The Fuzzy Classifier System: A Classifier System for Continuously Varying Variables

Manuel Valenzuela-Rendón
mvalenzu@mtecv2.mty.itesm.mx
Centro de Inteligencia Artificial
Instituto Tecnológico y de Estudios Superiores de Monterrey
Sucursal de Correos "J," C.P. 64849 Monterrey, N.L., Mexico

Abstract

This paper presents the fuzzy classifier system which merges the ideas behind classifier systems and fuzzy controllers. The fuzzy classifier system learns by creating fuzzy rules which relate the values of the input variables to internal or output variables. It has credit assignment mechanisms which reassemble those of common classifier systems, but with a fuzzy nature. The fuzzy classifier system employs a genetic algorithm to evolve adequate fuzzy rules. Preliminary results show that the fuzzy classifier system can effectively create fuzzy rules that imitate the behavior of simple static systems.

1 INTRODUCTION

In spite of the potential of classifier systems as a learning paradigm, they have found little application in the adaptive control of processes. It can be said that this is due in part to the limitations of the classifier syntax when representing continuously varying variables. Fuzzy controllers are a controller design approach which is not based on a mathematical description of the process being controlled. A fuzzy controller models the knowledge used by human operators as a set of rules in which variables take linguistic values [13].

In a fuzzy controller, relations between inputs and outputs are expressed as fuzzy rules. The fuzzy controller implements a fuzzy relation between all the possible values of the inputs and the indicated values of the outputs. This concept has been successfully applied in many occasions: control of a cement kiln [5], traffic control [9], robot arm control [11], and temperature control of a heated air-stream [6], are only a few examples. In a typical fuzzy controller [7], fuzzy rules are derived either from manuals of operation or from human operators that have successfully controlled the system and which have acquired their rules through experience. Algorithms that modify the fuzzy relation implemented by the fuzzy controller have been proposed [8, 10]. The fuzzy classifier system (FCS) is motivated by the ideas of fuzzy controllers, but departs from previous implementations in the manner in which it creates new rules and adjusts the contribution of the existing rules to the system outputs.

The FCS merges the credit assignment mechanisms of common classifier systems [4, 2] and the use of fuzzy logic of fuzzy controllers. It represents its fuzzy rules as binary strings on which a genetic algorithm operates, therefore allowing for the evolution of adapted rule sets. The FCS allows inputs, outputs, and internal variables to take continuous values over given ranges, thus it could be applied for the identification and control of dynamic systems. In the rest of this paper, the FCS will be described. Preliminary results will be presented that show that the FCS can successfully identify simple static systems.

2 FUZZY LOGIC

Similarly to fuzzy controllers, the FCS represents all its knowledge by means of fuzzy rules. To introduce the concept of fuzzy rules, let us first review fuzzy sets and their operations.

2.1 FUZZY SETS AND OPERATIONS

Fuzzy set theory [1, 13] can be defined as a generalization of common set theory in which the membership of an element to a set is defined by a membership function. In this way, an element can partially belong to a set. In fuzzy set parlance, common sets are also called *crisp*.

Set operations can be defined over fuzzy sets analogously to crisp set operations. The generalization of these crisp set operations to fuzzy sets can be performed in different ways. The following definitions are

one of such possible manners, they constitute a consistent framework, and they are the most commonly used.

Let A and B be fuzzy sets over the variable $x \in [x_0, x_f]$ where $x_0, x_f \in \Re$. Let A and B be defined by the membership functions $\mu_A(x) \in [0,1]$ and $\mu_B(x) \in [0,1]$, respectively.

Definition 1 *The union of A and B, denoted by $A \cup B$, is defined by the following membership function:*

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

Definition 2 *The intersection of A and B, denoted by $A \cap B$, is defined by the following membership function:*

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

Definition 3 *The complement of A, denoted by A', is defined by the following membership function:*

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

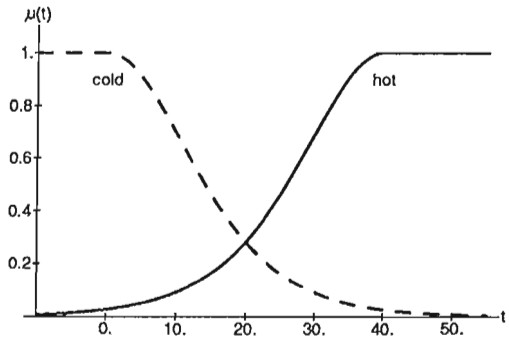

Figure 1: Membership functions of sets *cold* and *hot*.

