FUZZY BOXES: A DISTRIBUTED ADAPTIVE NEUROCONTROLLER USING REINFORCEMENT LEARNING

S Marriott and R F Harrison

Department of Automatic Control and Systems Engineering
The University of Sheffield,
Mappin Street, Sheffield, S1 3JD, U.K

Research Report No. 682

August 1997

FUZZY BOXES: A DISTRIBUTED ADAPTIVE NEUROCONTROLLER USING REINFORCEMENT LEARNING

S Marriott and R F Harrison

Department of Automatic Control and Systems Engineering
The University of Sheffield,
Mappin Street, Sheffield, S1 3JD, U.K

ABSTRACT

A modified reinforcement learning architecture is presented here as an extension of the seminal implementation of Barto, Sutton and Anderson and is applied to a well-known control task. The motivation is to improve the performance of the original system by distributing state information across state-space. By fuzzifying the fixed state-space boundaries of the original system and modifying the learning algorithm, both the learning-rate and control performance have been improved. A further benefit of this system is that a set of fuzzy rules for the control task is generated automatically.

Key Words: Reinforcement Learning, Neurocontrol, Fuzzy Control.

1. INTRODUCTION

This paper presents an extension of the seminal implementation of reinforcement learning (RL) of Barto, Sutton and Anderson [1]. By fuzzifying the RL system, improvements in learning rate and control action can be achieved. Furthermore, a set of fuzzy rules is generated which specify the resulting controller. The fuzzy RL system has been given the name "FUZBOX" to distinguish it from the original BSA system (Marriott, [2]).

Following Michie and Chambers [3] and Barto et al [1] the cart-pole system (inverted pendulum) is used to exemplify some aspects of neurocontrol and is a highly non-linear system involving the characterisation of complex state-space trajectories. A computer simulation of the cart-pole system (including friction effects) is used (Barto et al [1]). Information from the simulation is minimal; only the state vector and a coarse failure signal are supplied. If either the pole or the cart exceeds pre-set boundaries then a failure signal is sent to the neurocontroller and a new trial is begun from the initial conditions.

2. REINFORCEMENT LEARNING

Reinforcement learning (RL) (Barto et al [1], Sutton [4], Sutton et al [5]) arose out of earlier work on classical conditioning (Sutton and Barto [6], Barto and Sutton [7]). In its simplest form, RL consists of using a single scalar variable to indicate the performance of an artificial neural system. This signal is generated by a "critic" which rewards favourable system responses and punishes undesirable ones. Earlier work, known as "BOXES" (Michie and Chambers [3]), was entirely failure driven. The system considered here is the seminal implementation of Barto, Sutton and Anderson (BSA) (Barto et al [1]) which consists of an associative search element (ASE) and an adaptive critic element (ACE), see Figure 1.

The BSA implementation of an RL-based controller uses a fixed state-space partitioning of 162 distinct regions or boxes. A decoder assigns a unique output line to each state-space region. During processing, a state vector enters the decoder which activates the appropriate input line to the ASE and subsequently issues a control action. Depending upon the outcome and a prediction of future reinforcement, the information representing a region traversed in state-space is updated.

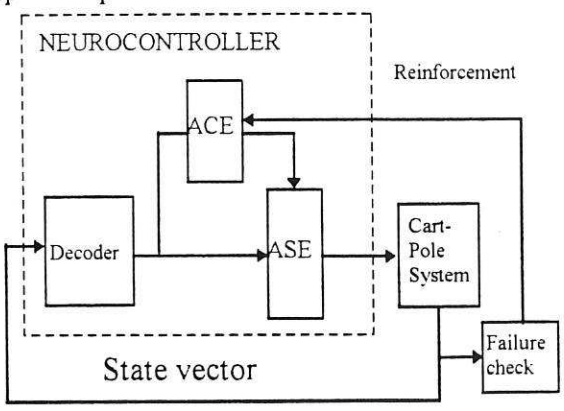

Figure 1. The ASE / ACE reinforcement learning system of Barto et al [1] See text for details

The choice of state-space regions was "...based on specific knowledge of the control task" (Barto, Sutton and Anderson, [1]). Replication experiments show that the original system is sensitive to changes in partitioning (Marriott, [2]).

3. A FUZZY STATE-SPACE DECODER

The decoder is a neurocontroller subsystem which lends itself to useful modification by allowing the properties of the controller to be altered whilst retaining the functionality of the ASE/ACE sub-units. Various methods of state-space partitioning become possible (Marriott and Harrison, [8], [9], [10]), including fuzzification as explored in this paper. Thus, the non-overlapping partitioning of state space is a sufficient but not necessary condition for using the ASE/ACE system.

