12

Learning in Control

Edward Grant
Department of Computer Science
The University of Strathclyde
Livingstone Tower
Richmond Street
Glasgow G1 1XH
U.K.

Abstract

In this Chapter we review the three separate methods by which intelligent control can be applied in dynamic system control; human control, passive-learning, and machine learning. Working from the knowledge that humans posses the ability to control complex dynamic systems by applying simple heuristics our first task was to establish those heuristics for a given dynamic domain, a pole and cart system. In our work two models of the pole and cart were constructed; one was a computer simulation, the other a physical system. First we "captured the soul" of a human who was sufficiently adept, and proficient, at controlling the simulator. This was their rules for control. Later, these same rules were encoded as a rule-base automatic controller for the purpose of conducting performance trials on the simulator and the physical system. A comparative study was also undertaken in this phase, where the performance of our controller was tested against a second rule-based controller, a controller using rules that were derived by interpreting the dynamic equations only. This concluded the non-learning phase. In the passive-learning phase the cause-effect signals recorded during our rule-based controller, controlling the physical system, were post-processed using rule-induction to continuously refine and tune the rules needed to control the process effectively. The last phase, machine-learned control, assumed no "a priori" knowledge of the process. Here, the two types of machine learning was examined; the 'BOXES' machine-learning algorithm and, neural networks. In this first series of trials both performed equally well in the simulated and physical worlds. However, there were certain features observed when using the machine learning algorithm in the physical domain that was particularly noteworthy.

1. INTRODUCTION

This chapter demonstrates how artificial intelligence techniques based on learning can be applied, in the control of complex dynamic systems. Our initial interest was the development of adaptive rule-based controllers where the control rules were derived through observing human skills, or by trial-and-error learning. Although it has long been recognised that humans are good at mastering the control of dynamic systems, it has been equally well understood that they are poor at articulating the heuristics they have been applying. Were such individuals able to express their intuitive rules comprehensively, an expert system could be constructed that would emulate their skills completely. However, previous experiences in building expert systems had taught us that this route was tortuous even for well structured, well understood systems. What chance of success could we then expect when we were dealing with a complex dynamic system, a pole and cart, in the physical and the computer simulated worlds. Controlling a system of this type requires instantaneous, reflex, responses. Rather than conduct extensive in-depth trials into the cognitive skills of a variety of subjects operating the system we attempted simply to "capture the sole" of a single skilled human, then use this captured knowledge to produce a rule-based automatic controller. The capabilities possessed by our rule-based controller were also measured in a series of comparative trials against a Makarovic controller [13]. This second rule-based controller, Controller B, uses control rules that were derived through considering the systems dynamic equations only. The pole and cart proved useful to test the efficiency and robustness of all types of controllers examined, human, rule-based, and later machine learned. Control was achieved by applying either a left or right force of constant magnitude to the cart, bang-bang control.

The second phase of our work dealt with passive-learning from human control decisions. In the context of our work passive-learning means using off-line rule induction [4, 9, 12, 17] to interpret and refine the cause-effect relationships between the rule-based controller and the pole and cart test-beds. When our rule-based controller was applied to the physical system the sensed outputs were processed by the c4.5 induction rule generation algorithm to demonstrate passive-learning. Passive-learning of this type provided feed-back and allowed a direct comparison to be made between human and machine generated rules. After a limited series of passive-learning experiments it was decided to move quickly to the final phase to be dealt with, a comparative study of machine learned adaptive controllers. An alternative passive-learning method we adopted used neural networks. Here, experimental data was qualitatively partitioned into binary patterns using a hand-crafted rule. These patterns were then used to train the neural network, which was then became a controller.

Initially, the experiments in the final phase were to be undertaken completely using machine learning algorithms, this decision was later reviewed and neural-network controllers were included in the programme. These machine-learned controllers showed that they could adapted to changing system parameters. Learning experiments conducted on the physical system, under human control,

have shown that it is impossible for a human to control even the simplest physical apparatus.

However, through the use of observation, training on the simulator and on a physical system, we have produced effective controllers and control algorithms that can adapt to changing system parameters. Insight into the strengths and limitations of using Machine Learning Control (MLC) in adaptive control situations, machine learning algorithms and neural-networks, was gained through working in the physical world. For example, in our trials into MLC the neural-network controller eventually performed better than a controller based on the 'BOXES' learning algorithm acting as an automatic controller. This, in part, may be due to the manner in which sensory data is handled by the 'BOXES' learning algorithm used as the MLC.

Artificial intelligence techniques have begun to enter control engineering [6, 18]. The requirement for an accurate mathematical model of the process to be controlled and the inability to set meaningful goals for the adaptive mechanism are the two major difficulties in designing an adaptive controller using conventional control theory. The expert controller equipped with the control engineers' knowledge and skills overcomes the above two difficulties. Additionally when attempting to implement an expert controller in real-world problems, the experts must transfer their skill into the controller. This knowledge transfer phase is never completely achieved by asking the expert to articulate their rules because their skill is semi-intuitive. To tackle real-world problems with this kind of expert behaviour, an expert controller with learning capabilities must be built, so that the expert can transfer skill by a "tutorial" process of showing examples. That is, the machine must be able to build up it's skill in the domain automatically in the form of appropriate control actions triggered by the sensed state variables.

By machine learning adaptive control here we mean the control scheme is independent of the mathematical modelling of the object dynamics and the precise estimation of its physical parameters. Both the above are required by adaptive control using conventional control theory. Learning controllers acquire their control skills either by watching or by doing. Michie and Chambers 'BOXES' algorithm [14] was an early example of learning control actions by self-optimization. The task chosen was that of learning by trial and error to balance the pole in an inverted pendulum and cart system. Since then the pole and cart problem has become a widely accepted bench-mark problem in the area of machine learning. One of the interesting features of the 'BOXES' algorithm is its threshold subdivision of state space into local regions. More recently, this method of reducing the problem space to a manageable size has been adopted by many authors in developing connectionist-net based control-learning procedures. Among them are ASE/ACE [3] and a multilayer extension of this work by Anderson [1]

The paradigm of learning by imitating was demonstrated by Donaldson [7] as long ago as 1960, using the pole balancing problem as a test case. In his experiment

the balancing skill was adaptively acquired from a skilled human's control responses. Widrow and Hoff [25] made a similar acquisition-from-tutor approach to the pole balancing problem based on their Perceptron-like ADALINE device. Chambers and Michie [5] also demonstrated how an acquisition-from-tutor capability could be incorporated into their 'BOXES' algorithm.

However, one objective that has not been realised by any tutorial-style use of neural nets is for the learning algorithm to discover and form a summary of the trainers' strategies. Controller design for the inverted pendulum and cart system (pole and cart system) is well understood in terms of modern control theory based on its Newtonian dynamics [8, 16, 21]. The task is to balance a pole that is mounted on a movable cart within the limits of a track by applying a force of fixed magnitude either left or right to the cart. However, to make such a controller adaptable remains to be a non-trival task in control engineering field.

Michie and Chambers [14] were among the first trying to solve adaptive control problems using a machine learning approach. They developed the 'BOXES' algorithm to balance the pole-cart system. More recently, the Adaptive Heuristic Control (AHC) algorithm developed by Selfridge et.al.[22] and the CART algorithm developed by Connell and Utgoff [6] both used the pole and cart problem as a test case. Sammut [20] presented a detailed survey and simulation results on pole balancing using the above three machine learning algorithms. The major findings were (i) the learning algorithms were independent of the equations of motion and, (ii) control skill is acquired by empirical derivation from sample data. Through implementing these algorithms some of their individual deficiencies became apparent, such as (iii) learning only taking place after trial failure and, (iv) a dependency on a set of learning parameters that are a function of the system. Because of (iii), the practical use of these algorithms is limited. The result of (iv) is that the controller finds it difficult in adapting to changing system parameters. Recently, Makarovic [13] used a totally different approach to the data-oriented one above. He derived control rules based on a study of the equations of motion for the pole and cart system according to classical mechanics. His theory-oriented approach was semi-intuitive and it does not work if the system to be controlled is too complex.

In this Chapter, we present an approach which is different from both the theory-oriented and data-oriented approaches outlined above. Our technique requires that the control skill is acquired by modelling the experience of a skilled human controller. The skill of a human is embedded in a set of rules which in turn leads to developing a rule-based adaptive controller for the pole-cart system. Also here, we report on our investigations into the use of a connectionist-net of the 'Rumelhart' type [19], for learning and exercising intelligent control of the pole and cart system. The controller consists of a three layer feed-forward network, with a sigmoidal activation function, that is trained using Rumelhart's back-propagation algorithm [19]. Unlike previous efforts, the state variables of the pole-cart system are not directly used as the input to the network, they are pre-processed to form

patterns which give a qualitative description of the system. These patterns are in turn used as the input to the network. The output of the network is used as the control signal for the pole-cart system. The training data is generated by recording each system state and the corresponding action taken by a human teacher when he controls the computer simulated pole-cart system. When presented with the same input pattern as the human was during training, the neural-net must produce the same output response as the human [10]. This is accomplished by the net adjusting the weights of its internal inter-connections and its activation thresholds to conform with these input-output constraints. This learning process is performed off-line. Results show that the neural network controller is able to learn a control law which is not explicitly known to the human teacher and generate stable control without the existence of the human intervention.

We have gained a knowledgeable insight into the use of rules for controlling dynamic systems. The rules we we used to develop a range of rule-based controllers has proved robust, efficient and adaptable, and they have been applied in the control of other simulated and physical process [24]. One rule has also become the basis of the neural-net controller. The work reported on here catalogues experiments into human-supervised learning, passive-learning, and machine learning.