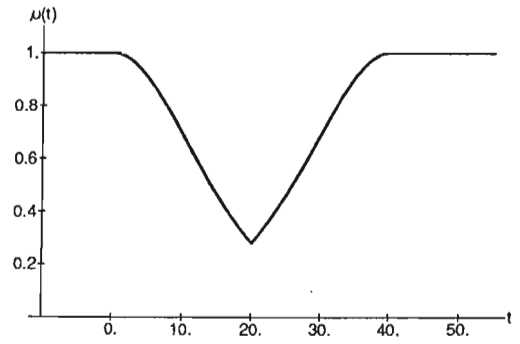

Figure 2: Fuzzy union of sets *cold* and *hot*.

For example, consider two fuzzy sets, *cold* and *hot*, over the variable t (temperature in Celsius degrees), and defined by the membership functions shown in Figure 1. The membership functions of the union and the intersection of these sets are shown in Figures 2 and 3.

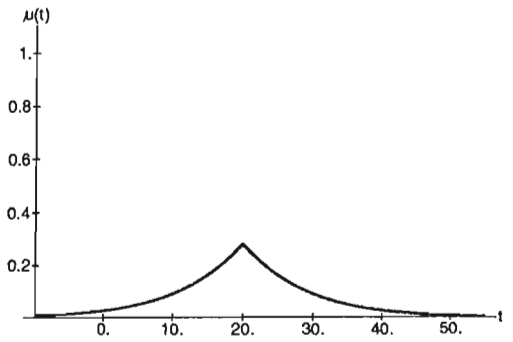

Figure 3: Fuzzy intersection of sets *cold* and *hot*.

2.2 FUZZY RULES AND RELATIONS

A fuzzy rule is an *if–then* expression in which conditions and action are fuzzy sets over given variables. Fuzzy rules are also called linguistic rules, because they represent the way in which people usually formulate their knowledge about a given process. The following are examples of fuzzy rules:

if the temperature is high,
then slightly reduce the gas intake;

if the pavement is very wet,
then moderately reduce your driving speed.

A fuzzy relation maps one or more independent variables into a dependent variable. A set of fuzzy rules, in which the antecedents refer to the same independent variables and the consequences refer to a same dependent variable, form a fuzzy relation. In other words, a way of expressing a fuzzy relation is by a set of fuzzy rules. Expressing a fuzzy relation as a set of fuzzy rules is computationally convenient because in general it requires less memory storage than expressing the relation in tabular form.

2.3 FUZZY RULES IN THE FCS

The FCS operates over variables that can be inputs, outputs, or internal. The FCS relates the values of the inputs and the internal variables and generates outputs according to fuzzy rules, similarly to a fuzzy controller. These rules or classifiers are represented as binary strings that encode the membership functions of the fuzzy sets involved in the fuzzy rule. To allow for a uniform procedure, the FCS linearly maps all variables to the range [0,1]. For each variable, n component fuzzy sets are defined so that their membership functions span the interval [0,1]. The number of these component sets is defined by the user according to the precision required. The peaks of the membership functions of the component sets that span a variable are equally spaced. The following expression defines the

membership function of the i-th fuzzy set for the variable $x \in [x_0, x_f]$:

$$\lambda(x, h_i) = \frac{4e^{-(x-h_i)/\sigma}}{\left(1 + e^{-(x-h_i)/\sigma}\right)^2}, \qquad (1)$$

where

$$h_i = (i-1)\frac{(x_f + \delta) - (x_0 - \delta)}{n-1} + (x_0 - \delta),$$

for $i = 1, 2, \ldots, n$. The parameter σ controls the width of the component sets. The user must choose a value of σ according to the value of n so that the interval is adequately covered. The parameter δ expands the effect of the component sets outside the range of x thus increasing the precision for values near x_0 and x_f. The user must choose a value of δ according to the sensitivity of his application to values of x close to x_0 or x_f.