The Cerebellar Model Articulation Controller (CMAC) of Albus [11], [12] has been used as a state-space decoder for the ASE / ACE system (Lin and Kim, [13]). This allows some generalisation within the neurocontroller which improves learning. However, fuzzification of the existing BOXES decoder in the BSA implementation appears to be a more natural extension of the original work and readily lends itself to adaptive fuzzy rule-base generation.

In addition to distributing state-information across the state-space decoder, both the ASE and ACE have been modified to combine rules from the rule-base rather than to select a single rule (state-space location) as did the original system. The FUZBOX architecture demonstrates that distribution of both the ASE and ACE modules is indeed possible, and that learning—in the case of FUZBOX—is accelerated compared with the original BSA system. Furthermore it produces a set of fuzzy rules which are open to semantic interpretation.

4. THE FUZBOX IMPLEMENTATION

Fuzzy systems are a natural choice for developing a prototypical distributed system owing to their graded membership functions. Furthermore, the use of a rule-base is a natural extension of the BOXES concept where the boxes form a crude rule-base in the original non-distributed formulations (Michie and Chambers, [3]; Barto, Sutton And Anderson, [1]). Like the BSA, system, the FUZBOX system is also based upon treating state boxes as rules and using fuzzy membership to distribute learned information. Bang-bang control is retained using a special case of the Sugeno method (Sugeno and Nishida, [14]) where linguistic variables are not used at the output. The Sugeno method is used to retain compatibility with the original BSA system and its bipolar output.

The maximum possible number of rules for this particular configuration of FUZBOX is 625 which is determined by the use of four state variables and five linguistic variables for each of the state intervals. Each of the 625 possible rule antecedents is assigned a single output value only.

The relevance of a rule for a given input is measured by the *rule antecedent strength* (RAS) (Marriott, [2]) which combines the membership values of each state variable belonging to the fuzzy set associated with each linguistic variable. Rules are added incrementally if the hash code of the rule with the highest possible RAS indicates a non-existent rule. This is to ensure that previously encountered state-space regions are represented in the rule-base by at least the most relevant rule. A RAS threshold is chosen such that all existing and new rules which exceed this threshold will be used to compute the current output. Thus, rules with negligible effect will not be included.

The modified actor / critic elements of FUZBOX, labelled distributed ASE (DASE) and distributed ACE (DACE) respectively, operate in the same way as the original BSA implementation when a *single* rule is chosen using winner-takes-all competition based upon the RAS (equivalent to choosing a single "box" as before). Both the DASE and DACE dynamics are similar to those of the original ASE and ACE systems but with a normalised scalar parameter (RAS) used to weight the individual rule contributions. Each rule is assigned an RAS which determines the effect of a given rule when the rules are combined to give an overall control output.

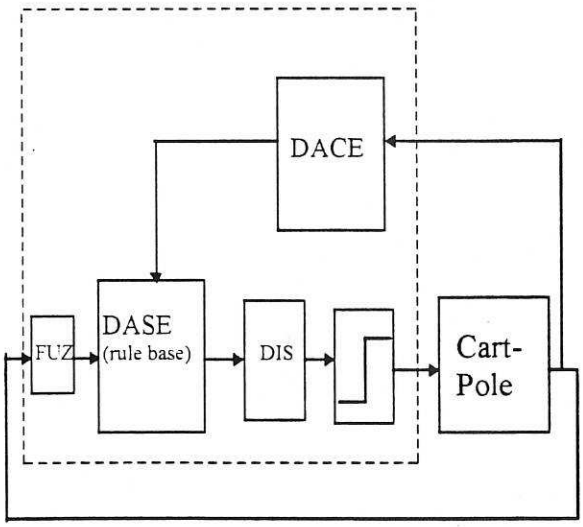

Figure 2. The FUZBOX neurocontroller. FUZ denotes the fuzzification process detailed in the text and DIS denotes the combination of rule information (distribution) to give the weighted average output. This is then used to generate the actual control output.

5. THE BSA AND FUZBOX IMPLEMENTATIONS: SOME RESULTS

Replication studies were carried out as detailed in Barto, Sutton and Anderson, [1]. The simulation conditions and parameters were similar to those of the BSA

implementation except that runs were not terminated when the trial of a particular run first reached the ceiling of 500,000 time steps because learning was still occurring in some cases and the system had to reach the ceiling value a large number of times consecutively to indicate convergence. Rules are added incrementally if the rule does not exist which would have the highest possible RAS, that is, a fuzzy region of state-space is entered which is not covered by the existing fuzzy rule-base and is highly representative of the current state.