2. THE SIMULATED DYNAMIC SYSTEM

All majority of the work undertaken in the three phases of experimentation, non-learning human control, passive-learning, and machine-learning, were carried out on pole and cart systems. Basically, the pole and cart can be considered as a trolley that moves in the plane of the track upon which is mounted a hinged inverted pendulum, see Figure 2.1. The control objective is to keep the pole as near the vertical as possible, and the cart within the bounds of the track. Such a system is unstable in the open loop because small disturbances of the cart make the pole rotate, or oscillate, about the pivot. Control is obtained by applying a right or left force to the cart via a d.c. motor, see Figure 3.2. The pole and cart is just one example of a general class of dynamic system that is made up of a collection of interacting and interdependent objects. The state of the system is altered by the external stimuli, in our case the d.c. motor input, whose role is to control the system within strict bounds by manipulating the input. The system outputs are related to the system states, but it may be neither possible, or desirable, to measure all such variables. The controller; in our case the human, the rule-based automatic controller, or the MLC, is responsible for determining the appropriate control action based on knowledge of the input signal, monitoring of the output channels, and feedback. The controller must also be capable of dealing with disturbances, which can be both internally and/or externally generated. For such systems one of the major difficulties is the provision of an accurate model of the system, this is particularly so if there is an associated physical system, Figure 3.1. The force analysis of the system in Figure 2.1 begins by considering the pole and the cart as two separate, unconnected units.

Having noted that the angle of the pole, θ, which needs to be kept small for a

control strategy to be developed, gives only a partial description of the dynamic state of the system the complexity of the analytical analysis of the model must be reduced. By eliminating the common vertical and horizontal forces that act on both the pole and the cart separately, and linearising parts of the equations through considering small perturbations about the operating point, we are left with the two second order differential equations. These can be represented either as one 4th order differential equation, or, four 1st order differential equations. Assuming a solution exists, these two second order equations can be solved if (i) the four initial conditions are known and, (ii) an input force is specified. The simulation uses the following non-linear differential equations [8]:

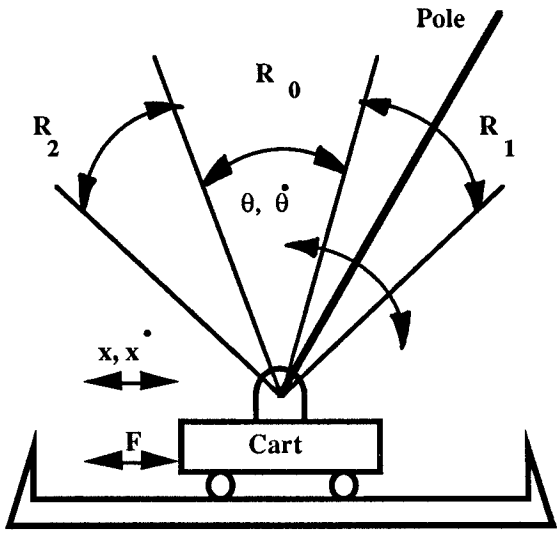

Figure 2.1 The Pole and Cart Notation

where:

m_c = 1.0 kg, mass of cart.
m_p = 0.1 kg, mass of pole.
l = 0.5 m, half length of pole.
g = 9.8 ms^{-2}, acceleration due to gravity.
F = 10 newtons, force applied to carts' centre of mass.

$$\ddot{\theta} = \frac{g \sin\theta - \cos\theta \left[\frac{F + m_p l \dot{\theta}^2 \sin\theta}{m_c + m_p} \right]}{l \left[\frac{4}{3} - \frac{m_p \cos\theta^2}{m_c + m_p} \right]}$$

$$\ddot{x} = \frac{F + m_p l [\dot{\theta}^2 \sin\theta - \ddot{\theta} \cos\theta]}{m_c + m_p}$$

A fourth-order Runge-Kutta method was used to solve the above differential equations, the sampling rate chosen (h) equalled 100 Hz.

3. THE PHYSICAL SYSTEM

The physical system is shown in Figure 3.1.

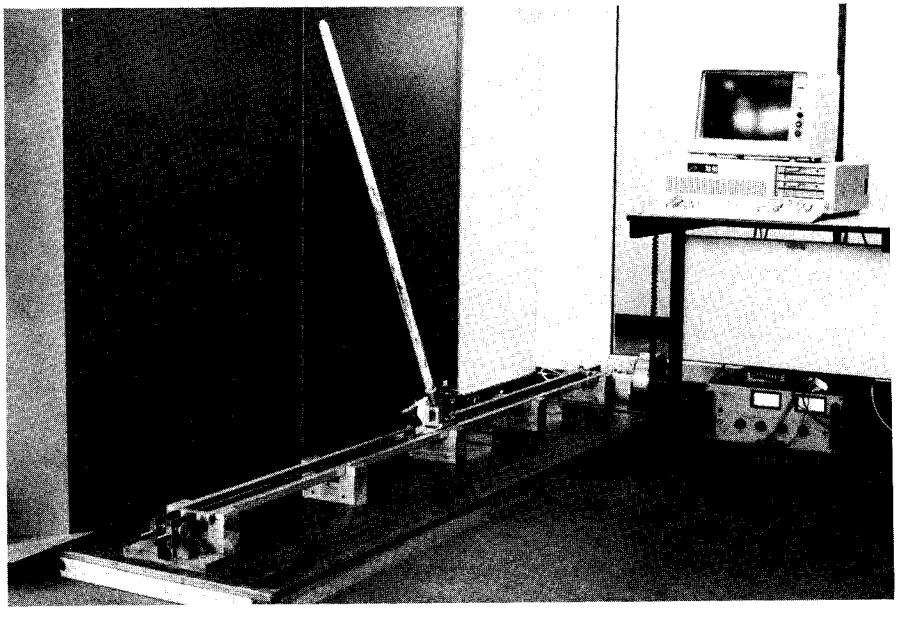

Figure 3.1 The Physical Pole and Cart Apparatus

The motion of the cart mounted on a parallel track is controlled by a d.c. motor via a wire and pulley configuration. The motion of the cart is either "left" or "right", so the control action provided to the cart by the d.c. motor is 'bang-bang' control. There is no intermediate 'do-nothing' control action.

The apparatus shown in Figure 3.1 consists of a 2-degree of freedom system acting in a single plane. The pole was a round wooden rod of uniform density and rotates in 1-degree of freedom about a pivot mounted on the cart, which operates linearly in 1-degree of freedom. End stops are provided on the pole to prevent it passing 20 degrees from the vertical in either the clockwise or the anti-clockwise direction. Optical shaft encoder's were used to monitor the angle of the pole and, after calibration, the linear position of the cart. Data from these shaft encoder's was also used to compute the respective angular and linear velocities of the pole and the cart. These angular velocities were calculated as the average rate of change of angle, and linear position, relative to a previous time period. Obviously this may not totally reflect the current state but it does indicate the trend. Reed switches were mounted at each end of the track to limit the operating range of the cart. These switches were activated by a magnet attached to the cart, their status was monitored through the I/O port of the controller, which was resident in an IBM PC/AT, or a compatible, computer. The computer was responsible for the execution of whichever control algorithm was currently under test. The high level hardware architecture is shown in Figure 3.2.

4. CONTROL STRATEGIES & EXPERIMENTS

In this section we will describe the range of experiments that were conducted during the course of the project. After careful consideration we believed that these experiments should be partitioned into three separate and distinct modules; (i) Non-Learning, or more correctly non-machine learning but human control, (ii) Passive-Learning, a set of experiments similar to (i) but with rule-induction monitoring the cause-effect relationships and, (iii) Machine-Learned Control, a set of experiments where no "a priori" knowledge of the system is known, and where the system is controlled totally using machine-learned controllers that are adaptive. Schematics showing the full range of the experiments and an appropriate key code are shown in Figures 4.1 to 4.4.

4.1 Non-Learning (Human) Control

It is a well understood fact that human operators have the capability to operate complex control processes in instances where conventional numeric techniques proved inadequate. Problems that exhibit such ill-defined complexity can, in many instances, be controlled manually. We define an ill-defined process as one where the interaction between all the process variables is unclear. In such cases manual control using heuristic rules can be effective, this is because humans' handle complexity in an imprecise manner. To arrive at a control decision based on a set of vague rules a human operator will often approximate quantitative data, and consider qualitative information, prior to making a judgement. Rule-based controllers have been applied to physical processes, but these processes were

chosen because their response times are measured in minutes or even hours. In such cases the response times of the simulation can closely match those of the physical system. Physical systems of the type dealt with in this Chapter require an instantaneous response so any automatic controller constructed from trials on a simulator, where response times can be slowed, might be totally unsuitable for controlling the physical process. However, it was considered that these controllers might prove useful during the passive learning experiments where the experimental data could be used to induce rules. Rules induced from experimentation could then be compared with the hand-crafted rule to optimise the parameters. In pole balancing problem, as in Figure 2.1, there are four state variables, these are:

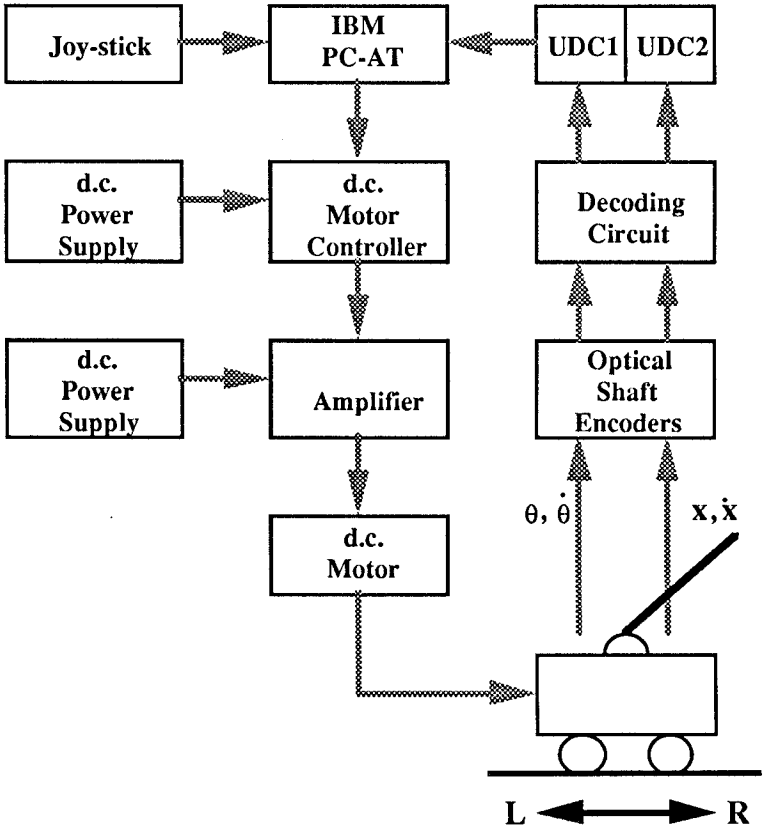

Figure 3.2 The High Level Hardware Architecture

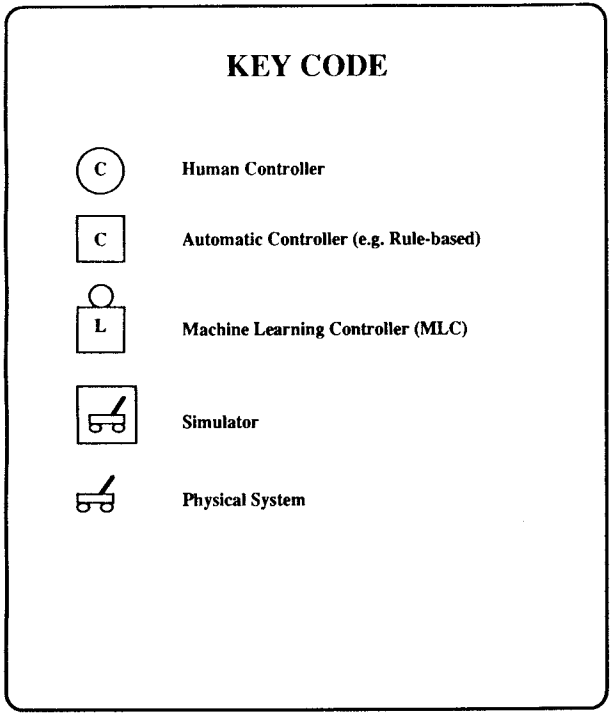

Figure 4.1 Key Code

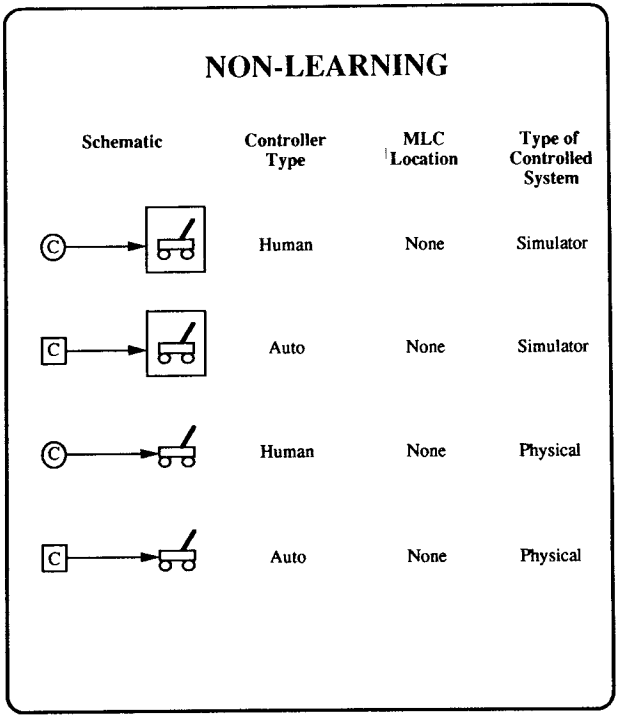

Figure 4.2 Non-Learning Experiments

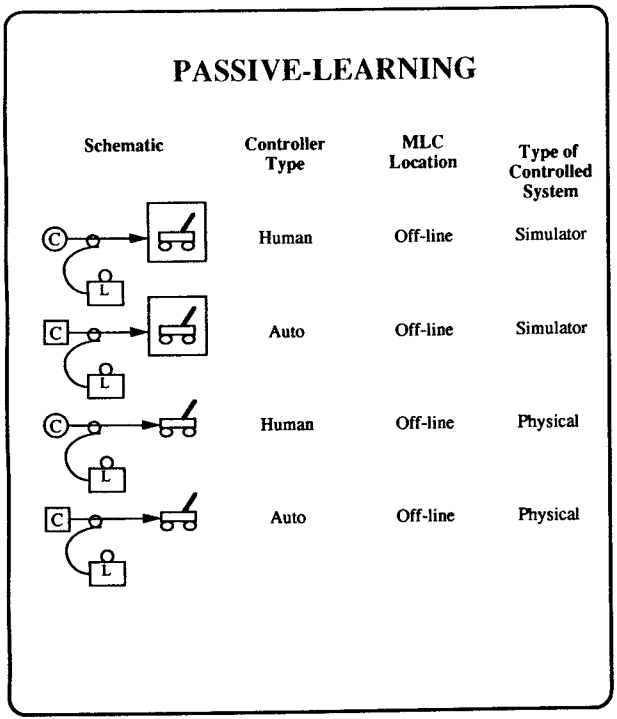

Figure 4.3 Passive-Learning Experiments

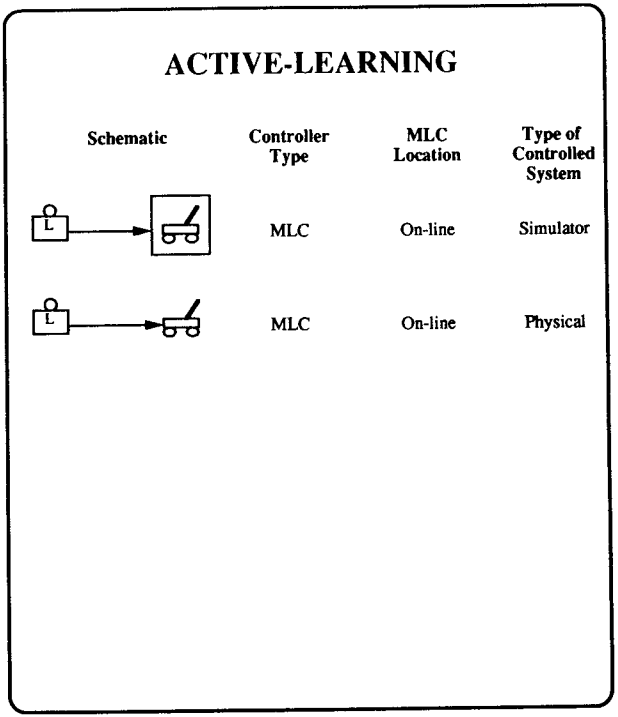

Figure 4.4 Active-Learning Experiments

x = the position of the cart on the track

$\dot{x}$ = the linear velocity of the cart

θ = the angular displacement of the pole from the vertical

$\dot{\theta}$ = the angular velocity of the pole

First of all, we identified one system variable whose out-of-range condition was allowed to trigger system failure, we call this the trigger variable. In our system θ was chosen as that appropriate variable. We then decided upon a threshold value that would divide the range of θ into three regions; region one is where the absolute value of θ is within a given threshold (we call this region R_0), the other two regions are where the value of θ is positively greater than the threshold (Region R_1) and negatively less than the threshold value (Region R_2) respectively. Figure 2.1 shows the notation used for describing this. Working with these thresholds means a control decision is made using three sets of rules, that is there is a rule which corresponds to each of the three regions. If the magnitude of the force applied is constant under bang-bang control, the rules for region R_2 is consistent with, but opposite too, the rule for region R_1. Applying a left or a right force to the cart will produce one of the following responses:

(a) θ increasing and $\dot{\theta}$ increasing

(b) θ increasing and $\dot{\theta}$ decreasing

(c) θ decreasing and $\dot{\theta}$ increasing

(d) θ decreasing and $\dot{\theta}$ decreasing

4.2 Non-Learning (Human) Control of a Simulator

A fourth-order Runge-Kutta method was used to solve the differential equations used in the pole-cart simulation, see the section on simulated dynamic systems, the sampling rate chosen (h) equalled 100 Hz. Also, the track was assumed to be 4.8 m long and the pole angle could not exceed 12 degrees (0.2 radians). These parameters were held constant for all experiments using simulation irrespective of the domain being examined, non-learning, passive-learning or active-learning. The first series of experiments were undertaken after a graphics interface had been constructed for a simulator running on a Compaq 386/20. Human control of the pole-cart system is achieved via a joy-stick (Archer Super-Deluxe Cat. No. 270-1703), that generates control signals from its internal micro-switches. This joy-stick was chosen because its micro-switches gave the necessary "on-off" control action that is the main feature of the type of adaptive control being researched.

Experiments were conducted with a variety of subjects in order to find an appropriate delay time for the simulator, a delay time that would match a range of

human response times. When the simulator had been adjusted accordingly preliminary trials were conducted, again with a variety of human subjects. These trials showed the performances of the human subjects varied considerably, they ranged from poor to excellent. Only boredom, or a lack of concentration, affected these subjects after a period of time. It became analogous to playing a computer game. Five subjects were involved in these trials, some had extensive knowledge of "bang-bang" control of the simulated system on a Compaq, and of qualitative rules. Typical results achieved using the simulator were encouraging, certain individuals were able to balance the pole-cart for periods that extended to several minutes. However, in order to achieve these results it was necessary to "tune" the simulator by adding a delay described above thereby allowing enough time for a human to decide on an appropriate control action.