All the membership function used by the FCS are generated from Equation 1 with different values of h and σ. Conditions and recommended actions are binary coded so that the number of bits in a condition or an action is the number of fuzzy sets defined over the given variable. A "1" indicates that the corresponding fuzzy set is part of the condition or action. Additionally, a non-fuzzy binary tag is attached. This tag indicates to which variable the condition or action is referring to. Tags are followed by a colon. Consider for example the classifier 0:110/1:001 in which three component sets are defined over variables x_0 and x_1. The membership functions of condition 0:110 and action 1:001 are shown in Figures 4 and 5. The rule

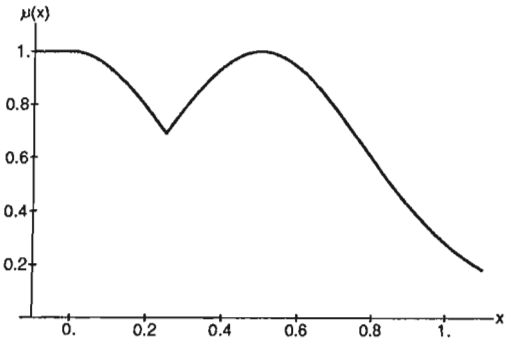

Figure 4: Membership function of condition 0:110.

represented by these sets can be expressed in words as "*if* x_0 is low or medium *then* x_1 should be high." Notice that the FCS syntax does not include the wildcard character #.

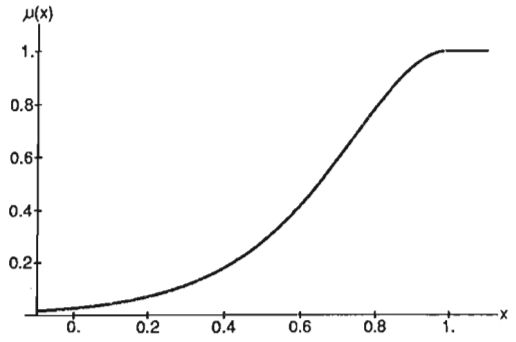

Figure 5: Membership function of action 1:001.

3 OPERATION OF THE FCS

The operation of the FCS is similar to that of a common classifier system. Figure 6 presents a block diagram of an FCS. The basic cycle of a FCS is as follows:

1. The input unit receives input values, encodes them into fuzzy messages, and adds these messages to the message list.
2. The classifier list is scanned to find all classifiers whose conditions are satisfied by the messages in the message list.
3. The message list is erased.
4. All matched classifiers are fired, and the messages produced are stored in the message list.
5. The output unit detects the output messages, and erases these messages from the message list.
6. In the output unit, output messages are decomposed into minimal messages.
7. Minimal messages are *defuzzified* and transformed into output values.
8. Payoff from the environment and classifiers is transmitted through the messages to the classifiers.

The following subsections explain the operation details of the FCS.

3.1 FUZZY MESSAGES AND FUZZY MATCHING

The values taken by variables are broadcast to the classifiers as *messages*. Each classifier will verify if its conditions are matched, i.e. satisfied, by the messages, and if so, will post a new message according to its indicated action.

There are two alternative and equivalent ways in which the process of a fuzzy rule being matched can be viewed. First, a fuzzy rule can be thought of as receiving real valued variables, and performing a fuzzy

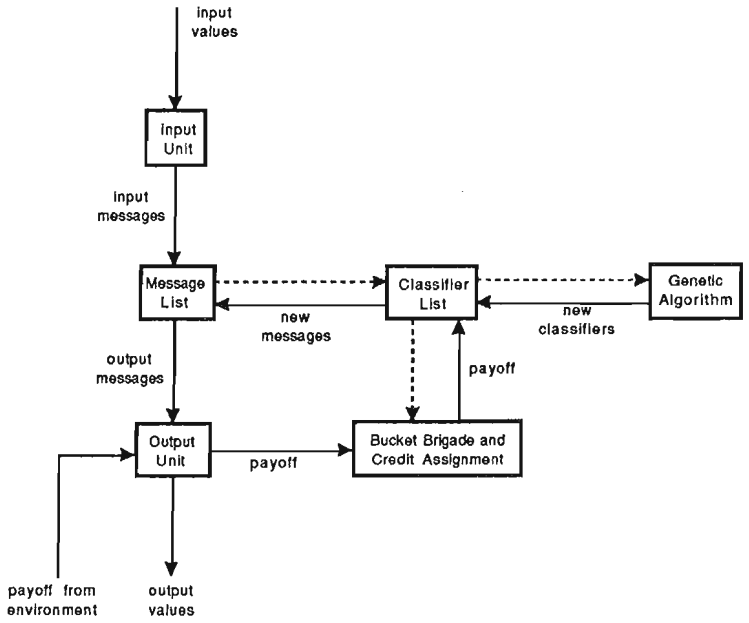

Figure 6: Block diagram of the FCS.

match between the values of these variables and the fuzzy sets defined in its condition. A second view is possible, one more convenient when handling sets of rules. A fuzzy rule can be thought of as receiving fuzzy values, and performing a perfect match between these fuzzy values and its condition.

