9

Fuzzy and Neural Control

Hamid R. Berenji
Sterling Software
Artificial Intelligence Research Branch
NASA Ames Research Center
Mountain View, CA 94035

Abstract

Fuzzy logic and neural networks provide new methods for designing control systems. Fuzzy logic controllers do not require a complete analytical model of a dynamic system and can provide knowledge-based heuristic controllers for ill-defined and complex systems. Neural networks can be used for learning control. In this chapter, we discuss hybrid methods using fuzzy logic and neural networks which can start with an approximate control knowledge base and refine it through reinforcement learning.

1. INTRODUCTION AND MOTIVATION

What is the fundamental difference between Fuzzy Logic Controllers (FLCs) and those that are based on conventional control theory? How can FLCs learn and adaptively change their performance? These questions are among the main questions that I will discuss in some details in this chapter. However, to briefly answer the first question, FLCs do not require a complete analytical model of a dynamic system and can provide knowledge-based heuristic controllers for ill-defined and complex systems. As for the second question above, in this chapter, we consider neural networks to provide learning capability for FLCs although other learning methods of artificial intelligence may also be used.

This chapter is not intended to provide a complete survey on either FLCs or applications of neural networks in control, since other appropriate surveys on these topics are already available (e.g., see Berenji [8], Sugeno [29], Barto [5], and Antsaklis [2]). However, in this chapter, we will first cover some basics of fuzzy set theory and their application in designing FLCs. Next we discuss some issues related to the stabil-

ity analysis of FLCs and some applications of this theory. Then we briefly describe the application of neural nets in control with a view toward a special family of techniques known as reinforcement learning. We then discuss how FLCs can learn from experience through reinforcement learning. We conclude the chapter by listing a number of research directions for this field.

2. FUZZY LOGIC FOR INTERPOLATIVE REASONING

In his seminal work, Zadeh [42] devised the fuzzy set theory as an extension of the set theory. Non-fuzzy sets only allow full membership or no membership at all, where fuzzy sets allow partial membership. In other words, an element may partially belong to a set. This *partial memberships* can take values ranging from 0 to 1. Here, we review some basic concepts of fuzzy sets; however, see [17, 8] for more complete discussion.

Assuming that A and B are two fuzzy sets with membership functions μ_A and μ_B respectively, then the *complement* of fuzzy set A is a fuzzy set $\bar{A}$ with membership function

$$\mu_{\bar{A}} = 1 - \mu_A(x).$$

Traditionally, in fuzzy logic, the *union* and the *intersection* of sets A and B are defined using *Max* and *Min* operators:

$$\mu_{A \cup B} = \max\{\mu_A, \mu_B\}.$$

$$\mu_{A \cap B} = \min\{\mu_A, \mu_B\}.$$

However, the generalized family of these operators, known as *triangular norms* and *triangular co-norms* have also been extensively studied in the past. Berenji, et al. [9] have studied a different generalized family of operators known as *Ordered Weighted Averaging (OWA)* operators and have applied it to control . The OWA operators introduced by Yager [40] for multi-criteria decision making provide a facility to implement various types of aggregation operators commonly used in fuzzy control. The OWA operators generalize the ordinary *and* and *or* functions used in rule-based control [40].

3. BASIC ARCHITECTURE OF FLC

In the design of a fuzzy controller, one must identify the main control variables and determine a term set that is at the right level of granularity for describing the values of each variable. For example, a term

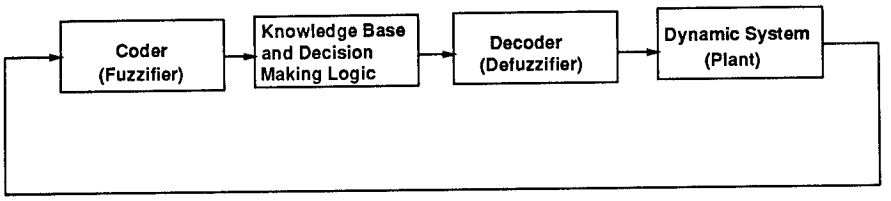


Figure 1: A simple architecture of a fuzzy logic controller.

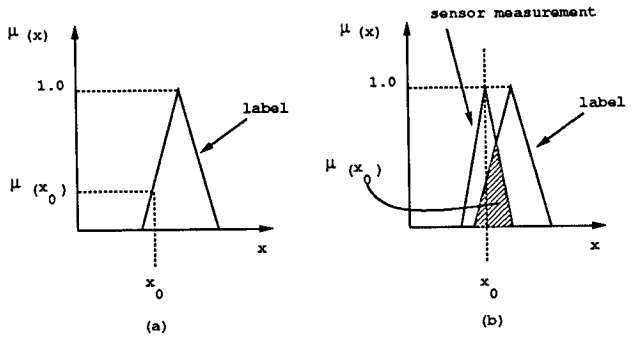


Figure 2: Matching a sensor reading x_0 with the membership function $\mu(x)$ to get $\mu(x_0)$; (a) crisp sensor reading (b) fuzzy sensor reading.

set including linguistic values such as { *Small, Medium, Large*} may be satisfactory in some domains; whereas other domains may instead require the use of a five term set such as {*Very Small, Small, Medium, Large, and Very Large*}.

After the linguistic term sets for the main control variables are determined, a knowledge base is developed using these control variables and the values that they may take. If the knowledge base is a rule base, more than one rule may fire simultaneously; hence it requires a *conflict resolution* method for decision making, as will be described later.

Figure 1 illustrates a simple architecture for a fuzzy logic controller. This architecture consists of four modules whose functions are described next.