Let us consider the rules in region R_1 where our convention makes θ positive (clockwise). From (a) and (b) above, the pole has the tendency to fall right, so a right force needs to be applied. If (c) applies, the pole has a tendency to fall left, so a left force is necessary. Finally, if case (d) applies then the pole is moving towards the upright with a low momentum, in this instance a right force needs to be reinforced. Therefore, the rules for region R_1 are as follows:

if ($\theta(k)$ > threshold)
then if ($\theta(k) > \theta(k-1)$) **then** apply a right force
    if (($\theta(k) < \theta(k-1)$) and
      ($|\theta(k) - \theta(k-1)| > |\theta(k-1) - \theta(k-2)|$)) **then** apply a left force
    if (($\theta(k) < \theta(k-1)$) and
      ($|\theta(k) - \theta(k-1)| < |\theta(k-1) - \theta(k-2)|$)) **then** apply a right force

where, $\theta(k)$ is the current value of pole angle and $\theta(k-1)$ is the value of pole angle at the previous time step. A similar set of symmetrical rules can be used to control the system in region R_2, because it is a mirror image of region R_1. In the region R_0 where the pole is almost upright, a different set of rules apply. In this region the pole will not fall suddenly so rules to control the cart position only, e.g. move it to the centre of the track, need to be generated. As before, four states prevail only in this instance they describe cart position relative to applied force:

(e) x moving to the right and $\dot{x}$ increasing

(f) x moving to the right and $\dot{x}$ decreasing

(g) x moving to the left and $\dot{x}$ increasing

(h) x moving to the left and $\dot{x}$ decreasing

Since the goal is to move the cart to the centre of the track, any control rules that

apply when the cart is on the right half of the track will be complementary to those on the left half of the track. Two separate control strategies can be adopted, the first is found by observing the rules used by acrobats balancing a long pole, the second is controlling the rate of change of motion of the cart. The rules are generated when the pole is in region R_0 and the cart is in the left half of the track. For case (e), with the cart is moving towards the centre of the track quickly, the cart must be slowed down and a right force needs to be applied. In case (f), the cart is moving right so its movement needs to be reinforced by applying a left force. In cases (g) and (h), the cart is moving in the wrong direction, therefore a left force is needed to move the cart to the track's centre.

The corresponding rules are shown below:

if (|θ(k)| < threshold
then if (x(k) < 0.0)
then if ((x(k) > x(k-1)) and
(|x(k)-x(k-1)| > |x(k-1)-x(k-2)|)) **then** apply a right force
if ((x(k) > x(k-1)) and
(|x(k)-x(k-1)| < |x(k-1)-x(k-2)|)) **then** apply a left force
if (x(k) < x(k-1)) **then** apply a left force

In the second series of Non-Learning experiments, the human controllers' expertise was tested on a physical pole-cart system.

4.3 Non-Learning (Human) Control of Physical Systems

Trials of human performance were carried out using joy-stick control, via the IBM PC/AT (see Figure 3.2). Also, the same joy-stick that was used in the bang-bang control experiments on the simulator was used again. Five different subjects attempted to control the physical system and their performances were less than impressive. The experimental trials lasted for a period of 2 hours, and no single individual could control the system for more than a few seconds. None of the subjects could balance the pole using heuristic rules because the response time required was so short. The trials that lasted longest were commonly based on random control decisions.

A later set of trials were conducted on a separate physical system constructed from an X-Y plotter, and again using an analog joy-stick to produce the control signal. Over a period of 2 hours approximately 60 trials took place. Trials ended when the cart reached the end of the track, which was short, or when the pole fell beyond 45 degrees. Typical peak trials lifetimes were consistently low, 3-4 seconds, and average lifetimes were lower still, 1-2 seconds. More importantly, over the duration of the experiments no significant improvement of performance was noticed. This was also true of the case where the pole length was extended, a technique used to slow down the physical system response time. From these trials we concluded that the physical systems tested were beyond human control. The

main problem is the speed at which a physical system gets into an irrecoverable state. We suggest that significant changes to the physical system, e.g. the inclusion of damping, and intensive experimentation if human control is to be investigated further.

4.4 Comparative Rule-Based Controller Trials

In this series of experiments the objective was to test the set of rules derived from the human experts control decisions against the theory-oriented control rule derived by Makarovic [13] which is as follows:

if $\dot{\theta} >$ Threshold $\dot{\theta}$ **then** Push Right

if $\dot{\theta} <$ – Threshold $\dot{\theta}$ **then** Push Left

if $\theta >$ Threshold θ **then** Push Right

if $\theta <$ – Threshold θ **then** Push Left

if $\dot{x} >$ Threshold $\dot{x}$ **then** Push Right

if $\dot{x} <$ – Threshold $\dot{x}$ **then** Push Left

if $x >$ Threshold x **then** Push Right

if $x <$ – Threshold x **then** Push Left

His was the bench-mark rule against which we tested our experience derived rules. Trails were carried out once we had encoded these two rules into rule-based controllers, Controller A and Controller B respectively. Each trial started from a random position within the following bounds:

x is in the range –1.0 to 1.0 metres

$\dot{x}$ is in the range –0.2 to 0.2 metres/second

θ is in the range –0.1 to 0.1 radians

$\dot{\theta}$ is in the range –0.2 to 0.2 radians/second

Using our rule, Controller A, the system could be successfully balanced for as long as 200 seconds, the cut-off time in the experiment. In addition, several runs were performed with the cut-off period extended to 2000 seconds. Again, Controller A performed well leading us to believe that the system could be balanced indefinitely using this controller. Comparisons were made with Makarovic's rule, Controller B, derived from examination of the systems' differential equations. To test the robustness of the rules the system parameters were varied in a random fashion to make the task more difficult. Table 1 summarises the results.

It can be seen that Controller B works equally as well as Controller A in the original system configuration. Both Controllers successfully balanced the system 20 out of 20 runs for a period of 200 seconds. However, when system parameters change, the Controller B cannot guarantee success. This was because of the arbitrary choice of threshold values by Makarovic. One set of threshold values is not ideal for a system configuration that alters. In contrast, our rule was written without any threshold values being placed on observation. Here the condition part of the rules only deals with the sign of errors and with the sign of variations of observed system state variables. It reflects the human control heuristics. It can adapt itself to different system configurations and perform consistently well in all circumstances.

System Parameters				Failure Rate		Success Rate	
Mc (kg)	Mp (kg)	L (m)	F (N)	Controller		Controller	
				A	B	A	B
1.0	0.1	0.5	10	0/20	0/20	20/20	20/20
0.5	0.1	0.15	10	0/20	18/20	20/20	2/20
0.5	0.1	0.25	10	0/20	3/20	20/20	17/20
0.5	0.1	0.25	15	0/20	17/20	20/20	3/20

Table 1 Summary of Rule-Based Controller Trials

5. PASSIVE LEARNING EXPERIMENTS

In this section we deal with the introduction of a category of learning that shall be called Passive Learning. By Passive learning here we mean learning through observation, interpretation, and conclusion. An analogy that might be drawn is that of an aeroplane co-pilot. Throughout a period of training the co-pilot learns much from observing the responses of the pilot to given situations, in effect learning the rules being applied in a passive manner. Our interpretation of this situation is twofold. First, we produced a rule to qualitatively partition experimental data, these qualitative patterns were then fed into a neural network to train it as the systems controller. Second, we used rule induction, a technique that generalises from specific examples, in an attempt to refine the human rule used in the construction of Controller A.

5.1 The Back-Propagation Network Theory

In this section, we give a brief review of the Rumelhart semi-linear network on which our controller is based.

Hecht-Nielsen [11] showed that a three-layer neural network was able to represent any continuous mappings from input to output using the Kolmogorov Existence Theorem [23]. However, the introduction of the hidden layer between the input and output layers has made learning in multi-layered (three layers or more) networks difficult. Since the input units are not directly connected to the output units, it is difficult to decide under which circumstance the hidden units should be active and subsequently contribute to the desired input-output mapping. This position has changed since Rumelhart, Hinton and Williams [19] developed a novel algorithm which could give an effective learning procedure for the neural networks with hidden units. A multi-layer neural network of Rumelhart type is illustrated in Figure 5.1. In such a network, the output of units in layer i are multiplied by appropriate weights ω_{ji} which in turn become the input to the next layer. If output of the units in layer i is y_i then the total input to a unit in layer j is:

$$x_j = \sum_i \omega_{ji} y_i$$

The output of a unit in layer j is:

$$y_j = f(x_j)$$

where f is the activation function, or the function that determines the probability with which the unit transits to one-state. The most commonly used activation function is the sigmoidal function, Figure 5.2. The sigmoidal function is expressed in the form:

$$y_j = f(x_j) = \frac{1}{1 + e^{\frac{-\Delta E_j}{T}}}$$

where $\Delta E_j = x_j + h_j$ is an energy gap between the current and the new states, h_j is the internal 'threshold' for unit j, and T is the 'temperature' of the system. Rumelhart's back-propagation learning algorithm adjusts the weights to obtain a minimized system energy function. This is done by repeatedly propagating backward the difference between the actual and desired outputs to the hidden layer. There the weighting of each unit is adjusted in relation to the derivatives of the error function, the ones used to adjust the weights for the output layer units. The error at any output unit in layer k is:

$$e_k = t_k - y_k$$

where t_k is the desired output for that unit and y_k is the actual output. The total error function can be written as:

$$E = \frac{1}{2}\sum_{k}(t_k - y_k)^2$$

The error is minimized by starting with any set of weights and repeatedly changing each weight by an amount proportional to $\frac{\partial E}{\partial \omega}$.

$$\Delta\omega_{kj} = -\varepsilon\frac{\partial E}{\partial \omega_{kj}} = \varepsilon\delta_k y_j$$

where the error signal δ_k is given by:

$$\delta_k = (t_k - y_k)y_k(1 - y_k)$$

at an output unit k, and

$$\delta_j = y_j(1 - y_j)\sum_{k}\delta_k\omega_{kj}$$

at any hidden unit j. The thresholds are learned by taking h_j as being equivalent to weight ω_{j0} thereby connecting the unit j to a bias unit which is always active. As the learning parameter ε tends to zero and the number of updates tends to infinity, the learning algorithm is guaranteed to find the optimised set of weights that gives the least mean square error of the total error function. In practice, a momentum term is added so that the learning rate ε can be increased without causing oscillations:

$$\Delta\omega_{jk}^{n+1} = \varepsilon\delta_k y_j + \alpha\Delta\omega_{kj}^{n}$$

where the momentum parameter α determines the effect of past weights on the current direction of movement in weight space.