Suppose for example two classifiers, 0:010/1:100 and 0:001/1:001, each with a condition over the input variable x_0 and an action over the output variable x_1; also suppose that x_0 takes the value of 0.3. It is equivalent to say that classifier 0:010/1:100 is matched with a degree of $\lambda(0.3, h_2)$ and classifier 0:001/1:001 is matched with a degree of $\lambda(0.3, h_3)$, or to say that the messages 0:010 and 0:001 are present with *activity levels* of $\lambda(0.3, h_2)$ and $\lambda(0.3, h_3)$ respectively, and that classifiers 0:010/1:100 and 0:001/1:001 are perfectly matched by these messages.

The FCS implements the second approach by having an input unit that fuzzifies inputs into fuzzy messages. Each message has an associated activity level which measures the degree of belonging of the input variable to the fuzzy set defined by the membership function represented by the message. Fuzzy classifiers match these messages and generate new messages with activity levels that correspond to the degree in which the conditions of the classifier are satisfied. This *fuzzification* of inputs is accomplished by creating *minimal messages* one for each fuzzy set defined over the variable. A minimal message has a single "1."

The matching of a condition in a classifier by a message is done in two steps. First, the tags of the message and condition are compared; if they are the same they refer to the same variable, if so, the rest of the message and condition are compared. Second, if at least there is one position in which the condition has a "1" and the message also has a "1," then the condition is satisfied.[1] The satisfaction level of a condition is equal to the maximum activity level of the messages that match this condition; this implements a fuzzy union. The activity level of a classifier is equal to the minimum of the satisfaction levels of all its conditions; therefore, implementing a fuzzy intersection. When a classifier fires, it generates a new message with an activity level proportional to the classifier's own activity level.

3.2 DEFUZZIFICATION IN THE OUTPUT UNIT

The FCS must translate messages referring to output variables into real values. This process of *defuzzification* is accomplished in the *output unit* by decomposing each output message into its corresponding minimal messages. The activity levels of minimal messages corresponding to the same variable and component set are added up and the messages are substituted by a single message. For each output variable, a fuzzy union is performed over the component sets represented by the minimal messages multiplied by their activity levels. Then, the gravity center of the union is taken. This

[1] This can be explained by recalling that a condition is a fuzzy union of the fuzzy sets that correspond to the bits that are "1." A condition as 0:101 in words would be "variable 0 takes the value low or high." This is satisfied by any of the messages 0:100, 0:001, 0:101, 0:110, 0:011, or 0:111.

gravity center is the output value.

The total gravity center can be obtained by first calculating the gravity center of the contribution of each minimal message. Then the total gravity center G is calculated as:

$$G = \frac{\sum A_i g_i}{\sum A_i},$$

where A_i is the area contribution of the i-th minimal message, and g_i is the gravity center of A_i. Figure 7 shows the intersection of three minimal messages with activity levels of 1.0, 0.6, and 0.8. A_1, A_2, and A_3 are the area contributions of these messages.

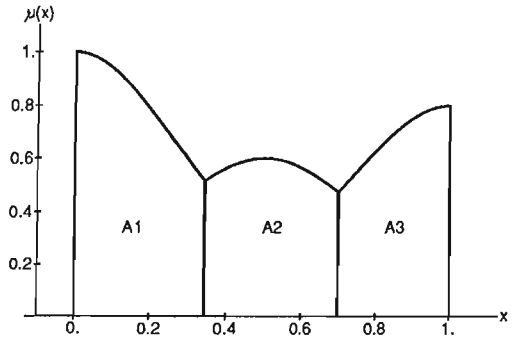

Figure 7: Intersection of three minimal messages and their area contributions.

3.3 CREDIT ASSIGNMENT AND FUZZY BUCKET BRIGADE

After the output unit has produced an output value, the environment judges this output, and accordingly, gives payoff to the output unit. The output unit distributes this payoff through the minimal messages to the classifiers that produced the output messages. The payoff is distributed in such a manner that classifiers that contributed more to the output taking a specific value receive a larger share of the payoff. In this way, classifiers that are directly involved in producing correct outputs receive increases in their strength.