3.1 Encoder

In coding the values from the sensors, one transforms the values of the sensor measurements in terms of the linguistic labels used in the preconditions of the rules. If the sensor reading has a crisp value, then the fuzzification stage requires matching the sensor measurement

against the membership function of the linguistic label as shown in Figure 2(a). If the sensor reading contains noise, it may be modeled by using a triangular membership function where the vertex of the triangle refers to the mean value of the data set of sensor measurements and the base refers to a function of the standard deviation (e.g., twice the standard deviation, as used in [39]). Then in this case, fuzzification refers to finding the intersection of the label's membership function and the distribution for the sensed data as shown in Figure 2(b).

3.2 Knowledge Base

There are two main tasks in designing the control knowledge base. First, a set of linguistic variables must be selected which describe the values of the main control variables of the process. Both the main input variables and the main output variables must be linguistically defined in this stage using proper term sets. The selection of the level of granularity of a term set for an input variable or an output variable plays an important role in the smoothness of control. Secondly, a control knowledge base must be developed which uses the above linguistic description of the main variables. Sugeno [29] has suggested four methods for doing this: Expert's Experience and Knowledge, Modeling the Operator's Control Actions, Modeling a process, and Self Organization.

Among these methods, the first method is the most widely used [21]. In modeling the human expert operator's knowledge, fuzzy control rules of the form:

IF Error is small and Change-in-error is small, Then force is small.

have been used in studies such as [30]. This method is effective when expert human operators can express the heuristics or the knowledge that they use in controlling a process in terms of rules of the above form. Applications have been developed in process control (e.g., cement kiln operations [15]). Beside the ordinary fuzzy control rules which have been used by Mamdani and others, where the conclusion of a rule is another fuzzy variable, a rule can be developed whereby its conclusion is a function of the input variables. For example, the following implication can be written:

$$\text{IF } X \text{ is } A_1 \text{ and } Y \text{ is } B_1, \text{ Then Z} = f_1(X, Y)$$

where the output Z is a function of the values that X and Y may take.

The second method, directly models the control actions of the human operator. Takagi and Sugeno [35] and Sugeno and Murakami [30] have used this method for modeling the control actions of a driver in parking a car.

The third method deals with fuzzy modeling of a process where an approximate model of the plant is configured by using implications that describe the possible states of the system. In this method, a model is developed and a fuzzy controller is constructed to control the fuzzy model, making this approach similar to the traditional approach taken in control theory. Hence, structure identification and parameter identification processes are needed. For example, a rule discussed by Sugeno [29] is of the form:

$$\text{If } x_1 \text{ is } A_1^i,\ldots,x_m \text{ is } A_m^i \text{ Then } y = p_0^i + p_1^i x_1 + \ldots + p_m^i x_m,$$

for $i = 1,\ldots,n$ where n is the number of such implications and the consequence is a linear function of the m input variables.

Finally, the fourth method refers to the research of Mamdani and his students in developing self-organizing controllers [26]. The main idea in this method is the development of rules which can be adjusted over time to improve the controllers' performance.

3.3 Decision Making Logic

As mentioned earlier, because of the partial matching attribute of fuzzy control rules and the fact that the preconditions of rules do overlap, usually more than one fuzzy control rule can fire at a time. The methodology which is used in deciding what control action should be taken as the result of the firing of several rules can be referred to as *conflict resolution.* The following example, using two rules, illustrates this process. Assume that we have the following rules:

Rule 1: IF X is A_1 and Y is B_1 THEN Z is C_1
Rule 2: IF X is A_2 and Y is B_2 THEN Z is C_2

Now, if we have x_0 and y_0 as the sensor readings for fuzzy variables X and Y, then for Rule 1, their *truth values* are represented by $\mu_{A_1}(x_0)$ and $\mu_{B_1}(y_0)$, where μ_{A_1} and μ_{B_1} represent the membership function for A_1 and B_1, respectively. Similarly for Rule 2, we have $\mu_{A_2}(x_0)$ and $\mu_{B_2}(y_0)$ as the truth values of the preconditions. Assuming that a *minimum* operator is used as the conjunction operator, the *strength* of Rule 1 can be calculated by:

$$w(1) = min(\mu_{A_1}(x_0), \mu_{B_1}(y_0)).$$

Similarly for Rule 2:

$$w(2) = min(\mu_{A_2}(x_0), \mu_{B_2}(y_0)).$$

The control output of Rule 1 is calculated by applying the matching strength of its preconditions to its conclusion[1]:

$$z(1) = \mu_{c_1}^{-1}(w(1)),$$

and for Rule 2:

$$z(2) = \mu_{c_2}^{-1}(w(2)).$$

This means that as a result of reading sensor values x_0 and y_0, Rule 1 is recommending control action $z(1)$ and Rule 2 is recommending control action $z(2)$.

3.4 Decoder

Also known as *Defuzzifier*, the decoder produces a nonfuzzy control action that best represents the membership function of an inferred fuzzy control action as a result of combining several rules. Several defuzzification methods such as center of area (COA) and mean of maxima (MOM) have been suggested. The COA method calculates the center of the area resulted from superimposing the conclusions of the firing rules, and the MOM method averages out the values for which the membership of the combined membership function reaches the maximum. These methods are reviewed in [8]. In the example discussed above and shown in Figure 3, the combination of the rules produces a nonfuzzy control action z^* which is calculated using Tsukomoto's defuzzification method:

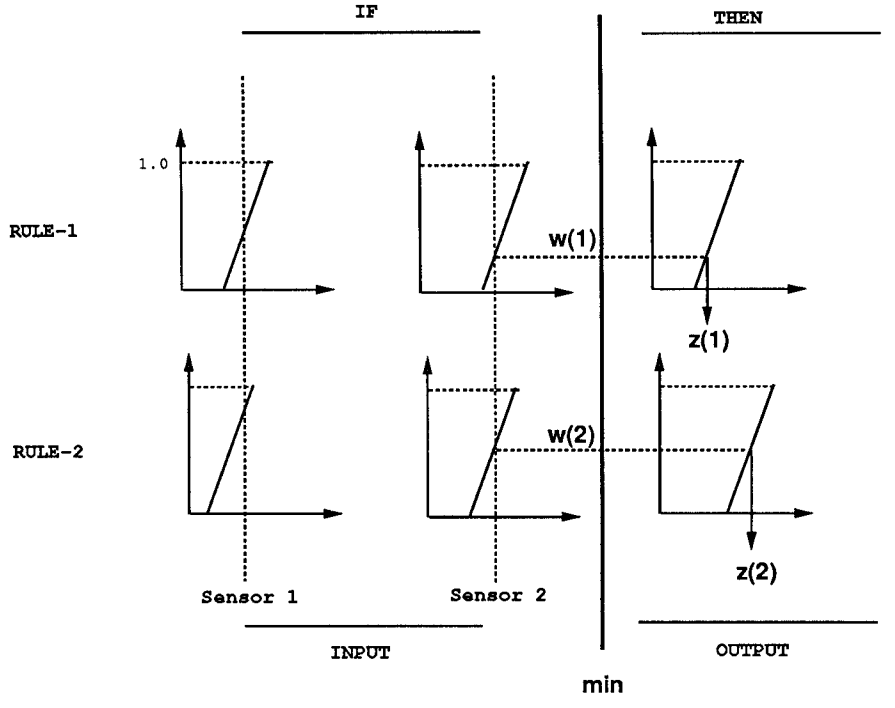
$$z^* = \frac{\sum_{i=1}^{n} w(i)z(i)}{\sum_{i=1}^{n} w(i)}$$

where n is the number of rules with firing strength, $w(i)$, greater than 0 ($n = 2$ in the above example) and $z(i)$ is the amount of control action recommended by rule i.

3.5 Hierarchical Fuzzy Control

Berenji, et al. [11] have proposed the following algorithm for the design of fuzzy controllers with multiple goals.

1. Here, it should be noted that the inverse functions can only be defined for monotonic membership functions. Since most fuzzy membership functions are defined using non-monotonic functions, other mapping functions have been used in the literature, which are reviewed in [8]. For simplicity, we explain the mapping and defuzzification processes using monotonic functions only, although other approaches (also reviewed in [8]) are equally applicable.



$$z^* = \frac{z(1)\,w(1) + z(2)\,w(2)}{w(1) + w(2)}$$

Figure 3: Defuzzification of the combined conclusion of rules using Tsukamoto's monotonic membership functions.

1. Let $G = \{g_1, g_2, ...g_n\}$ be the set of goals to be achieved and maintained. Notice that for $n = 1$ (i.e., no interacting goals), the problem becomes simpler and may be handled using the earlier methods in fuzzy control (e.g., see [21]).

2. Let $G' = p(G)$ where p is a function that assigns priorities among the goals and G' is the priorotized list of goals in G. We assume that such a function can be obtained in the given domain. In many control problems, it is possible to specifically assign priorities to the goals. For example, in the simple problem of balancing a pole on the palm of a hand and also moving the pole to a pre-determined location, it is possible to do this by first keeping the pole as vertical as possible and then gradually moving to the desired location. Although these goals are highly interactive (i.e., as soon as we notice that the pole is falling, we may temporarily set aside the other goal of moving to the desired location), we still can assign priorities fairly well.

3. Let $U = \{u_1, u_2, ..., u_n\}$ where u_i is the set of input control variables related to achieving g_i.

4. Let $A = \{a_1, a_2, ..., a_n\}$ where a_i is the set of linguistic values used to describe the values of the input control variables in u_i.

5. Let $C = \{c_1, c_2, ..., c_n\}$ where c_i is the set of linguistic values used to describe the values of output control variables.

6. Acquire the rule set R_1 of approximate control rules directly related to the highest priority goal. These rules are in the general form of

$$\text{IF } u_1 \text{ is } a_1 \text{ THEN } Z \text{ is } c_1.$$

7. For $i = 2$ to n, form the rule sets R_i. The format of the rules in these rule sets is similar to the ones in the previous step except that they include aspects of *approximately achieving the previous goal*:

$$\text{IF } g'_{i-1} \textit{ is approximately achieved} \text{ and } u_i \text{ is } a_i \text{ THEN } Z \text{ is } c_i.$$

The approximate achievement of a goal in step 7 of the above algorithm refers to holding the goal parameters within smaller boundaries. The interactions among the goal g'_i and goal g'_{i-1} are handled by forming rules which include more preconditions in the left hand side. For example, let us assume that we have acquired a set of rules R_1 for keeping a pole vertical. In writing the second rule set R_2 for moving to a pre-specified location, aspects of approximately achieving g'_1 should be combined with control parameters for achieving g'_2. For example,

a precondition such as *the pole is almost balanced* can be added while writing the rules for moving to a specific location. A fuzzy set operation known as *concentration* [43] can be used here to systematically obtain a more focused membership functions for the parameters which represent the achievement of previous goals. The above algorithm has been applied in cart-pole balancing and more details can be found in [11].

3.6 Stability Analysis

Stability analysis of fuzzy logic controllers is an important issue for which only a limited number of studies have been performed. For example, Braae and Rutherford [13] used control rules as transition functions between the states of the system in terms of a generalized state space. Kiszka, et al. [16] have provided an *energistic* approach to the analysis of stability and robustness of FLCs, where an energy function can be found which consistently decreases along a solution trajectory. This approach is similar to Chen's approach [14] in using the concept of cell to cell mapping, but Chen uses a Lyapunov based approach. Finally, Langari [18] provides a stability analysis for FLCs under the assumption that plant structure with unknown but bounded parameters is available; however, some assumptions are placed on plant dynamics. For further analysis of stability in FLC, refer to Langari and Berenji [19].