5.2 Passive-Learning of Human Control on the Simulated Pole and Cart

In this section, we use a Rumelhart semi-linear network as the basis of a design for a passive-learning controller, a controller that learns by observing human supervised control decisions.

The control process consists of developing pattern formations to give the required motor drive control. The latter is implemented with a connectionist-net of the Rumelhart semi-linear feedforward type. At each instant in time, the values of a training set of the systems state variables are processed into a single pattern which in turn is applied to the input layer of the connectionist net. The response, at the output layer of the net, is used as the control signal for that instant.

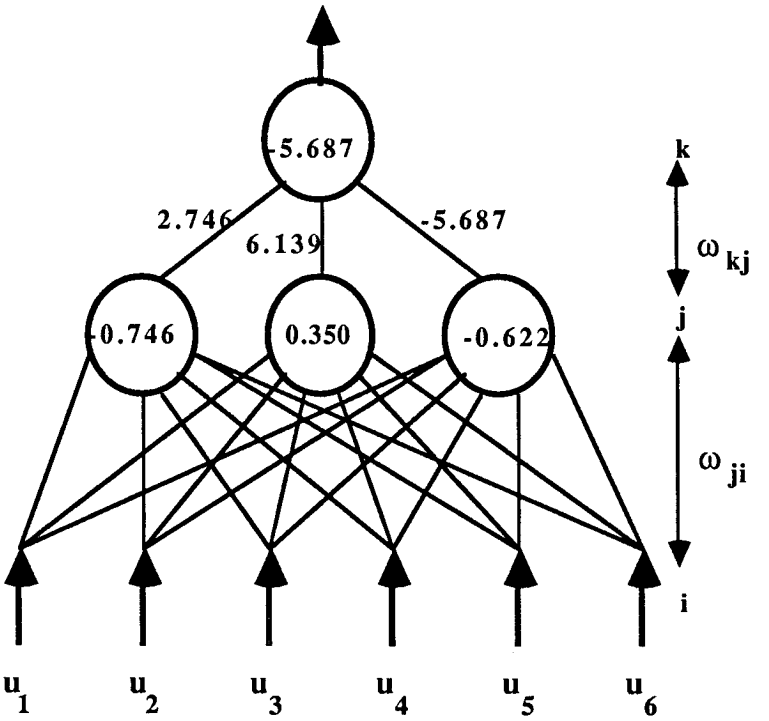

Figure 5.1 The Synthesized Neural-Net Showing Certain Weights and Thresholds

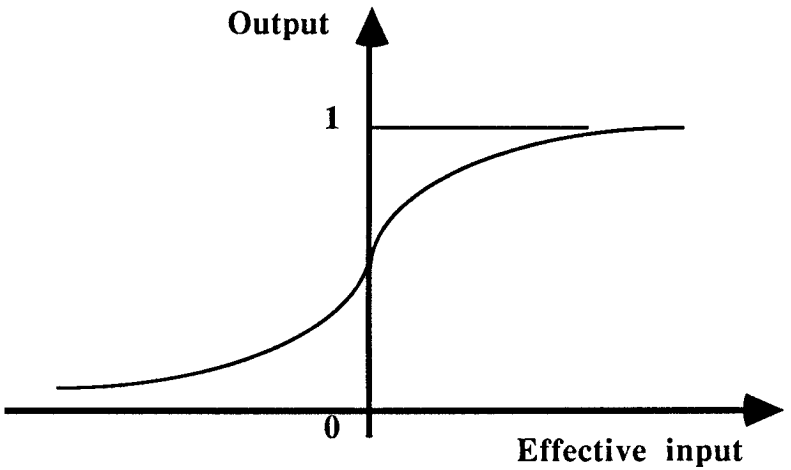

Figure 5.2 Sigmoidal Activation Function

During the learning period, the system is controlled by a human operator, and the neural-net learns to mimic human control by back-propagating the human's decisions through the network and updating the synaptic weights. The neural-net was trained on 8 out of 96 possible input patterns and then tested on the remaining 88. Simulation results show that it generalized correctly and hence can take over the control task from the human teacher. In addition, the behaviour of the trained neural-net produces a set of rules which would be very difficult to elicit from the human operator.

A multi-layer neural network of Rumelhart type is illustrated in Figure 5.1. In our network, the output of units in layer i is multiplied by the appropriate weights ω_{ji}, which in turn become the input to the next layer. Rumelhart's back-propagation learning algorithm adjusts the weights to obtain a minimized system energy function. This is done by repeatedly propagating backward the difference between the actual and desired outputs to the hidden layer. There the weighting of each unit is adjusted in relation to the derivatives of the error function, the ones used to adjust the weights for the output layer units. The latter is implemented with a connectionist-net of the Rumelhart semi-linear feed-forward type. At each instant in time, the values of a training set of the systems state variables are processed into a single pattern which in turn is applied to the input layer of the connectionist net, see Table 2. Using the neural net as a controller, the task is to generate the control decision whether a right or a left force should be applied. The features of the input pattern to the neural net are:

u_1 -- pole angle.
u_2 -- change of pole angle.
u_3 -- variation of change of pole angle.
u_4 -- cart position.
u_5 -- change of cart position.
u_6 -- variation of change of cart position.

Through adopting the sigmoidal activation function, the outputs of the hidden units are constrained between zero and one. The input patterns are chosen to be in the same range so that their associated weights can be adjusted for fast convergence. The value of the input patterns are discrete and fixed so there are only a finite number of inputs, and all pattern apart from u_1 have two values.

There are three values for pattern u_1, one, zero and 0.5 which is the output of the sigmoidal function when the activation value of the unit equals its threshold value. These features are formed by pre-processing the system state variables using the following qualitative partitioning rule where $\theta(k)$ and $x(k)$ are the current value of pole angle and cart position respectively, $\theta(k-1)$ and $x(k-1)$ are the values of pole angle and cart position at last step. This qualitative partition of the system state yields 96 patterns corresponding to all of the combinations of the feature values.

The output of the neural net can be considered as the probability of applying a 'right' force. An output of '1' indicates applying a right force (the probability of applying a 'right' force is 1.0). An output of '0' denotes a 'left' force (the probability of applying a 'right' force is 0.0).

```
if (θ(k) > THRESHOLD) u1 = 1.0
if (θ(k) < -THRESHOLD) u1 = 0.0
if (|θ(k)| < THRESHOLD) u1 = 0.5
if (|θ(k)| > |θ(k-1)|) u2 = 1.0
      else u2 = 0.0
if (|θ(k) - θ(k-1)| > |θ(k-1) - θ(k-2)|) u3 = 1.0
      else u3 = 0.0
if (x(k) > 0.0) u4 = 1.0
      else u4 = 0.0
if (|x(k)| > |x(k-1)|) u5 = 1.0
      else u5 = 0.0
if (|x(k)-x(k-1)| > |x(k-1)-x(k-2)|) u6 = 1.0
      else u6 = 0.0
```

From Figure 5.1 it can be seen that our neural network has six input units, three hidden units and one output unit. Experiments were conducted in three phases, the knowledge acquisition phase, the learning phase or knowledge transfer phase and the controlling phase. Phase one consisted of data being acquired through observing and recording a skilled human controlling the simulated system. In the knowledge acquisition phase this training data was transformed into the training patterns that were to be used in the learning phase.

The training data was presented in groups of three, and input-output patterns are formed by processing the data in each group. During the second phase of the experiment, the neural-net was trained using the training patterns obtained in the first phase. The neural-net learned all eight patterns in an autonomous manner, in other words, the human's knowledge was successfully transferred to the neural-net. Comparisons of the actual outputs with the desired outputs are listed in Table 2. Finally, the remaining eighty-eight patterns in the problem space were presented to the neural-net. The training was done off-line and it took about 10 minutes on a Sun 3 machine. The values of certain weights and thresholds, autonomously acquired by the neural-net through training, are illustrated in Figure 5.1.

Since the values of the output are the probabilities of applying a 'right' force, it was seen that the neural-net could successfully discriminate between 'left' category input patterns where the output of the net is less than 0.5 and 'right' category input patterns where the output of the net is greater than 0.5. Of particular importance to us is the ability of the neural-net could passively learn a control law which is embedded in the human operator. From our observations we can express a neural-net rule as follows:

Data Group and Pattern No.	System State Variables				Human Control Action	Pattern Features						Desired Output	Actual Output
	x (m)	θ (rad)	$\dot{x}$ (m/s)	$\dot{\theta}$ (rad/s)		u_1	u_2	u_3	u_4	u_5	u_6		
	-0.016	-0.071	0.331	-0.182	left								
1	-0.009	-0.074	0.137	0.086	left	0	0	0	0	0	0	0	0.026
	-0.006	-0.073	-0.056	0.354	left								
	0.026	-0.084	0.326	-0.07	left								
2	-0.020	-0.086	0.132	0.185	left	0	0	1	0	0	0	1	0.944
	-0.017	-0.082	0.061	0.449	right								
	-0.037	-0.078	0.520	-0.346	left								
3	-0.026	-0.084	0.326	-0.079	left	0	1	0	0	0	0	0	0.007
	-0.020	-0.086	0.132	0.185	left								
	-0.238	-0.071	-0.140	0.150	right								
4	-0.241	-0.068	0.055	-0.163	left	0	1	1	0	0	0	0	0.045
	-0.240	-0.073	-0.138	0.106	left								
	-0.244	0.080	0.527	-0.492	left								
5	-0.233	0.070	0.331	-0.175	left	1	0	0	0	0	0	1	0.945
	-0.227	0.066	0.134	0.138	right								
	-0.205	0.083	-0.122	-0.216	right								
6	-0.208	0.078	0.071	-0.481	left	1	0	1	0	0	0	0	0.056
	-0.206	0.069	-0.125	-0.164	left								
	-0.233	0.070	0.331	-0.175	left								
7	-0.227	0.066	0.134	0.138	right	1	1	0	0	0	0	1	0.959
	-0.206	0.069	0.328	-0.132	right								
	-0.397	0.055	0.327	-0.103	left								
8	-0.390	0.053	0.131	0.206	right	1	1	1	0	0	0	1	0.959
	-0.388	0.057	0.325	-0.069	right								