Payoff to other classifiers that do not produce outputs, but post messages that allow for others to fire and receive payoff, is distributed following the *bucket brigade algorithm* of common classifier systems. According to the bucket brigade algorithm, matched classifiers *bid* a small portion of their strength for the right to fire. Firing classifiers pay their bids to those that posted the messages that allowed them to fire. In this way, a competitive economic system is established in which classifiers that produce or help produce good outputs have their strengths increased, and other have their strengths decreased. A basis against classifiers that do not participate in the competition is achieved through a living tax by which all classifiers have a small portion of their strength deducted every time step.

3.4 CREATION OF NEW RULES THROUGH A GENETIC ALGORITHM

In a FCS, classifiers are selected by a genetic algorithm (GA) [2, 3] for reproduction according to their strength: stronger classifiers are selected more than weaker ones. Pairs of classifiers for mating are obtained by choosing classifiers randomly according to their selection probabilities. Each couple reproduces, creating new rules by *crossover*. After crossover, mutation occurs with a specified small probability. To keep constant the size of the classifier list, weak classifiers are deleted.

4 A LEARNING TASK

As a first test for the FCS, a simplified task was chosen: the imitation of static one-input one-output systems in which the output depends only on the present input and not on past states of the system.

4.1 IDENTIFICATION WITH A STIMULUS-RESPONSE FCS

For this task we only require a stimulus-response FCS. A stimulus-response FCS is one in which classifiers post only output messages, and classifiers only respond to input messages. In this way, the bucket brigade does not need to operate. The FCS and the system are setup as shown in Figure 8. The FCS and the system to be identified receive the same input generated randomly among a range of possible inputs by an *input generator*. The output of the FCS is compared to that of the system, and a payoff is assigned to the FCS by a *performance evaluator*. The FCS receives higher payoff when it best imitates the behavior of the system. In this way, the FCS constructs a fuzzy model of the system.

The performance evaluator assigns payoff to the FCS output according to the following equation:

$$P = P_0|u - y|,$$

where P_0 is a constant, u is the FCS output, and y is the system output. The output unit distributes this payoff to the minimal output messages. Each minimal message receives a portion according to its contribution to the output.

The i-th minimal message receives a payoff given by the following expression:

$$p_i = \phi(-\mathrm{sgn}(\varepsilon_o)\varepsilon_i) \cdot P,$$

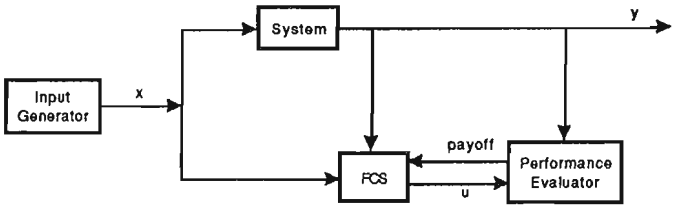

Figure 8: FCS in the identification setup.

where

$$\phi(x) = \begin{cases} -\eta_1 & \text{if } x \leq -\eta_2; \\ 1 & \text{if } x \geq \eta_2; \\ (1+\eta_1)(x+\eta_2)^2/4\eta_2^2 - \eta_1 & \text{otherwise.} \end{cases}$$

The output error ε_o and message error ε_i are calculated as follows:

$$\begin{aligned} \varepsilon_o &= u - y; \\ \varepsilon_i &= h_i - y. \end{aligned}$$

To understand these expressions, note that the product $-\text{sgn}(\varepsilon_o)\varepsilon_i$ is positive if, and only if, ε_0 and ε_i have opposite signs. Furthermore, note that this implies that one of the following is true: $u < y < h_i$ or $h_i < y < u$. Therefore, an adequate increase in the i-th minimal message activity level could reduce the output error to zero. Function $\phi(x)$ gives a large positive payoff to messages satisfying the above conditions, and gives a lower payoff to all others. The constants η_1 and η_2 must be chosen according to the problem.

This payoff distribution scheme assures that minimal messages that would contribute to minimize the error will receive a higher payoff.[2]

4.2 PAYOFF DISTRIBUTION IN THE STIMULUS-RESPONSE FCS

In the stimulus-response FCS, the specificity of the classifiers is involved in the payoff distribution scheme. A specificity ρ is defined only for the action of a classifier in the following manner:

$$\rho = \left(\frac{1 + \text{number of 0's}}{\text{action length}} \right)^2.$$

A message has a specificity equal to that of the classifier that posted it.