3.7 Applications of Fuzzy Logic Controllers

Mamdani and Assilian [21] were the first to apply fuzzy set theory to control problems (e.g., the control of a laboratory steam engine). This experiment triggered some other applications, and in recent years there has been a very significant increase in the number of applications of fuzzy logic control. Currently, there are numerous products on the market which use fuzzy logic control (mostly designed in Japan); Berenji [8] reviews a number of these applications. In the following, we discuss a few of these systems in more detail.

3.7.1 *Automatic train control*

Yasunobu and Miyamoto at Hitachi, Ltd. [41] have designed a fuzzy controller for the Automatic Train Operation (ATO) system which has been in use in the city of Sendai, Japan since July 1987. The two main operations of the system are Constant Speed Control (CSC) and Train Automatic Stop Control (TASC). The CSC operation results in maintaining a constant target speed (specified by the operator at

the start of the train operation) during the train travel. The TASC operation controls the speed of the train in order to stop the train at the prespecified location. The system uses only a few rules (i.e., 12 rules for each of the CSC and the TASC operations), and the control is evaluated every 100 milliseconds. These operations use the evaluation of safety, riding comfort, traceability of target velocity, accuracy of stop gap, running time, and energy consumption criteria in deciding a control strategy. The control rules are of the predictive fuzzy form:

$$\text{If } (\ u \text{ is } C_i \rightarrow x \text{ is } A_i \text{ and } y \text{ is } B_i) \text{ Then } u \text{ is } C_i.$$

For example, when the train is in the TASC zone, the following rule is used:

If the control notch is not changed and
the train will stop at the predetermined location, then
the control notch is not changed.

The system performs as skillfully as human experts do and superior to an ordinary PID[2] automatic train operation controller in terms of stopping precision, energy consumption, riding comfort, and running time.

3.7.2 *Sugeno's model helicopter*

Sugeno has initiated several projects on applying fuzzy logic to the control of a model helicopter. Among these are radio control by oral instructions, automatic autorotation entry in engine failure cases, and unmanned helicopter control for sea rescue [28]. Although these projects have just started, several interesting results have already been achieved. The input variables from the helicopter include pitch, roll, and yaw, and their first and second derivatives. The control rules written for the helicopter regulate the up/down, forward/backward, left/right, and nose direction. For example, the longitudinal stick controls pitch and, therefore, forward/backward movement of the rotorcraft.

An example of a fuzzy control rule for hovering is the following:

If the body rolls, then
control the lateral in reverse.

Or as another example for hovering control:

If the body pitches, then
control the longitude in reverse.

2. Proportional, Integral, and Derivative.

The helicopter is inherently unstable and the helicopter control problems under study in the above projects are challenging control problems. These studies have already produced results which illustrate the strength of the fuzzy logic control technology.

4. NEURAL NETWORKS FOR LEARNING CONTROL

Connectionist learning approaches [5] can be used in learning control. Here, we can distinguish three classes: *supervised learning*, *reinforcement learning*, and *unsupervised learning*. In supervised learning, at each time step, a teacher provides the desired control objective to the learning system. In reinforcement learning, the teacher's response is not as direct and informative as in supervised learning and it serves more to evaluate the state of the system. In unsupervised learning, the presence of a teacher or a supervisor to provide the correct control response is not assumed.

If supervised learning can be used in control (e.g., when the input-output training data is available), it has been shown that it is more efficient than reinforcement learning (e.g., faster learning [6, 1]). However, many control problems require selecting control actions whose consequences emerge over uncertain periods for which input-output training data are not readily available. In such domains, reinforcement learning techniques are more appropriate than supervised learning.

4.1 Reinforcement Learning in Control

As mentioned earlier, in reinforcement learning, one assumes that there is no supervisor to critically judge the chosen control action at each time step. The learning system is told indirectly about the effect of its chosen control action. The study of reinforcement learning is related to the *credit assignment* problem where, given the performance (results) of a process, one has to distribute reward or blame to the individual elements contributing to that performance. In rule-based systems, for example, this means assigning credit or blame to individual rules engaged in the problem solving process. Samuel's checker-playing program is probably the earliest AI program which used this idea [27]. Michie and Chambers [23] used a reward-punishment strategy in their BOXES system, which learned to do cart-pole balancing by discretizing the state space into *non-overlapping* regions (boxes) and applying two opposite constant forces. Barto, Sutton, and Anderson [4] used two neuron-like elements to solve the learning problem in cart-pole balancing. In these approaches, the state space is partitioned into non-overlapping regions and then the credit assignment is performed

on a local basis.

Reinforcement learning has its roots in studies of animal learning and psychological research on human behavior (e.g., [3]). It directly relates to the theory of *learning automata* originated by the work of Tsetlin [36] and further developed by the work of Narendra and Thathachar [25], Narendra and Lakshmivarahan [24], and Mendel and McLaren [22] in control engineering. Since reinforcement learning techniques can be used without an explicit teacher or supervisor, they construct an internal evaluator, or a *critic*, capable of evaluating the dynamic system's performance. How to construct this critic so that it can properly evaluate the performance in a way that is useful to the control objective, is itself a significant problem in reinforcement learning. Given the evaluation by the critic, the other problem in reinforcement learning is how to adjust the control signal. Barto [5] discusses several approaches to this problem based on the gradient of the critic's evaluation as a function of control signals.