The results indicate that the number of trials required to converge to a solution of the control problem is generally lower for FUZBOX in comparison with the BSA system given the same cart-pole and noise conditions. This is confirmed by the average convergence time of 45.9 trials for FUZBOX (over 10 runs) compared with 83.8 trials for the original BSA system. The results for twenty runs give an average convergence of 61.2 trials. Twenty runs of the original BSA system give an average convergence time of 126.45 trials.

These results for FUZBOX indicate that distribution of information across several boxes decreases the learning time required to acquire a successful control strategy for the given initial conditions. Figure 3 illustrates the performance of FUZBOX for the first 10 runs. All of these runs converged within 100 trials. The solid curve shows the average pole-balancing time over the 10 runs for each trial. A single point is plotted to indicate the average of each bin of 5 consecutive trial (ensemble) averages. The dotted curves show 1 standard deviation either side of the mean. The circles at the top of the graph indicate at which trial the members of the 10 run set converged.

Note the difference between the low and high numbers of trials required prior to convergence (29 and 70 trials respectively). This difference is a consequence of the stochastic nature of reinforcement learning where weight perturbations allow exploration of the state-space. In some cases, the reinforcement learning system will find a solution quickly if the explored region of state-space is representative of the current operating region. In other cases, the exploratory trajectories are forced away from the operating region and the reinforcement learning system has to learn to recover from these perturbations.

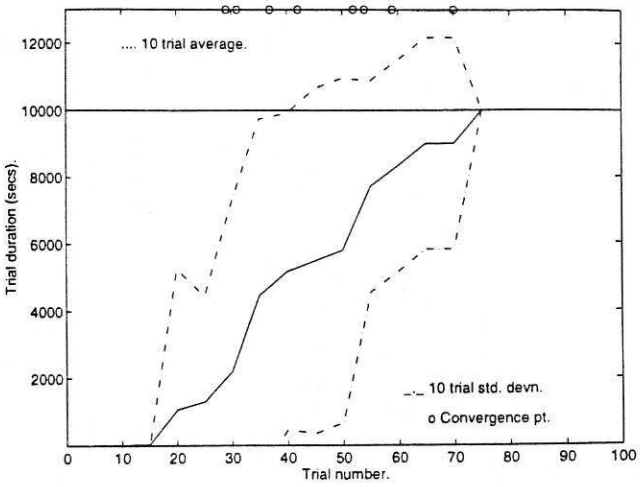

Figure 3 FUZBOX simulation results showing the first 10 runs. Note that there are two coincident convergences at 31 and 54 trials respectively.

Figure 4 illustrates the increase in number of rules (boxes) as a function of trial number. The curve appears to approach an average asymptotic value of approximately 200 rules. This means that approximately 425 rules remain unused for this particular set of cart-pole initial conditions[1].

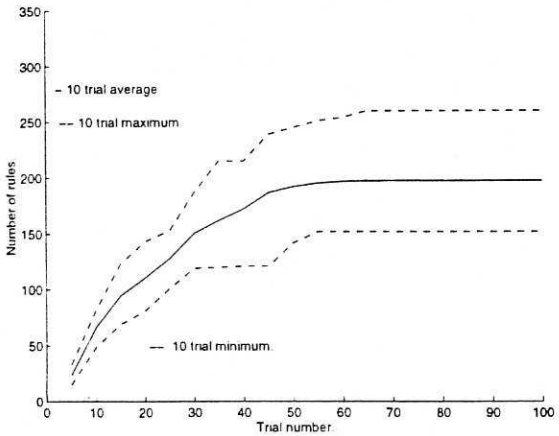

Figure 4 The average number of rules for ten runs using the FUZBOX neurocontroller. Note that about 200 rules are generated on average compared with the possible 625.

Dynamic allocation of rules prevents the allocation and use of redundant memory, thus, reducing computational overheads. It is likely that more rules will be required for more demanding initial conditions and will be allocated accordingly. Another advantage of dynamic allocation of rules is that it facilitates pruning of redundant rules. Rules may be removed from the rule-base and thereby from the storage requirements of the system. Rule redundancy and removal is discussed further in Section 7.

[1] The set of initial conditions was used for each run but the random number seed for the perturbation noise was varied.

