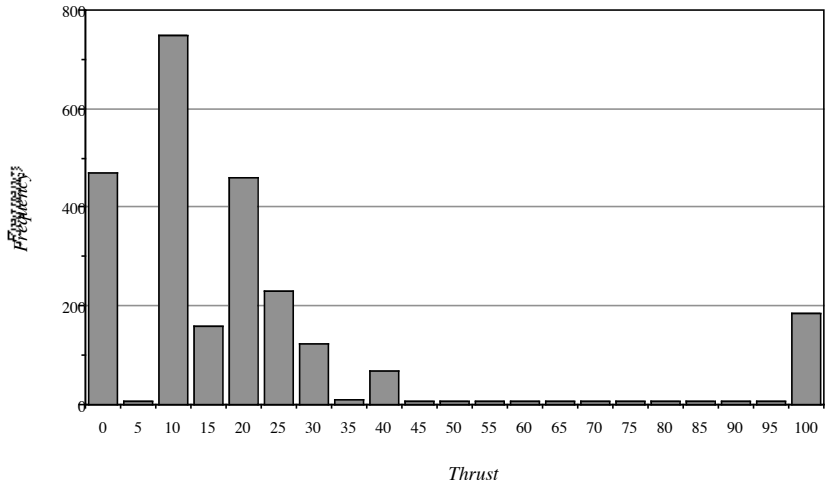
3.5 Determining class values

C4.5 expects class values to be discrete but the values for elevator, rollers, thrust and flaps are numeric. No ILP system currently allows numeric values. For these experiments, a preprocessor breaks up the action settings into sub-ranges that can be given discrete labels. Sub-ranges are chosen by analysing the frequency of occurrence of action values. This analysis must be done for each pilot to correctly reflect differing flying styles. There are two disadvantages to this method. One is that if the sub-ranges are poorly chosen, the rules generated will use controls that are too fine or too course. Secondly, C4.5 has no concept of ordered class values, so classes cannot be combined during the construction of the decision tree.

Figure 5 shows the frequency of thrust values in stage 6 of the data for one pilot. Since thrust is controlled by a keystroke, it is increased and decreased by a fixed amount, 10%. The values with very low frequencies are those that were passed through on the way to a desired setting. The graph reflects the facts that this pilot held the thrust at 100% until the approach to the runway began. The thrust was then brought down to 40% immediately and gradually decreased to 10% where it remained for most of the approach. Close to the runway, the thrust was cut to 0 and the plane glided down the rest of the way.

In this case, class values corresponding to 0, 10, 15, 10, 25, 30, 35, 40 and 100 were used. Anything above 40% was considered full-throttle. Anything below 10% was considered idle. Another reasonable clustering of values could be to group values from 15 to 35 together.

Figure 5. Frequency of thrust values in stage 6



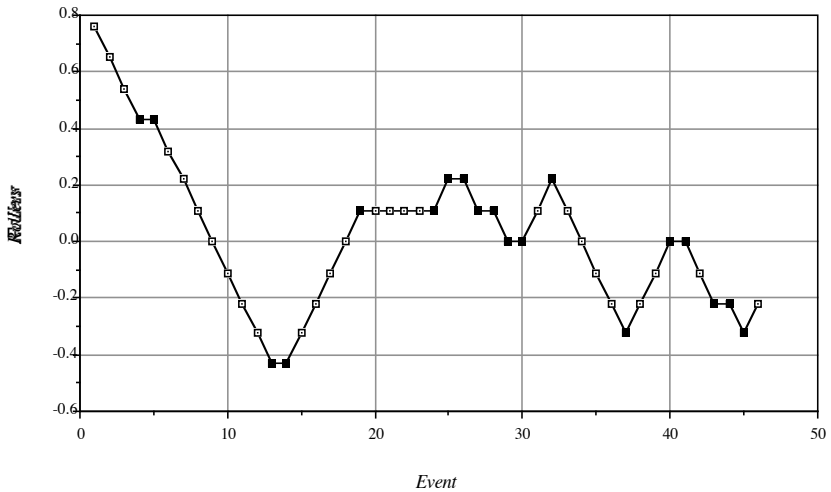
3.6 Absolute and Incremental Controls

An event is recorded when there is a change in one of the control settings. A change is determined by keeping the previous state of the simulation in a buffer. If any of the control settings are different in the current state, a change is recognised. For example, if the thrust is being reduced from 100% to 40%, all of the values in between are recorded. For thrust, these values are easily eliminated as noise during induction.

It is not so easy to eliminate spurious values from the elevator and rollers data. Both thrust and flaps can be set to a particular value and left. However, the effects of the elevator and rollers are cumulative. If we want to bank the aircraft to the left, the stick will be pushed left for a short time and then centred since keeping it left will cause the airplane to roll. Thus, the stick will be centred after most elevator or roller actions. This means that many low elevator and roller values will be recorded as the stick is pushed out and returned to the centre position.

To ensure that records of low elevator and roller values do not swamp the other data, another filter program removes all but the steady points and extreme points in stick movement. Figure 6 shows a small sample of roller settings during a flight. Each point

Figure 6. Change in rollers



on the graph represents one event. Clearly many of the points are recorded as part of a single movement. The filter program looks for points of inflection in the graph and only passes those on to the induction program. In this graph, only the points marked in black will get through the filter.