4.1.1 *Temporal Difference methods*

Related to reinforcement learning are the Temporal Difference (TD) methods, a class of incremental learning procedures specialized for prediction problems, which were introduced by Sutton [32]. The main characteristic of these methods is that they learn from successive predictions; whereas, in the case of supervised learning methods, learning occurs when the difference between the predicted outcome and the actual outcome is revealed (i.e., the learning model in TD does not have to wait until the actual outcome is known and can update its parameters *within* a trial period). The difference between the TD methods and the supervised learning methods becomes clear when these methods are distinguished as single-step versus multi-step prediction problems. In the single-step prediction (e.g., Widrow-Hoff rule [38]), complete information regarding the correctness of a prediction is revealed at once. Whereas in multi-step predicvtion, this information is not revealed until more than one step after the prediction is made; however, partial information becomes available at each step. Barto, et al. [7] have recently shown a stronger relation between a specific class of these methods, called *TD algorithm*, and dynamic programming.

4.1.2 *Q-Learning*

One of the most promising techniques in reinforcement learning relates to *Q-Learning* as developed by Watkins [37]. In Q-Learning, a real-valued function, Q, of states and actions is estimated. For example, $Q(x, a)$ represents the expected discounted sum of future reward for

performing action a in state x and performing optimally thereafter. The Q-Learning algorithm keeps an estimate of Q, updating this estimate at each time step by observing the reinforcement received at that time step, applying the selected action, and observing the state of the system at the next time step. Q-Learning shows a strong relationship between reinforcement learning and on-line dynamic programming [33].

5. HYBRID NEURAL AND FUZZY CONTROLLERS

Neural and fuzzy controllers are similar in that they both allow *interpolation.* For example, once a neural network has been trained for a set of data, it can interpolate and produce answers for the cases not present in the data set.

The main idea in integrating the fuzzy logic controllers with neural networks is to use the strength of each one collectively in the resulting neuro-fuzzy control system. This fusion allows:

1. A human understandable expression of the knowledge used in control in terms of the fuzzy control rules. This reduces the difficulties in describing the trained neural network which is usually treated as a black box.

2. The fuzzy controller learns to adjust its performance automatically using a neural network structure and hence learns by accumulating experience.

The emphasis of research on hybrid neural and fuzzy controllers has been on automatic design and refinement of the membership functions (e.g., see [20, 34]). In this section, we discuss how reinforcement learning techniques can be used in refining fuzzy membership functions.

5.1 Approximate Reasoning-based Intelligent Control (ARIC)

The ARIC architecture, introduced by Berenji [10], provides adaptive learning capability for fuzzy logic controllers. Figure 4 illustrates the ARIC architecture which contains three main elements:

- The Action-state Evaluation Network (AEN), which acts as a critic and provides advice to the main controller.

- The Action Selection Network (ASN) which includes a fuzzy logic controller.

- The Stochastic Action Modifier (SAM), which changes the action

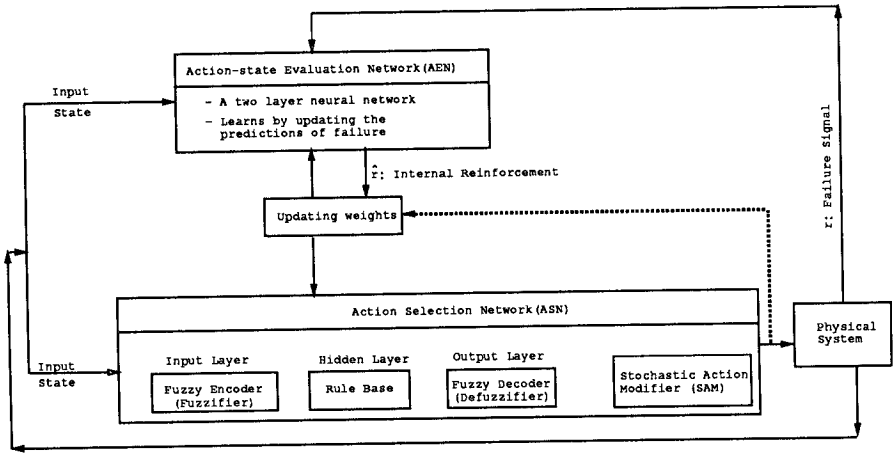


Figure 4: The ARIC Architecture.

recommended by the ASN based on internal reinforcement. The modified action is then sent to the plant.

The AEN is based on Anderson's [1] extension to Sutton's [31] AHC algorithm in which the single-layer neural network in AHC was extended to a multi-layer neural network. The ASN is a multi-layer neural network representation of a fuzzy logic controller, with as many hidden units as there are rules in the control knowledge base. The inputs to a hidden unit are the preconditions of a rule and its output is the conclusion of a rule. The ASN learns search heuristics as a probabilistic mapping from states to actions. The SAM stochastically modifies the action selected by the fuzzy controller. This change in the recommended action by ASN is more significant for a state if that state does not receive high internal reinforcements (i.e., probability of failure is high). On the other hand, if a state receives high reinforcements, SAM changes the action selected by the fuzzy controller by smaller magnitudes. This means that if the fuzzy logic controller embedded in ASN is performing well (e.g., after it has learned to control the system), then its recommendation is followed with no or only minor changes to it. However, when a state receives weak internal reinforcement (e.g., at the beginning of the learning process), SAM modifies the action recommended by the fuzzy controller more significantly. The details of this process are discussed in [10]; however, it should be noted that the learning element of ARIC's architecture is similar to learning skills in humans who start with a collection of general rules (e.g., fuzzy control rules in ARIC) and refine them through practice (e.g., reinforcement learning through a number of trials).

In summary, the ARIC's algorithm proceeds as follows.

1. Given an input, the ASN
   - determines an action using fuzzy rules
   - determines a measure of confidence in the fuzzy system's conclusion.
2. These two are appropriately used to produce the final action, which is sent to the physical system. As a result of this action, the system moves to a new state.
3. The AEN evaluates the new state, and a comparison of this evaluation with the score of the previous state gives a measure of internal reinforcement.
4. This reinforcement controls the modification of weights in both AEN and ASN, which leads to learning.
5. Over time, the AEN improves and becomes a good state evaluator, and reinforcement estimates become more reliable. Also, the ASN improves so that the recommended control actions of the fuzzy system become more correct with higher measures of confidence.