6. CONTROL

Johnson and Smartt [15] note that the pole angle oscillates considerably in the original BSA implementation and just manages to stay within the failure limits. This is commensurate with observations made during the replication studies featured in this work.

It was observed that once FUZBOX had acquired an effective control strategy, it was able to maintain control well within the error boundaries. Although FUZBOX is still using bang-bang control, the combination of information across fuzzy regions of state-space allows a more informed choice of output. An example of FUZBOX control is presented here to illustrate the quality of control and to emphasise that the assessment of reinforcement learning must take into account more than just the learning-rate (Sammut and Crib [16])

A single run of FUZBOX was carried out using the conditions given for the 20 run set except that the pole angle was initialised to 11 degrees from the vertical for training and testing. Figure 5 shows a phase plane plot for this run. Figure 6 shows the cart-position for the first 8.5 seconds and illustrates clearly the use of predominantly right directed forces to rectify the pole. This control policy pushes the cart to around 1.2m away from the origin after which corrective action attempts to push the cart back to the origin without losing control of the pole.

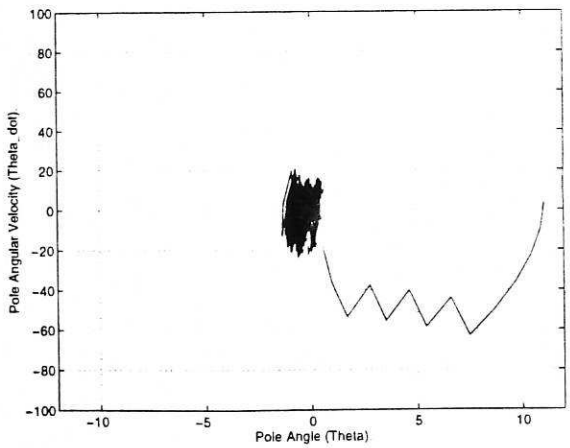

Figure 5. Phase plane plot for the 11 degree initial condition FUZBOX run. Note how the angle is brought into the stable region in the centre of the phase plane.

Figure 7 is commensurate with this and shows the transition between positive cart velocity and negative cart velocity as control emphasis switches from the pole to the cart. In other words, for the pole initial condition of 11 degrees, control forces have to be predominately right-directed giving the cart a positive velocity (and displacement). To compensate for this, the cart velocity is made negative with rapid switching to maintain the pole balance (Fig 7). The cart then moves towards the origin.

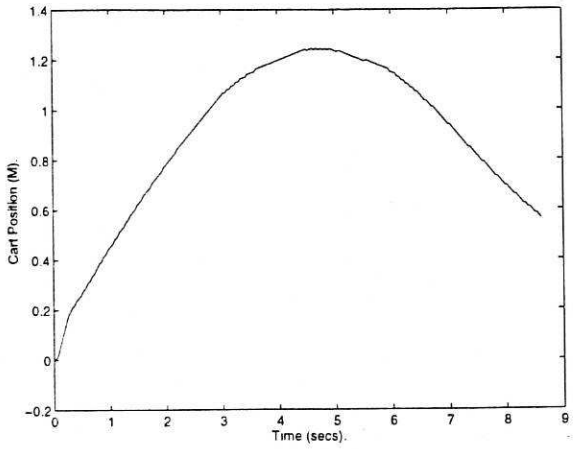

Figure 6 Cart displacement plot for the 11 degree initial condition FUZBOX run. Note the significant move away from the origin as the pole angle is corrected. The large displacement is then corrected.

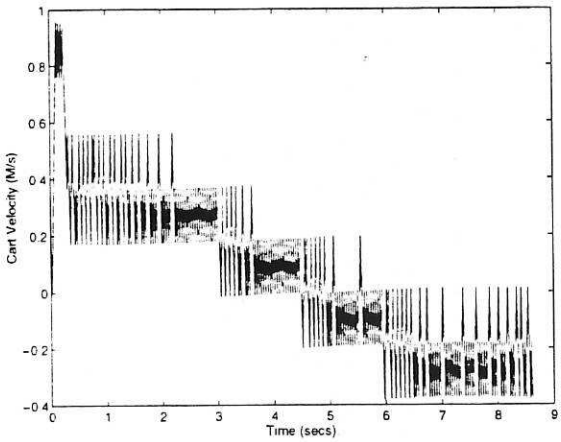

Figure 7 Cart velocity plot for the 11 degree initial condition FUZBOX run.

The pole angle evolution is shown in Figure 8 where there is an initial rapid compensation forcing the pole towards zero followed by oscillation between zero degrees and -2 degrees for about six seconds.