Table 2 Neural Network Training Patterns

if pole angle is greater than the threshold
then
if pole angle is increasing
then apply a right force;
if pole angle is decreasing and the variation of pole angle is increasing
then apply a left force;
if pole angle is decreasing and the variation of pole angle is decreasing
then apply a right force;
if pole angle is less than the threshold
then
if pole angle is increasing
then apply a left force;
if pole angle is decreasing and the variation of pole angle is increasing
then apply a right force;
if pole angle is decreasing and the variation of pole angle is decreasing
then apply a left force;
if pole angle is within the threshold
then
if cart is on the right half of the track
then
if cart position is increasing
then apply a right force;
if cart position is decreasing and the variation of the cart position is increasing
then apply a left force;
if cart position is decreasing and the variation of the cart position is decreasing
then apply a right force
if cart is on the left half of the track
then
if cart position is increasing
then apply a left force
if cart position is decreasing and the variation of the cart position is increasing
then apply a right force
if cart position is decreasing and the variation of the cart position is decreasing
then apply a left force

The Neural Net Rule

5.3 Passive-Learning from Human Control of the Physical Pole-Cart System

The objective of this section was to see if passive-learning could be achieved observing the human responses in the control experiments of section on the human control of the physical pole and cart system. Since the physical system was uncontrollable by a human it is considered here that no passive-learning could take place. It was therefore decided to conduct experiments on the physical system using the rule-based controllers, derived from experiments carried out on the simulator. It was also decided to introduce inductive learning here, first as a means of verifying the human rules and, second as a potential route to refining the human rules. This is possible because inductive inference automatically generates rules based on classification from presented attributes, in our case the attributes from the qualitatively partitioned state-space.

5.4 Inductive Inference and Inductive Learning

In the field of machine learning much attention has been given over recent years to the use of inductive inference, and in particular the use of inductive learning for expert system development [4, 9, 12, 17]. Inductive inference is used for generalising from specific examples. In our work we applied the c4.5 algorithm, one of Quinlans' [17] ID3 family, to our probabilistic data to induce a decision tree that is representative of the relationship between the attributes and the classes. The c4.5 induction algorithm was chosen because it is based on both statistical and information theoretic techniques, as such it is suited to handling probabilistic data of the type we were dealing with. This algorithm is also suited to handling 'noisy' data.

The data samples given to the algorithm were obtained from an experimental trial, each sample was described in terms of the attribute values, in our case linear and angular displacement, rate of change of linear and angular displacement, and a class label. Since c4.5 uses a hierarchical mutual information approach it yields a 'top-down' tree that maximises the mutual information gain at each partitioning step. Put simply, it gives valuable information on the relevent importance of the various attributes which can be an advantage over more conventional classification techniques. In common with such algorithms Shannon's entropy (H) of a class random variable (C) is used as a measure of information, and is defined as:

$$H(C) = \sum_{i=1}^{n} p_i \log_2 (1/p_i)$$

If there are N attributes each denoted by A_i, where $1 \leq i \leq N$. Also, if we consider that each attribute A_i has n_i values, and that n_i is finite, then $I(C; A_i)$ is the mutual information between C and A_i according to Shannon. So, the 'top-down' tree classifier algorithm recursively seeks to determine whether a node is a 'leaf', or not, and if not it determines which attribute provides the most information at that node.

5.5 Passive-Learning from Automatic Control of the Physical System

The results obtained from the trials conducted in section on the automatic control of the physical system, for the automatic controllers based on the rules of Controller A and Controller B, were processed by an induction rule generator. The rule generator used in these experiments was c4.5 because it accepted data directly in the form that it had been logged in the trials. In the case of the trial using Controller A, 1704 items of training data were used, while in the Controller B 1601 items of data was used. The rule generator c4.5 was set up with a minimum of 10 items on either side of the threshold test, and with a pruning confidence level of 1%. With this arrangement the results proved very different. The decision tree generated from the Controller A experiments produced a tree of 15 nodes from the three attributes considered. This tree is considerably larger than the hand-crafted rule tree developed by Controller A, which only had 8 nodes. The generated tree from the Controller B experiments was closer to the Controller B rule. In this instance c4.5 produced a tree with 5 nodes from 4 attributes, 3 nodes less than his hand-crafted rule. At this stage these results are reported on as an demonstration of an ability to passive-learn rules. The technique requires further examination and experimentation before more focussed conclusions can be drawn.

6. MACHINE-LEARNED CONTROL (MLC)

In this section, we propose an application of the rule to control the pole-cart system by using machine learning techniques.

6.1 'BOXES'

Very briefly, the 'BOXES' learning algorithm [14] is an adaptive controller that develops its control strategy on a vector based on the system's state at any instant, or, on the receipt of failure signal after the system has gone out of control. Should a failure occur the system is re-set, in a random manner, then experimentation continues. Using the reference text as a guide, we will try here to give an overview of the learning mechanism as presented by Michie and Chambers [14]. First the *n*-dimensional state-space is partitioned into the *n* representative variables first given in a previous section, see Control Strategies and Experiments, section4.

Here, the terms for linear and angular velocity can be found by differencing their associated displacement terms with respect to a time interval. Being confronted with an infinite number of 4-element vectors that were available in the state-space Michie and Chambers adopted a novel quantisation technique to reduce the time that the algorithm needed to calculate the required control signal. The 'model' they adopted is shown in Figure 6.1, close examination shows that they set thresholds for the variables, 5 'boxes' each for the displacement terms, 3 'boxes' each for the velocity terms, a total of 255 'boxes' in all.

The decision making process for the 'BOXES' algorithm is best understood if we consider that each of the 255 'boxes' is inhabited by a local demon, a control switch, and a score-board. All the local demons are overseen by a single global

demon. Local demons make their left and right control decisions based on experience, a weighted sum of the 'life' of his 'box' (LL and RL), and a weighted sum of the 'usage' of his 'box' (LU and RU), Figures 6.2, 6.3. A local demon's control decision, the setting of his switch (S), is therefore a function of 'life', 'usage', and TARGET, Figure 6.4. This is the basis of the 'BOXES' algorithm. In its original form, the ability of 'BOXES' to learn was primarily determined by the quantisation of the state-space. Although Michie and Chambers did say in their paper that boundary adaption, 'lumping' and "splitting" as they put it, would be a desirable extension to their work it was never implemented. In part, the work described in this Chapter deals with that particular extension to their work.

6.2 Learning to Partition the State Space

In Michie and Chambers learning algorithm 'BOXES' [14], the physical state space is partitioned into boxes. The algorithm learns to set correct decisions for each box through trial-and-error. Unfortunately, state space partitioning prior to experimentation is arbitrary because it is reliant on human knowledge. If the original partitioning is wrong, the algorithm can not learn to correct it. In the following, we will show how our rule can be used to partition the state space in pole-cart application.

Each 'box' contains a setting which is either 'left' or 'right' and a record of any changes made. A decision is made by the rule whether the last move is good or bad. If it is a good move, then the setting S_i in box_i , which contains the system states in the last step will remain unchanged. The same move will be repeated in box_j , the box which contains the current system state. Otherwise, the alternative setting will chosen both for the box_i and box_j . The number of setting changes in each box is recorded. If this number is greater than two, this means that box boundaries have been set badly; the control decision is either left or right. This information is used as a signal to trigger a subdivision of the box. This is achieved by subdividing the interval within which one of the state variables fell in the preceding step. Our partitioning approach is similar to binary search.

The subdividing should only affect the box concerned, box_n, so after splitting it is divided into n+1 sub-boxes. Thereafter, n more boxes are added to the total, where n is the number of dimensions of state space. Alternatively, when two or more adjacent boxes are found to agree on the same setting, they can lumped into a single box. Figure 6.5 illustrates the control system. The decoder transforms each state vector into a box, this contains the current system state. The entries in the decision table, which are indexed by the boxes, corresponds to the settings in the boxes. These settings are either left or right. A supervisor or learning automaton learns to set the correct entries for the decision table by memorising past states and evaluation of current system performance. The decision table is dynamic and can be expanded or contracted by the learning automaton during the trial. The learning automaton also sets the correct function for the decoder, which means that for each box decoded there exists only one possible decision.