Each minimal message distributes its payoff to its corresponding messages according to their activity level and to their specificity. The payoff to a message m is the sum of all the payoffs it receives from its minimal messages:

$$\text{payoff to message } m = \sum_{i \in \mathcal{M}} \frac{p_i \, \rho_m \, (\text{activity level}_m)}{\text{activity level}_i},$$

where $\mathcal{M}$ is the set of minimal messages posted by message m. The activity level of a message is equal to the sum of the activity level multiplied by the bid of the classifiers who posted it.

Firing classifiers pay a bid B equal to a small fraction k of their strength S. Each classifier receives a payment proportional to its specificity and activity level, the payoff received by the message it posted, and divided by the number of classifiers that posted the same message.

4.3 RESULTS

The stimulus-response FCS just described was trained to imitate a system in which the output is equal to the input, i.e., $y = x$. The value of δ was set to 0 for the input and 0.2 for the output. The parameters $\sigma = 0.11$, $\eta_1 = 0.15$, and $\eta_2 = 1.0$ were obtained by trial and error. The genetic algorithm ran over a population of 40 classifiers.

After learning, the payoff distribution scheme and the genetic algorithm were turned off, and the input range was scanned to obtain the input-output behavior of the FCS. Figure 9 shows the performance obtained from the fuzzy rules found by the FCS using four fuzzy intervals after 64,000 cycles. The FCS approximates the straight line with an absolute error[3] of 1.72%.

[2] This payoff distribution scheme involves information about the correct output, and thus, it is not a blind reinforcement scheme.

[3] The absolute error was calculated as

$$\text{absolute error} = \frac{1}{x_f - x_0} \int_{x_0}^{x_f} |u - y| \, dx.$$

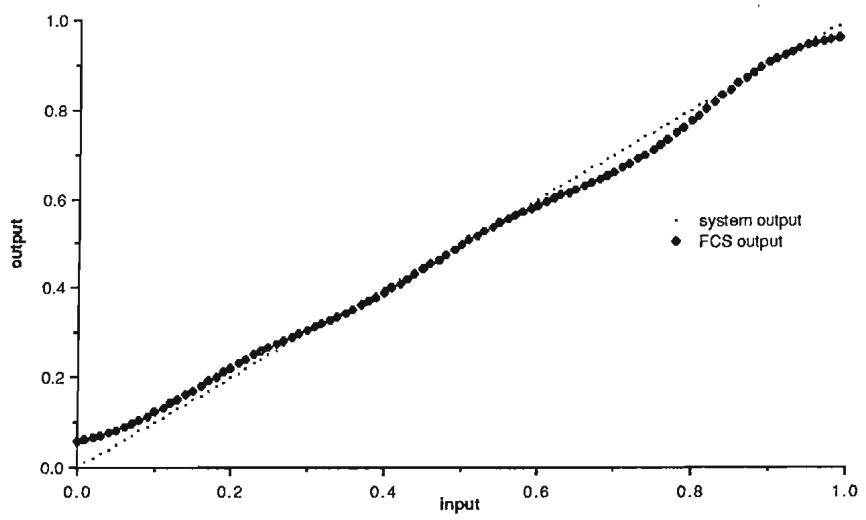

Figure 9: FCS imitation of the straight line $y = x$ using four fuzzy component sets.

When analyzing this result, we must consider the following step-ladder function:

$$\gamma(x) = \begin{cases} 1/2n & \text{if } 0 \leq x < 1/n; \\ 3/2n & \text{if } 1/n \leq x < 2/n; \\ \vdots & \\ (2n-1)/2n & \text{if } (n-1)/n \leq x < 1. \end{cases}$$

A learning system based on crisp intervals would tend to approximate the function $y = x$ with the step-ladder function. For a given number of intervals, the FCS finds a set of fuzzy rules that produces a smaller error than the step-ladder function.

The FCS was also setup to imitate the function $y = 4(x - 0.5)^2$. Figure 10 shows the performance of the FCS after 53,000 cycles. The absolute error was of 3.76%.

The previous results indicate that the FCS can learn to imitate simple, static systems. More experiments, involving other more complex systems, are required.

5 FUTURE WORK

Many issues remain open for research. Among the most important are increasing speed convergence while retaining stability; implementing pure reinforcement learning, so control as opposed to identification can be performed; allowing rule chaining, so that dynamic systems can be identified and controlled; allowing the FCS to adaptively change the membership functions of the component sets, so that greater sensitivity can be achieved where required; and implementing a scheme for continuous time output.