The pole velocity is shown in Figure 9. There is an initial negative pole velocity as expected followed by rapid oscillation of velocity around zero. The oscillatory behaviour around zero is predominantly positive as the neurocontroller compensates for the cart displacement.

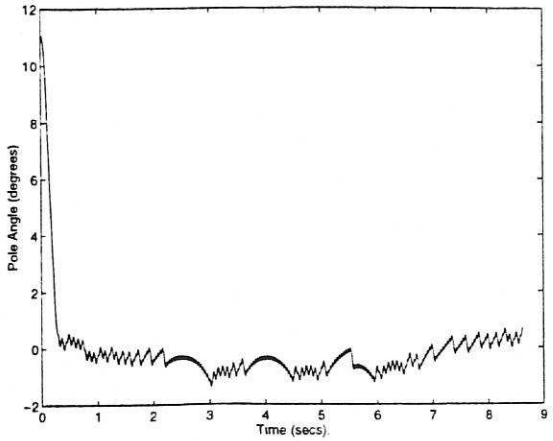

Figure 8 Pole angle plot for the 11 degree FUZBOX run.

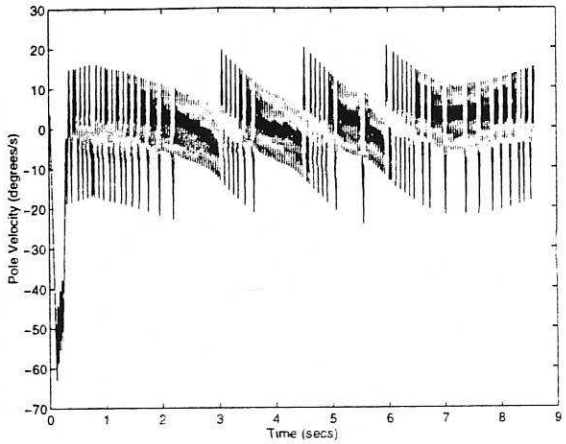

Figure 9. Pole angular velocity plot for the 11 degree initial condition FUZBOX run.

7. A TYPICAL RULE-BASE

A typical FUZBOX simulation was carried out using parameters identical to those used in the original BSA implementation. Following the simulation, the 14 most important rules—in terms of relative rule strength (RRS)—were selected out of a total of 152 generated by FUZBOX. These 14 rules accounted for 89.6 % of the total rule strength of unity. Figure 10 shows the cumulative rule strength with respect to the rule rank. Eleven new rules were generated for this run of five trials. The maximum RRS value of the newly generated rules was 0.02 or 2%. The total relative rule strength attributable to the 11 new rules was 5.3% which means that the total rule strength had increased from 89.6% to 94.7% indicating that a little information had been added to the *a priori* rule-base taken from the first run.

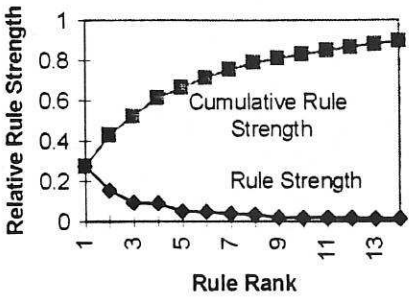

Figure 10. A plot showing the absolute and cumulative relative rule strength values for a ranked set of rules comprising a successful FUZBOX rule-base.

One of the features of self-organising autonomous systems is their adaptiveness to changing conditions during operation. To illustrate the adaptiveness of FUZBOX, the rules used in the previous simulation were negated, that is, positive outputs were made negative and vice versa. This *a priori* rule base was exactly opposite to a known successful rule base which meant that FUZBOX would not be able to balance the pole immediately.

FUZBOX required a total of 98 trials before converging to a consistent balancing time of 10000 seconds. A total of 194 rules were generated, although, only a small fraction of this total number might be required for balancing. This simulation demonstrates clearly that the self-organising nature of FUZBOX allows on-line recovery from changes in operating conditions; indeed, even changes as drastic as complete reversal of successful rules.

The rule base consisted of 14 rules selected from a total of 152. Pruning, in this case, was done by hand. It is conceivable that this may be carried out automatically. A simple method would be to remove the "weakest" nodes periodically if the relative rule strength drops below a given threshold. However, problems of preventing "rule decay" in regions of state-space no longer used, but which were once important, need to be addressed.