The AEN and ASN do not necessarily have the same number of hidden or output units. For example, as shown in Figure 5, in cart-pole balancing experiment, 5 and 13 units are used in the hidden layers of the AEN and ASN, respectively.

5.2 GARIC: Generalized ARIC

Berenji and Khedkar [12] have extended ARIC in many respects including the following:

- Learning is achieved by full integration of fuzzy inference into a feedforward network, which can then adaptively improve performance by using gradient descent methods.
- The fuzzy memberships used in the definition of the labels are modified (tuned) globally in all the rules, rather than being locally modified in each individual rule.
- GARIC can compensate for inappropriate definitions of fuzzy membership functions in the antecedent of control rules. GARIC is the first architecture to do this.
- GARIC introduces a new localized mean of maximum (LMOM) method for combining the conclusions of several firing control rules. This approach basically defuzzifies each rule individually and then combines the resulting defuzzified values from all the firing rules.

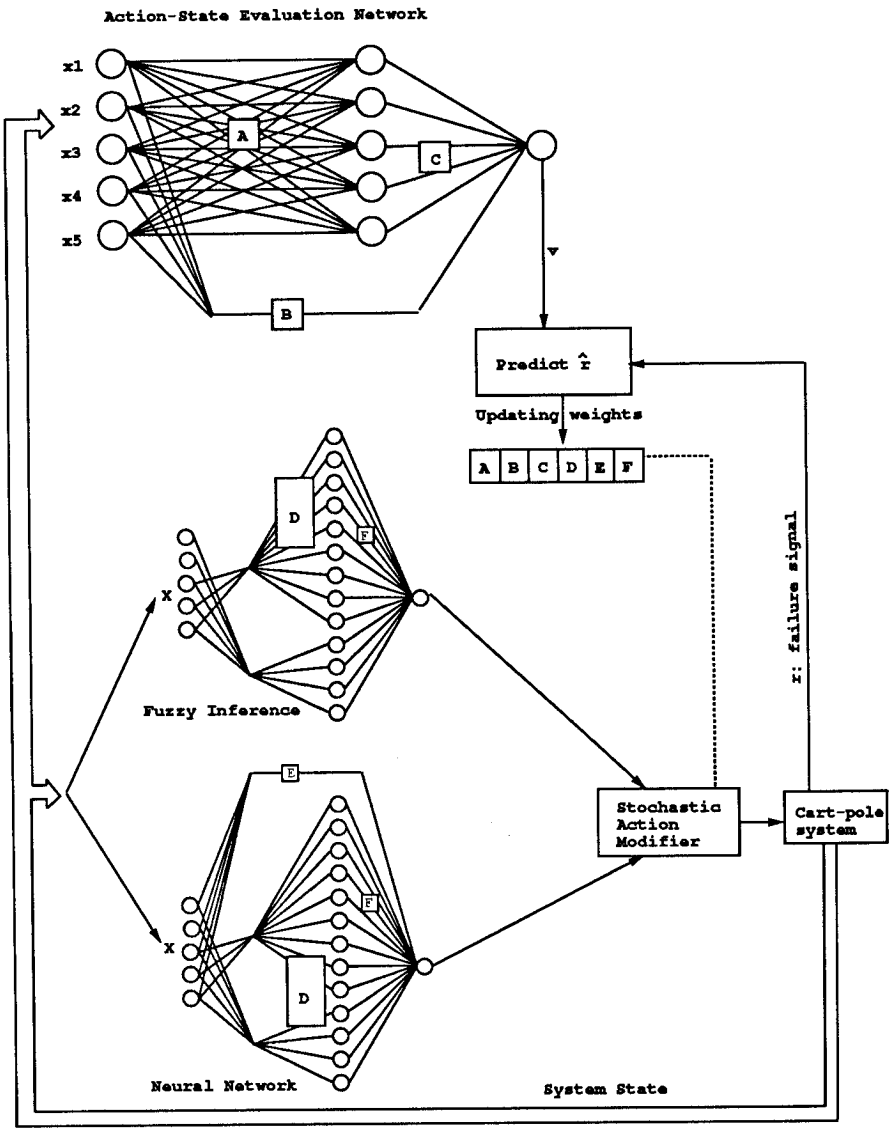


Figure 5: ARIC applied to cart pole balancing.

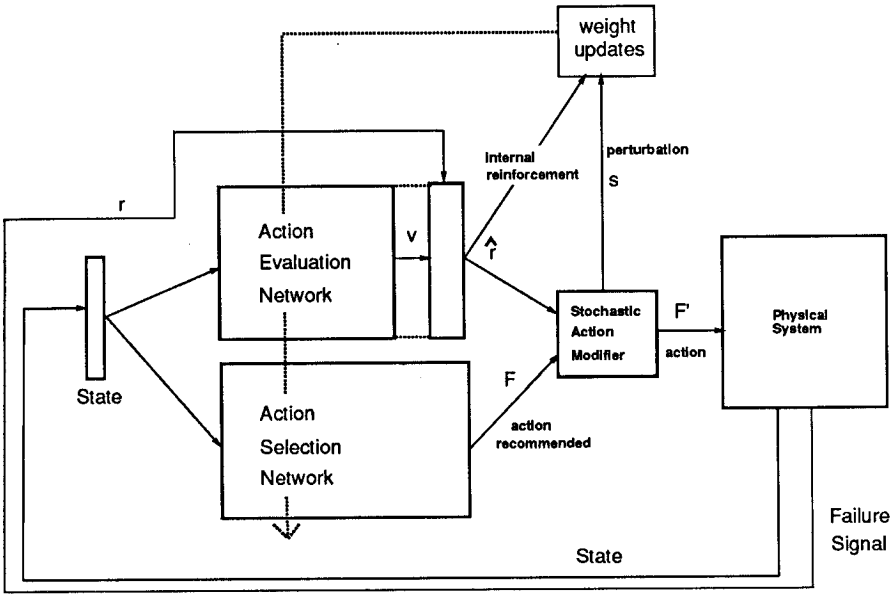


Figure 6: The Architecture of GARIC.