Efforts are currently being made to improve the convergence speed of the FCS while retaining stability. The following enhancements are now under evaluation.

- Mating restrictions
  For each classifier to be crossed, the genetic algorithm selects n classifiers and then chooses the one which is most similar to the classifier to be crossed.

- Replacement restriction
  When a classifier is created, the genetic algorithm selects the m weakest classifiers, then it deletes the one most similar to the new born classifier.

- Aging classifiers
  Classifiers have an age equal to the number of time cycles since their creation. Age limits for reproduction and death are defined. Classifiers under age are not selected for reproduction or deletion by the genetic algorithm.

- Limiting the weights of young classifiers
  Classifiers under a given age have a reduced weight in the decision of an output. The reduction is proportional to the difference between the age of the classifiers and a given age limit. This is possible because all classifiers help produce a single output. Reducing the weight of young classifiers on the output should improve the stability of the system.

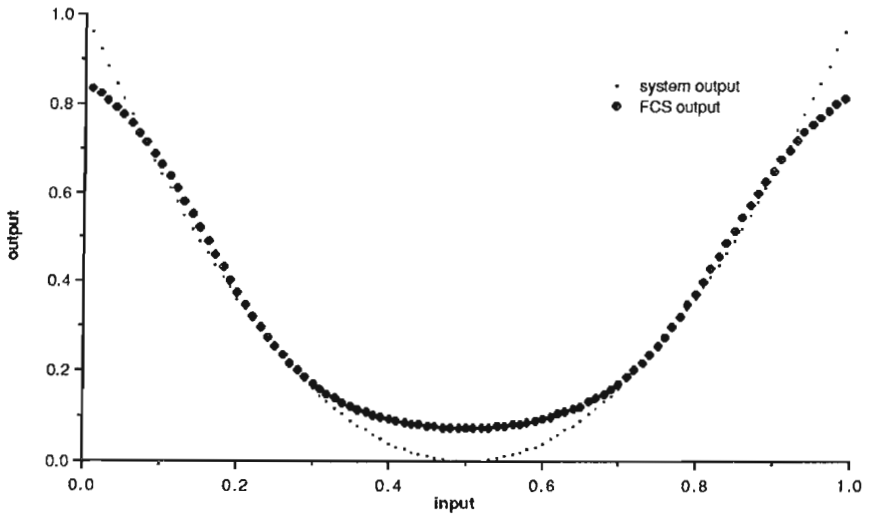

Figure 10: FCS imitation of the parabola $y = 4(x - 0.5)^2$ using four fuzzy component sets.

Acknowledgements

The author gratefully acknowledges the useful comments received from the reviewers.

References

[1] D. Dubois and H. Prade. *Fuzzy Sets and Systems: Theory and Applications.* Academic Press, New York, 1980.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[3] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.

[4] J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard. *Induction: Processes of Inference, Learning, and Discovery.* MIT Press, Cambridge, 1986.

[5] J.-J. Holmblad, L. P. Østergaad. Control of a cement kiln by fuzzy logic. In M. M. Gupta and E. Sánchez, editors, *Fuzzy Information and Decision Processes*, pages 389–399. North-Holland, Amsterdam, 1982.

[6] A. Ollero and A. J. García-Cerezo. Direct digital control, auto-tuning and supervision using fuzzy logic. *Fuzzy Sets and Systems*, 30:135–153, 1988.

[7] W. Pedrycz. *Fuzzy Control and Fuzzy Systems.* John Wiley, New York, 1989.

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

[9] T. Saski and T. Akiyama. Traffic control process of expressway by fuzzy logic. *Fuzzy Sets and Systems*, 26:165–178, 1988.

[10] S. Shao. Fuzzy self-organizing controller and its application for dynamic processes. *Fuzzy Sets and Systems*, 26:151–164, 1988.

[11] R. Tanscheit and E. M. Scharf. Experiments with the use of a ruled-based self-organising controller for robotics applications. *Fuzzy Sets and Systems*, 26:195–215, 1988.

[12] M. Valenzuela-Rendón. The fuzzy classifier system: motivations and first results. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, 330–334, Springer (Verlag), Berlin, 1991.

[13] H. J. Zimmermann. *Fuzzy Set Theory—and Its Applications.* Kluwer, Boston, 1988.