8. CONCLUSIONS AND FURTHER WORK

The simulations suggest that it is possible to retain Barto, Sutton and Anderson's ASE/ACE subsystems and their proven success whilst improving the performance of the overall system by extending their capabilities by fuzzifying and combining state-information. The fuzzification leads naturally to a self-organising rule-base and points to the possibility of

autonomous controllers capable of generating transparent linguistic rules.

The current drawbacks to the system are:

- the proliferation of uninformative rules caused by stochastic search of state-space during the early stages of establishment of the control mapping;
- the need for parameters which have to be set by the user;
- the lack of general rules which combine information from the specific rules involving all linguistic variables.

Possible solutions and indicators of further work include:

- the use of "relevance" pruning to remove rules created by state-space trajectories very rarely followed after control has been established—if operating conditions do change, new rules can be created dynamically and will not be pruned if significant;
- the use of self-tuning parameters to adapt the rule fuzzification during learning. This is possibly the most difficult solution and will require "meta" control at a hierarchical level above that of the ACE element to ensure intelligent tuning based upon overall performance;
- The use of rule "lumping" techniques to produce more general rules.

There is clearly much room for improvement in the current system but nevertheless it does provide an alternative approach to adaptive control. The achievement of increased neurocontroller autonomy (reduced operator intervention) is an ongoing process which will benefit from the combination of established neural network architectures in novel ways.

The authors would like to acknowledge the support of this work by the Engineering and Physical Sciences Research Council of the UK.

8. REFERENCES

[1] Barto, A. G. Sutton, R. S., and Anderson, C. W. , Neuron-like adaptive elements that can solve difficult learning control problems, *IEEE Trans. Syst. Man. Cybern,*. Vol. SMC-13, 1983, 834-846.

[2] Marriott, S. *The Application of Adaptive Resonance Theory and Reinforcement Learning to Mapping and Control,* Ph.D. Thesis, the University of Sheffield, 1996.

[3] Michie, D. and Chambers, R. A., BOXES: an Experiment in Adaptive Control, in *Machine Intelligence 2,* E. Dale and Michie, D. Eds. Edinburgh: Oliver and Boyd, 1968.

[4] Sutton, R. S., Learning to Predict by the Methods of Temporal differences, *Machine Learning,* **3**, 1988, 9-44.

[5] Sutton, R. S., Barto, A. G. and Williams, R. J., Reinforcement Learning is Direct Adaptive Optimal control, *IEEE Control Systems* Magazine, April, 1992, 19-22.

[6] Sutton, R. S., and Barto, A. G., Towards a Modern Theory of Adaptive Networks: Expectation and Prediction, *Psychological Review,* **88** (2), 1981, 135-170.

[7] Barto, A. G. and Sutton, R. S., Simulation of Anticipatory Responses in Classical Conditioning by a Neuron-Like Adaptive Element, *Behavioral Brain Research,* **4**, 1982, 221-235.

[8] Marriott, S. and Harrison, R. F., A Self-Organising State Space Decoder for Reinforcement Learning, *Research Report No 582,* The University of Sheffield, U.K., 1995.

[9] Marriott, S. and Harrison, R. F. (1996), A Self-Organising Adaptive Neurocontroller Using Reinforcement Learning, *Proc. Control'96 Exeter U.K,* 1113-1117.

[10] Marriott, S and Harrison, R. F. Can Machines Ever Learn from their Own Mistakes?, *Journal of Measurement and Control,* (In Press)

[11] Albus, J. S., A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC), *Trans. ASME. Jnl. Dyn. Sys. Meas. and Control,* **63,** (3), 1975, 220-227

[12] Albus, J. S. , Data Storage in the Cerebellar Model Articulation Controller (CMAC), *Trans. ASME. Jnl. Dyn. Sys. Meas. and Control.* **63** (3), 1975, 228 -233

[13] Lin, C-S. and Kim, H., CMAC-based Adaptive Critic Self-Learning Control, *IEEE Transactions on Neural Networks,*. **2**, 5, 1991, 530-533.

[14] Sugeno, M. and Nishida, M. , Fuzzy Control of Model Car, *Fuzzy Sets and Systems,* **16,** 1985, 103-113

[15] Johnson, J. A., and Smartt, H. B., Fuzzy Logic and the Associative Search Element *Proc. World congress on Neural Networks,* Portland, Oregon Vol. II, 1993, 52-55

[16] Sammut, C. and Cribb, J., Is Learning Rate a Good Performance Criterion for Learning?, *Proceedings of the Seventh International Workshop On Machine Learning,* Morgan Kaufmann, 1990, 170-178