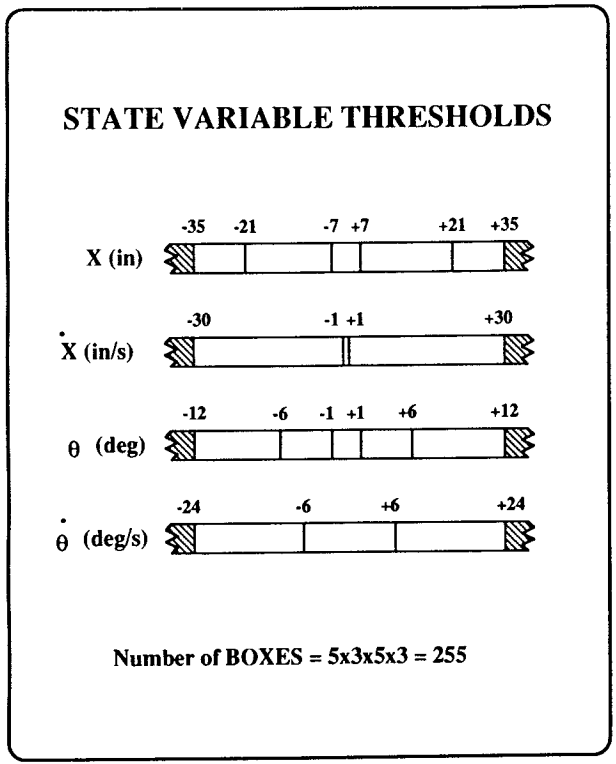

Figure 6.1 State Variable Threshold

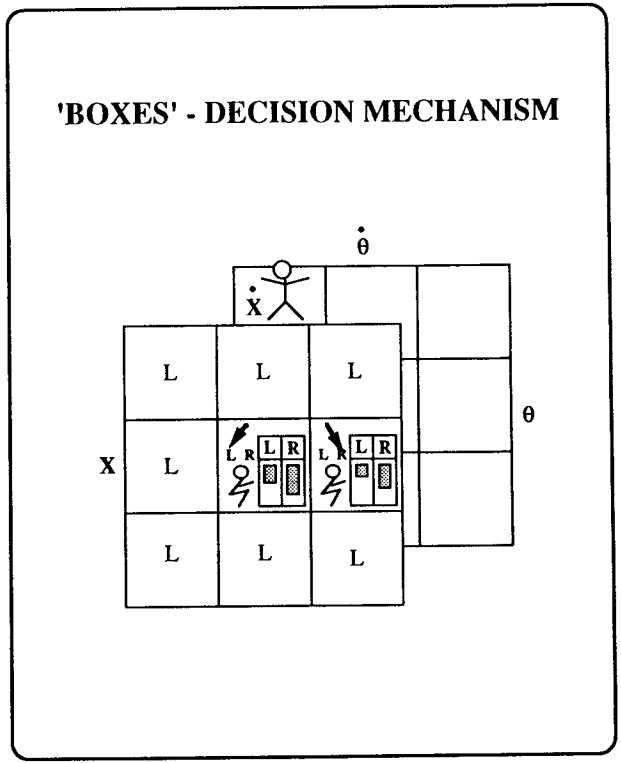

Figure 6.2 'BOXES' - Decision Mechanism

THE 'BOXES' SUPERVISED LEARNING MECHANISM

- LL & RL ("left-life" and "right-life")
  The weighted sum of the "lives" of the appropriate direction decisions taken on entry to a box.
- LU & RU ("left-usage" and "right-usage")
  The weighted sum of the appropriate direction decisions during previous runs.
- TARGET
  A constant multiple of the current mean life of the system. This is supplied by the supervising demon.
- TIME (T_1 , T_2, ... T_N)
  Times at which a box has been entered during the current run. The time between the start of the run and the entry to the box.

Figure 6.3 Supervised Learning Mechanism

DEMONS - GLOBAL & LOCAL

- Local Demon Totals (consider S decision "left")

$$LL = (LL' \times DK) + \sum_{I=1}^{N} (T_F - T_i)$$

$$LU = (LU' \times DK) + N$$
$$RL = RL' \times DK$$
$$RU = RL' \times DK$$

- Global Demon Totals

$$GL = (GL' \times DK) + T_F$$
$$GU = (GU' \times DK) + 1$$

$$Merit = GL/GU$$

$$Target = (C0 + C1) \times Merit \qquad (C0 >= 0;\ C1>=1)$$

- Control Action (S = "left" or "right")

$$Value\ (L) = \frac{(LL + K) \times Target}{(LU + K)}$$

Similarly, Value(R) is calculated. Control action S is decided depending on whether:

$$Value\ (L) <> Value\ (R)$$

Figure 6.4 Local and Global Demons

6.3 A Neural-Net MLC Controlling a Simulator

The third phase of the experiments carried out was the machine learned control (MLC) phase, could the neural-net learn sufficiently to become an MLC and take over the control task from the human teacher. Over 20 balancing runs under the control of the neural-net it was found that the neural-net successfully balanced the pole-cart system for as long as two hundred seconds, the simulation cut-off time. Several runs were performed for up to two thousand seconds, and there was an indication that the system can be balanced 'for ever'!

To test the robustness of the neural controller its performance was tested against that of the performances of two other automatic controllers, one based on the 'BOXES' learning algorithm, and the other based on Makarovics' Rule. The test was to see how they coped with varying system parameters, Table 3 summarizes the results. Success is measured by a system controlling the pole-cart simulation 200 seconds. From Table 3, it can be seen that the neural controller has successfully balanced the system for all test runs. This rule learned by the neural-net has been compared with the rule learned by 'BOXES' algorithm. The latter was derived by Sammut taking the output of a successful 'BOXES' trial as input to Quinlan's c4 induction program [15]. Table 3 shows a comparison of the results of the two different methods.

It can be seen that the 'BOXES' learned rule work equally as well as our neural-net learned rule using the original system configuration, both successfully balanced the system 20 out of 20 runs for a period of 200 seconds. However, when the system parameters change our neural-net rule adapts itself to system parameter changes and continues to perform consistently well in changed circumstances.

The neural-net uses a rule that is similar to that of the human pre-processed rule, Controller A or Controller B. However, we discovered that care must be taken during the pre-processing stage, the preparation of the training patterns for the net, too much intervention here can lead to biased learning. To ensure un-biased learning the net must be made to generate the training patterns automatically from the human responses as it is acquired. To ensure that no bias had occurred we conducted a series of trials where automatic pattern generation was the goal, some of the weights and thresholds are included in Figure 5.1. Once this neural-net had been trained a series of performance trials were conducted to see how it performed against a 'BOXES' controller and Controller B. The results of these tests is shown in Table 3. One point noted was that if θ is small and x is large the neural-net controller is not so exact as the rule derived by human observation.

7. CONCLUSIONS

This Chapter has dealt with how best to use artificial intelligence techniques in control engineering. The work reported on included technical issues such as human-supervised learning, passive-learning, and machine learning, and how best they can be applied to control engineering problem, pole-balancing. However, in

```
if θ > 0.105 then right
if θ̇ > 0.089 then right
if θ̇ > −0.870 then left
if θ < 0.015 then
                    if θ̇ > −0.875 then right
                    if ẋ < 0.565 then left
                    if θ < −0.11 then left
                    if θ < 0.005 then right else left
               else
                    if x > −0.795 then right
                    if ẋ < −0.475 then right
                    if θ < 0.095 then left
                    if ẋ < −0.39 then right
                    if ẋ < −0.375 then left else right
```

The Humanly Derived Rule

extensions to the work here we have used the same techniques to examine rocket control, others have used the techniques for manufacturing processes .

Initially, our work work dealt with the building a computer simulation of the pole-cart system. Once this was completed and a graphical interface had been constructed the simulation was used as a test-bed for ideas throughout the duration of our work. To ensure maximum flexibility these simulators were developed for both PC-based machines (IBM PC-AT and Compaq 386/20), and Sun workstations. At an early stage joy-stick control experiments were established on the PC's, this allowed human-supervised learning trials to begin. Through conducting these trials a better understanding of human control strategies was obtained, particularly in relation to human response time when controlling a highly dynamic system.

Once a suitable delay had been found for the simulator a series of trials was undertaken to assess human performance, the abilities varied considerably but the observations made were useful for inclusion in later non-learning and learning control work. Human-supervised learning became a major study area for others at the Turing Institute.

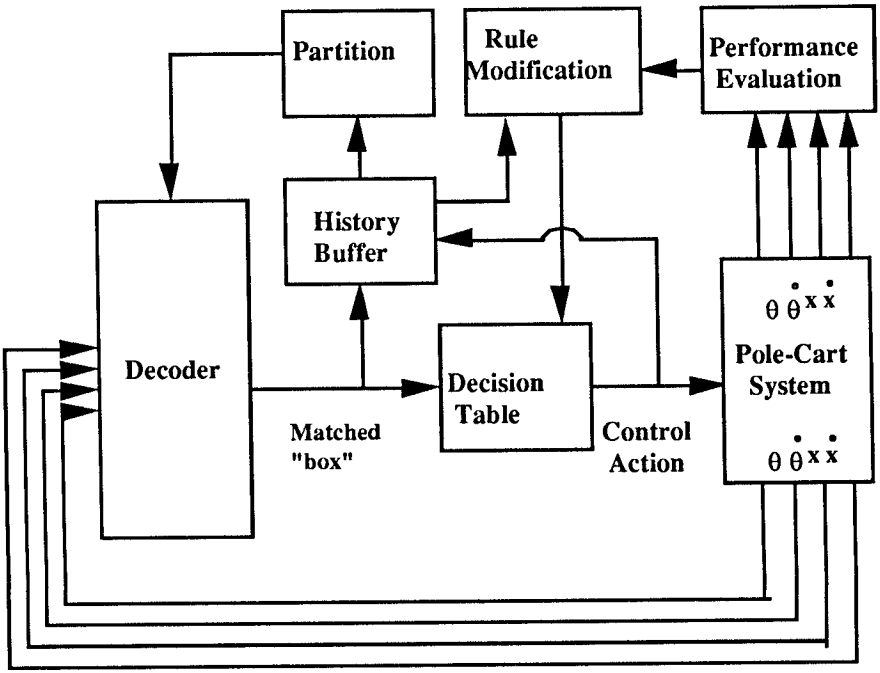

Figure 6.5 A Machine Learning Controller Based on 'BOXES'

In the next phase automatic controllers were developed, rule-based controllers, where rules were derived from observing human experts controlling the pole-cart simulation, or from interpretation of the systems dynamic equations. The human-supervised control data collected proved useful not only for developing a rule-based automatic controller, it was also used later when the neural-net controller was constructed. The rule, which became known as Controller A, was based solely on a control decision based on pole angle. The properties of Controller A have been compared with that of another rule-based controller, one where the rule, Controller B is derived from theory. Experiments conducted in simulation showed that our controller was more efficient when compared to the performance of the one based on theory, especially when the system parameters are gradually changed. That is, when both controllers were "tuned" to the dynamic system they performed equally well, but when the physical parameters were altered, tests showed Controller A adapted, Controller B did not. A more appropriate test was to be performed later, on the physical pole-cart apparatus.