- Only monotonic membership functions are used in ARIC; whereas, GARIC allows any type of differentiable membership functions to be used in constructing fuzzy logic controllers.

The architecture of GARIC, which is schematically shown in Figure 6, has three components:

- The Action Selection Network maps a state vector into a recommended action F, using fuzzy inference.
- The Action Evaluation Network maps a state vector and a failure signal into a scalar score which indicates state goodness. This is also used to produce internal reinforcement $\hat{r}$.
- The Stochastic Action Modifier uses both F and $\hat{r}$ to produce an action F' which is applied to the plant.

The ensuing state is fed back to the controller along with a boolean failure signal. Learning occurs by fine-tuning the free parameters in the two networks : in the AEN, the weights are adjusted and in the ASN, the parameters describing the fuzzy membership functions are changed.

Figure 7 presents the GARIC architecture as it is applied to cart-pole balancing. The AEN network has 4 input units, a bias input unit, 5 hidden units, and an output unit.

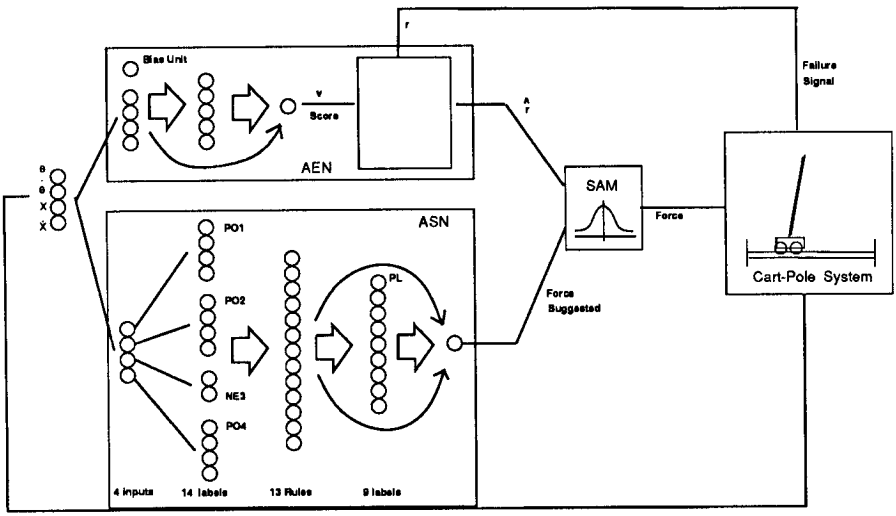


Figure 7: GARIC applied to cart pole balancing.

The GARIC architecture proposes a new way of designing and tuning a fuzzy logic controller. The knowledge used by an experienced operator in controlling a process can now be modeled using approximate linguistic terms and later refined through the process of learning from experience. GARIC provides a well-balanced method for combining the qualitative knowledge of human experts (in terms of symbolic rules) and the learning strength of the artificial neural networks. The architecture is general enough for use in other rule-based systems (besides controllers) which perform fuzzy logic inference.

6. CONCLUSION

Fuzzy logic controllers can use much of the heuristic knowledge of experienced human operators in controlling a process. In this chapter, we discussed how these controllers can provide alternative methods for the design of controllers for complex systems while requiring no analytical model of the process. We showed how neural networks can be effectively used to provide learning capabilities for these controllers. We discussed two such hybrid methods which use reinforcement learning. Architectures such as ARIC and GARIC, as discussed in this chapter, provide hybrid methods for combining the knowledge representation strength of fuzzy inference systems with the adaptive learning capability of neural networks.

For fuzzy logic controllers, structure identification issues, such as the

number of rules in the rule base, have to be resolved. Unsupervised learning and clustering methods are definitely useful here. FLCs can also use neural network methods in which a network is grown (i.e., new nodes are added to the network) based on the system's learning behavior. Parameter identification issues, such as the shape of membership functions, can be resolved using methods such as ARIC and GARIC. On the other hand, stability analysis of fuzzy logic controllers requires more general treatment than the limited classes which have already been studied. For neural networks, fuzzy logic provides a unique knowledge representation power for both including the prior knowledge in the network and explaining the results of learning. Here, issues such as the speed of learning, convergence, and stability of the neural networks for control need to be resolved.

References

[1] C. W. Anderson. *Learning and Problem Solving with Multilayer Connectionist Systems.* PhD thesis, University of Massachusetts, 1986.

[2] P.J. Antsaklis. Neural networks in control systems. *IEEE Control Systems Magazine*, April 1990.

[3] R.C. Atkinson, G.H. Bower, and E.J. Crothers. *An introduction to mathematical learning theory.* Weily, New York, 1965.

[4] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:834–846, 1983.

[5] A.G. Barto. Connectionist learning for control: An overview. COINS tech. Report 89-89, University of Massachusetts at Amherst, 1989.

[6] A.G. Barto and M.I. Jordan. Gradient following without backpropagation in layered networks. In *IEEE First Annual Conference on Neural Networks*, pages II629–II636, San Diego, CA, 1987.

[7] A.G. Barto, R.S. Sutton, and C.J.C.H. Watkins. Sequential Decision Problems and Neural Networks. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 686–693. Morgan Kaufmann, San Mateo, CA, 1990.