3.7 Good Pilots are Bad

Another problem for data collection is the surprising fact that poor pilots often provide better training data than good pilots. Of the three pilots for whom we have data, one flew very erratically, the other two flew smoothly. The data from the erratic pilot have been the easiest to synthesise rules from. This is because there are many more examples of actions being performed to correct errors. A good pilots provides few of these, so the induction program does not generate rules for course corrections when the aircraft strays. We must devise a learning system that is robust enough to produce reliable control rules for any pilot who is able to fly competently. One way of doing this is to incorporate disturbances in the simulation and record the pilot's corrective manoeuvres. This will also be necessary since we want to demonstrate that the autopilot can handle difficult conditions.

4. Solutions?

Three components are required in an algorithm for learning control rules: a representation language that includes variables and relations, the ability to invent new attributes and relations, and using background knowledge of causality.

These issues have, to some extent, been investigated by researchers in Inductive Logic Programming (ILP). Bratko, Muggleton and Varsek (1991) have shown that causal models can be learned by ILP when a theory of qualitative physics is supplied as background knowledge. This work has not yet been extended to learning control actions but it is very promising.

Unfortunately, current ILP systems cannot handle the noisy numeric data generated by a flight simulator and propositional learning programs (such as C4.5) that can cope with numeric data, cannot make use of background knowledge. Thus we are faced with the prospect of trying to extend one or the other method. As an alternative, hybrid systems as suggested by Lavrac, Dzeroski and Grobelnik (1991) are an interesting possibility. Using this approach, we take advantage of well-developed decision tree induction algorithms to perform a first pass of the data and then map the results of the decision tree analysis into a first-order language for further processing. In the transformation to a first-order language, constants are turned into variables, allowing the rules to be reused in a different context, thus enabling us to create building blocks for new flight plans.

The problem of constructing new attributes and relations to simplify the output of a learning program has been investigated but only using discrete valued data or simple numeric data. Our problem is that we are dealing with noisy numeric data that have very complex relationships. It is not reasonable to expect that a program will be able to

reconstruct flight equations from the data or perform complex transformations without some help. So, while humans are unable to explain their subcognitive skills, they are able to provide knowledge in the form of a theory to help guide induction. Thus, it is essential to provide a flexible way of describing background knowledge so that it can be used to preprocess that data before induction.

Causal relations illustrate the importance of data analysis prior to induction. A single training example for the induction program consists of an action, along with the state of the simulation when the action was taken. Normally, an induction program is predictive. For example, the thrust rule discussed earlier, predicts that when the speed is greater than 130 knots, the thrust will be at 100%. However, we know that the training examples are logged when an action is taken. For example, a thrust action is recorded when the thrust is changed from one setting to another. So we already have some clues about causality. We know that the thrust has changed. Now we want to know why. One way of doing this is to record the state of the simulation, not as it is at the time of the action, but as it was some time before. In addition, if we have a qualitative model that tells us that airspeed depends on thrust and elevation and that a certain airspeed must be maintained to avoid stalling then these values can be weighted to help the induction program choose the criteria for an action.

The presence of a causal model can also help remove brittleness from the control system. We envisage a complete control system consisting of a high-level planner that uses a qualitative model to establish goals for the system. Rules derived from human-data are invoked to achieve known goals. When the system wanders into regions for which there has been no training, the qualitative model may be used to guide control. This is necessarily slower and not desirable when rapid real-time response is required, but may be the only alternative in unknown regions.

5. Conclusion

We have presented a problem in learning control rules for real-time systems by observing human operators. This domain of applications has great significance for industry and is fruitful area of research. While existing induction algorithms go some way toward solving the problem, they are inadequate for further progress. This requires new techniques being pioneered in Inductive Logic Programming. However, ILP has not yet addressed a number of issues that are essential if it is to move into the domain of control systems. These include processing noisy numeric data and the invention of predicates to describe complex numeric relationships. In addition the work that has already begun in combining qualitative modelling and ILP must be developed further.

Acknowledgments

Donald Michie suggested the problem and the method of using induction to learn reactive strategies. Scott Hurst and Dana Kedzier were the chief test pilots and data analysts. Mark Pendrith performed many modifications to the flight simulator. Jim

Kehoe and Peter Horne conducted human reaction time studies and collected much valuable data. Silicon Graphics Incorporated made the source code of the flight simulator available. This research has been supported by the Australian Research Council and the University of New South Wales.

References

Bratko, I., Muggleton, S. and Varsek, A. (1991). Learning Qualitative Models of Dynamic Systems. In S. Muggleton (Ed.), *Proceedings of the International Workshop on Inductive Logic Programming*, (pp. 207-224). Viana do Castelo, Portugal.

Chambers, R. A. and Michie, D. (1969). Man-machine co-operation on a learning task. In R. Parslow,R. Prowse and R. Elliott-Green (Eds.), *Computer Graphics: Techniques and Applications* London: Plenum.

Donaldson, P. E. K. (1960). Error decorrelation: a technique for matching a class of functions. In *Proceedings of the Third International Conference on Medical Electronics*, (pp. 173-178).

Lavrac, N., Dzeroski, S. and Grobelnik, M. (1991). Learning Non-Recursive Definitions of Relations with LINUS. In Y. Kodratoff (Eds.), *Proceeding of the 1991 European Working Session on Learning* (pp. 265-281). Porto, Portugal: Springer-Verlag.

Michie, D., Bain, M., and Hayes-Michie, J.E. (1990). Cognitive Models from Subcognitive Skills. In M. Grimble, S. McGhee and P. Mowforth (Eds.) *Knowledge-base Systems in Industrial Control*.Peter Peregrinus.

Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies,* **27**, 221-234.

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

Widrow, B. and Smith, F. W. (1964). Pattern recognising control systems. In J. T. Tou and R. H. Wilcox (Eds.), *Computer and Information Sciences* Clever Hume Press.