In this Chapter we also show how the data obtained from human-supervision of a pole-cart simulator can be used for a connectionist-net machine learning approach.

Further this neural-net controller can be applied to achieve intelligent control. A trainable adaptive controller has been built to control a simulated mechanical apparatus, the pole-cart system. The controller is implemented with a three-layer neural network, based on the work of Rumelhart, whose intermediate layer contains three hidden units. Unlike previous neural-net efforts, in which the neural controllers learn to control through 'trial-and-error', our neural controller learns to control through training. That is, it initially observes a file of stored decisions made by an experienced human operator controlling the simulated pole-cart system. Then it emulates these actions. After a period of training, it is possible for the human operator to exit the control loop and allow the neural controller to take complete control. As in the 'BOXES'-based experiments of Chambers and Michie this property of learning without failure shows promise for the development of trainable adaptive computer-based controllers, where a computer is trained to control rather than being programmed to control. When a complete mathematical model of a complex dynamic is out of the question, which is often the case in real-world applications, control by learning shows promise and is worthy of being considered as the controller choice in these instances.

System Parameters				Rule Failure			Rule Success		
Mc [kg]	Mp [kg]	L [m]	F [N]	N-N	BO-XES	Contr-oller B	N-N	BO-XES	Contr-oller B
1.0	0.1	0.5	10	0/20	0/20	0/20	20/20	20/20	20/20
0.5	0.1	0.5	10	0/20	11/20	2/20	20/20	9/20	18/20
0.5	0.1	0.25	10	0/20	20/20	3/20	20/20	0/20	17/20
0.5	0.1	0.15	10	0/20	20/20	17/20	20/20	0/20	3/20
0.5	0.1	0.25	15	0/20	20/20	18/20	20/20	0/20	2/20

Table 3 Comparison of the Neural-Net Learned Rule with the BOXES Learned Rule and Controller B Rule

All of the above work was undertaken in simulation because of a lack of a physical test-bed, only recently has this been rectified. The limited series of testing that has taken place on the physical pole-cart system has provided invaluable information.

For example, under human-supervised control with a joy-stick the system is uncontrollable! When the automatic control trials were carried out with the two rule-based automatic controllers neither performed outstandingly well. Further, when the data from these trials was processed by an inductive-rule generator, for passive-learning, the number of rules generated from the Controller B trials was closer to the hand-crafted rules than that of Controller A.

The MLC experiments, like the passive-learning experiments above, at are an early stage of development. Initial trials with the 'BOXES' learning algorithm showed improved performance over the trials conducted, at this point it is too early to comment on the efficiency and effectiveness of MLC. An extensive period of experimentation is required, this should include a comparative study using other learning algorithms as the basis of the MLC, and included in this should be further work on neural-net controllers. The physical system is an ideal test-bed for passive-learning and MLC work. The trials that were conducted highlighted issues that are not apparent when operating in the simulated world. True, the simulated world can have the time delays removed allowing them to run as fast as the physical system. But, simulation does not have to address problems such as sensor resolution. In the results recorded it was noticed that only certain *boxes* were visited in the state-space, this was due entirely to the resolution of the rotary encoder's. This continual visiting of the same *boxes* caused an computer register overflow resulting in negative values appearing in the *boxes*. It was concluded that this register negation could effect MLC performance, trials are continuing.

An interesting piece of MLC work was having the simulator running on the PC and the MLC running on the Mac IIx. Although this was purely an exercise in connectivity it has provided a graphic illustration of how controllers may be resident in one machine while the process is hosted by a completely different machine. At this stage this has only been used for demonstration, no MLC trials have been conducted. When complete mathematical modelling of an automated system is out of the question, which is often the case, *learning* may be the only choice. Research is continuing towards building a trainable adaptive controller. This work see a new kind of computer control developed, one where the computer is *trained-to-control* rather than being programmed to control.

8. ACKNOWLEDGEMENTS

This study was jointly funded by the United Kingdom's National Engineering Laboratory, British Aerospace(Hotol), British Telecom Research Laboratories, Apple Computers (Europe), the Science and Engineering Research Council, and the British Council. We also acknowledge the support of the Directors of the Turing Institute Limited, and the Department of Computer Science at the University of Strathclyde. Thanks are due to Professor Donald Michie for contributions and comments. We acknowledge useful suggestions from Dr. Andrew Hay, formally of the National Engineering Laboratory and the helpful and constructive contributions of fellow researchers, Dr Bing Zhang and Mr. Michael Bain.

9. REFERENCES

[1] Anderson, C.W., "Strategy Learning with Multilayer Connectionist Representations", In: *Proceedings of the Fourth International Workshop on Machine Learning*, Pat Langley (ed.), Morgan Kaufmann, Los Altos, (1987).

[2] Astrom, K. J., Anton, J. J. and Arzen, K. E., "Expert Control", In: *Automatica*, Vol. 22, No. 3, pp 277-286, (1986).

[3] Barto, A. G., Sutton, R. S. and Anderson, C. W., "Neuronlike Elements That Can Solve Difficult Learning Control Problems", In: *IEEE Trans. on Systems, Man, and Cybernetics*, vol 13, pp 835-846, (1983).

[4] Bloomfield, B. P., "Capturing Expertise By Rule Induction", In: *The Knowledge Engineering Review*, Vol 2, Part 1, March, Cambridge University Press, pp 55-63, (1987).

[5] Chambers, R. A. and Michie, D., "Man-machine Co-operation on a Learning Task", In: *Computer Graphics: Techniques and Applications*, R. Parslow, R. Prowse and R. Elliott-Green (eds.), London:Plenum, (1969).

[6] Connell, M. E. and Utgoff, P. E., "Learning to Control a Dynamic Physical System", In: *Proceedings of the Sixth National Conference on Artificial Intelligence*, Morgan Kaufmann, Los Angeles. (1987).

[7] Donaldson, P. E. K., "Error Decorrelation: A Technique for Matching a Class of Functions", In: *Proceedings of the Third International Conference on Medical Electronics*, pp 173-178, (1960).

[8] Feng Zuren, Yin Zhengqi and Chen Huitang, "Stabilization of a Double Inverted Pendulum by Analogue Controller", In: *Proceedings of the 9th World IFAC Congress*, Budapest, Hungary, (1984).

[9] Goodman, R. M. & Padhriac, S., "Decision Tree Design Using Information Theory", In: *Knowledge Acquisition*, Vol 2, Part 1, March, Academic Press Ltd., pp 1-19, (1990).

[10] Grant, E. and Bing Zhang, "A Neural Net Approach to Supervised Learning of Pole Balancing", In: *Proceedings of the Fourth International Symposium on Intelligent Control*, 25-27 September, Albany, New York. (1989).

[11] Hecht-Nielsen, R., "UCSD Extension Class Notes", (1986).

[12] Kirkwood, C. A., Andrews, B. J. & Mowforth, P. H., "Automatic Detection of Gait Events: A Case Study Using Inductive Learning Techniques", In: *J. Biomed. Eng.*, Vol 11, November, Butterworth and Co. (Publishers) Ltd., pp 511-516, (1989).

[13] Makarovic, A., "A Qualitative Way of Solving the Pole-Balancing Problem", In: *Machine Intelligence 12*, J.E. Hayes, D. Michie and E. Tyugu (eds.), Oxford, Oxford University Press, (1989).

[14] Michie, D. and Chambers, R. A., "BOXES: An Experiment in Adaptive Control", In: *Machine Intelligence 2*, E. Dale and D. Michie(eds.), Oliver and Boyd, Edinburgh: Edinburgh University Press. (1968).

[15] Michie D. and Bain M., "Machine Acquisition of Concepts from Sample Data", Artificial Intelligence and Intelligent Tutoring Systems: 1989 Spring Symposium at the University of Maine, (1989).

[16] Mori, S., Nishihara H. and Furuta K., "Control of Unstable Mechanical System: Control of Pendulum", *International Journal of Control*, Vol 23, No. 5,

pp. 673-692, (1976).
[17] Quinlan, J. R., "Discovering Rules By Induction From Large Collections Of Examples", In: *Expert Systems in the Microelectronics Age*, D. Michie (ed.), Edinburgh University Press, pp 168-201, (1979).
[18] Reynolds, D. E., Boulton, C. B. and Martin, S. C., "AI Applied to Real-Time Control: A Case Study", In: *Proceedings of Conference on Applications of AI to Engineering Problems*, (1986).
[19] Rumelhart, D.E., Hinton G.E. and Williams R.J., "Learning Internal Representations by Error Propagation", In: *Parallel Distributed Processing: Exploring the Microstructure of Cognition Volume 1: Foundations*, D.E. Rumelhart and J.L. McClelland (eds.), Cambridge, MA, Bradford Books/MIT Press, (1986).
[20] Sammut, C., "Experimental Results From An Evaluation of Algorithms That Learn to Control Dynamic Systems", In: *Proceedings of the Fifth International Conference on Machine Learning*, J. Laird (ed.), San Mateo: Morgan Kaufmann, (1988).
[21] Schaefer, J. F. and Cannon, R. H., "On the Control of Unstable Mechanical Systems", In: *Proceedings of the 3rd World IFAC Congress*, (1966).
[22] Selfridge, O.G., Sutton, R. S. and Barto, A. G., "Training and Tracking in Robotics", In: *Proceedings of the Ninth International Conference on Artificial Intelligence*, Morgan Kaufmann, Los Altos, (1985).
[23] Simpson, P. K., *Naval Ocean Systems Center Technical Report*, Code 441, (1987).
[24] Valmiki, A., West, A. A., & Williams, D. J., "The Evolution of a Neural Network for Adhesive Dispensing", In: *Proceedings of IFAC Workshop on Computer Structures Integrating AI/KBS Systems in Process Control*, May 29-30, pp 93-101, (1991).
[25] Widrow, B. and Hoff, M. E., "Adaptive Switching Circuits", In: *1960 WESCON Conv. Record*, part 4, pp 96-104, (1960).