[8] H. R. Berenji. Fuzzy logic controllers. In R. R. Yager and L.A. Zadeh, editors, *An Introduction to Fuzzy Logic Applications in*

Intelligent Systems, pages 69–96. Kluwer Academic Publishers, 1991.

[9] H. R. Berenji, Y. Y. Chen, and R.R. Yager. Using new aggregation operators in rule-based intelligent control. In *29th IEEE Conference on Decision and Control*, pages 2198–2203, Honolulu, Hawaii, 1990.

[10] H.R. Berenji. An architecture for designing fuzzy controllers using neural networks. *International Journal of Approximate Reasoning*, 6(2):267–292, February 1992.

[11] H.R. Berenji, Y.Y. Chen, C.C. Lee, J.S. Jang, and S. Murugesan. A hierarchical approach to designing approximate reasoning-based controllers for dynamic physical systems. In *Sixth Conference on Uncertainty in Artificial Intelligence*, pages 362–369, 1990.

[12] H.R. Berenji and P. Khedkar. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Transactions on Neural Networks*, 3(5), 1992.

[13] M. Braae and D.A. Rutherford. Fuzzy relations in a control setting. *Kybernetes*, 7, no. 3:185–188, 1978.

[14] Y.Y. Chen. Stability analysis of fuzzy control—a Lyapunov approach. In *IEEE Systems, Man, Cybernetics, Annual Conference*, volume 3, pages 1027–1031, 1987.

[15] P. J. King and E. H. Mamdani. The application of fuzzy control systems to industrial processes. *Automatica*, 13(3):235–242, 1977.

[16] J.B. Kiszka, M.M. Gupta, and P.N. Nikiforuk. Energistic stability of fuzzy dynamic systems. *IEEE Trans. Systems, Man and Cybernetics*, SMC-15(6), 1985.

[17] G. J. Klir and T. A. Folger. *Fuzzy Sets, Uncertainty, and Information.* Prentice Hall, New Jersey, 1988.

[18] G.R. Langari and M. Tomizuka. Stability of fuzzy linguistic control systems. In *IEEE Conference on Decision and Control*, Hawaii, December 1990.

[19] R. Langari and H.R. Berenji. Fuzzy logic in control engineering. In David White and Donald Sofege, editors, *Handbook of Intelligent Control.* Van Nostrand, 1992 (to appear).

[20] C.C. Lee and H.R. Berenji. An intelligent controller based on approximate reasoning and reinforcement learning. In *Proc. of IEEE Int. Symposium on Intelligent Control*, Albany, NY, 1989.

[21] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.

[22] J.M. Mendel and R.W. McLaren. Reinforcement-learning control and pattern recognition systems. In J.M. Mendel and K.S. Fu, editors, *Adaptive, learning and pattern recognition systems: Theory and applications*, pages 287–318. Academic Press, New York, 1970.

[23] D. Michie and R. A. Chambers. Boxes: An experiment in adaptive control. In J.T. Tou and R. H. Wilcox, editors, *Machine Intelligence*, volume 2, pages 137–152. Oliver and Boyd, Edinburgh, 1968.

[24] K. Narendra and S. lakshmivarahan. Learning automata - a critique. *Journal of Cybernetics, and Information Science*, 1:53–65, 1977.

[25] K. Narendra and M.A.L. Thathachar. *Learning Automata: An Introduction.* Prentice Hall, Englewood, Cliffs, N.J, 1989.

[26] T. J. Procyk and E. H. Mamdani. A linguistic self-organizing process controller. *Automatica*, 15(1):15–30, 1979.

[27] A. L. Samuel. Some studies in machine learning using the game of checkers. Journal of R&D, IBM, 1959.

[28] M. Sugeno. Current projects in fuzzy control. In *Workshop on Fuzzy Control Systems and Space Station Applications*, pages 65–77, Huntington Beach, CA, 14-15 November 1990.

[29] M. Sugeno. An introductory survey of fuzzy control. *Information Science*, 36:59–83, 1985.

[30] M. Sugeno and K. Murakami. An experimental study on fuzzy parking control using a model car. In M. Sugeno, editor, *Industrial Applications of Fuzzy Control*, pages 125–138. North-Holland, Amsterdam, 1985.

[31] R.S. Sutton. *Temporal Credit Assignment in Reinforcement Learning.* PhD thesis, University of Massachusetts, 1984.

[32] R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[33] R.S. Sutton, A.G. Barto, and R.J. Williams. Reinforcement learning is direct adaptive optimal control. In *American Control Conference*, Boston, MA, 1991.

[34] H. Takagi and Isao Hayashi. Artificial neural network driven fuzzy reasoning. *Int. J. of Approximate Reasoning*, 5(3):191–212, 1991.

[35] T. Takagi and M. Sugeno. Derivation of fuzzy control rules from human operator's control actions. In *IFAC Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis*, pages 55–60, Marseille, France, 1983.

[36] M.L Tsetlin. *Automaton Theory and Modeling of Biological Systems.* Academic Press, New York, 1973.

[37] C.J.C.H. Watkins. *Learning with Delayed Rewards.* PhD thesis, Cambridge University, Psychology Department, 1989.

[38] B. Widrow and S.D. Stearns. *Adaptive Signal Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1985.

[39] Murayama. Y. and T. Terano. Optimising control of disel engine. In M. Sugeno, editor, *Industrial Applications of Fuzzy Control*, pages 63–72. North-Holland, Amsterdam, 1985.

[40] R. R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1), 1988.

[41] S. Yasunobu and S. Miyamoto. Automatic train operation by predictive fuzzy control. In M. Sugeno, editor, *Industrial Applications of Fuzzy Control*, pages 1–18. North-Holland, Amsterdam, 1985.

[42] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[43] L.A. Zadeh. A fuzzy-set-theoretic interpretation of linguistic hedges. *Journal of Cybernetics*, 2:4–34, 1972.