Fuzzy Logic Synthesis with Genetic Algorithms

Philip Thrift
Central Research Laboratories
Texas Instruments
P.O. Box 655936 M.S. 134
Dallas, Texas 75265
214 995-7906 thrift@resbld.ti.com

Abstract

This paper considers the application of a genetics-based learning algorithm to systems based on fuzzy logic. One of the more active areas in the application of fuzzy logic is fuzzy controllers. A fuzzy logic controller (FLC) is based on linguistic control strategies (or rules) that interface with real sensor and activator signals by means of fuzzification and defuzzification algorithms. The discrete nature of fuzzy strategies make them prime candidates for discovery by genetic algorithms. This approach is explored in this paper. Some general directions for genetics-based machine learning in fuzzy systems are outlined.

1 INTRODUCTION

There has been much recent activity in the use of fuzzy logic in the design of controllers from braking systems for trains to washing machines [10]. These systems are for the most part designed by a knowledge engineering process based on subjective experience and trial-and-error experimentation. There is much interest in the use of learning algorithms (including genetic algorithms [4,5]) to automatically synthesize such systems. This paper considers the use of genetics-based learning as a general approach to synthesizing fuzzy logic strategies. A genetic algorithm applied to a simple control example (cart centering) is presented.

2 FUZZY LOGIC CONTROLLERS

In this section we briefly review the basic concepts of fuzzy logic, and fuzzy logic controllers (FLCs) in particular. A general survey of the field as well as a fuzzy logic background can be found in [9].

Fuzzy logic is based on the concept of fuzzy sets [11]. A fuzzy set is a generalization of a classical set in that memberships are graded between 0 and 1 as opposed to being purely boolean. If x is some variable over some domain of discourse U, and X is a fuzzy set over U, then $\mu_X(x)$ is defined to be the degree of membership of x in X. As an example, U could be some measurable parameter of a system, such as **temperature**. X could be a fuzzy set, such as **cool**. Then $\mu_{\textbf{cool}}(x)$ would be a number between 0 and 1 inclusive as indicated in Figure 1.

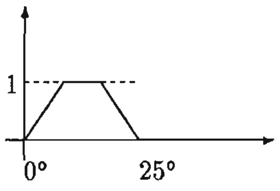

Figure 1: The fuzzy set **cool** over the domain **temperature**

Fuzzy set operations are a generalization of classical set operations. In particular,

$$\begin{aligned}\mu_{X_1 \cap X_2}(x) &= \min\{\mu_{X_1}(x), \mu_{X_2}(x)\} \\ \mu_{X_1 \cup X_2}(x) &= \max\{\mu_{X_1}(x), \mu_{X_2}(x)\} \\ \mu_{\overline{X}}(x) &= 1 - \mu_X(x)\end{aligned}$$

are definitions for fuzzy intersection, union, and complement. These are not the only ones (there is a product rule for intersection for example), but these are fairly standard.

Given a domain of discourse U, fuzzy sets over U can be identified with a set of names of linguistic variables. For example if U is identified with the parameter measuring **temperature**, then fuzzy sets over U can be

$$\{\textbf{cold}, \textbf{cool}, \textbf{moderate}, \textbf{warm}, \textbf{hot}\}.$$

Basic fuzzy sets may be modified by certain operators, such as *very*, or *slightly*, etc. These essentially change the shape of the membership function they modify.

A system of fuzzy sets over a domain can form a fuzzy partition of the domain as indicated in

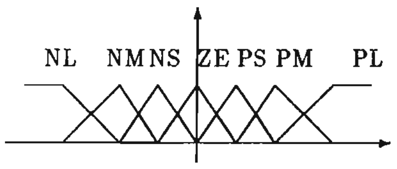

Figure 2: Fuzzy partitions of a domain

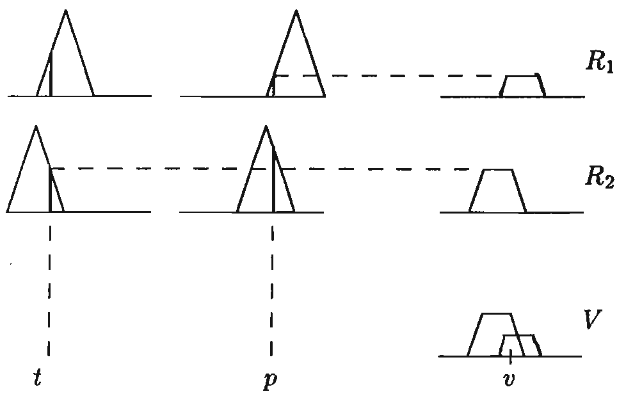

Figure 3: Fuzzy inference

Figure 2. Generically for linearly ordered parameter spaces, the fuzzy sets of a partition can be identified with lingustic variables such as **ZE**(zero), **PS**(positive small), **NS**(negative small), **NM**(negative medium), **PM**(positive medium), **NL**(negative large), **PL**(positive large). A subset of these can be chosen for a coarser resolution.

Given domains of discourse for a set of input and output variables for a system S, rules can be written in terms of fuzzy sets over these domains. Let us suppose that **temperature** (t) and **pressure** (p) are two input variables and the output variable is a **valve location** (v), then a fuzzy rule could be

if t is **hot** and p is **low** then turn v to **medium**.

A set of rules of this type constitute a fuzzy rule-base. Once a fuzzy rule-base has been specified, either by an expert or an adaptive procedure (such as a genetic algorithm), the system can map actual (crisp) input values to output values by means of fuzzification, fuzzy inference, and defuzzification. The system uses a method of inference called *sup-star composition*. This procedure is indicated in Figure 3. The fuzzification is performed by evaluating every input parameter with respect to the fuzzy sets in the premise of rules. For example in the above example, $\mu_{\mathbf{hot}}(t)$ and $\mu_{\mathbf{low}}(p)$ would be evaluated. They are combined by

$$s = \mu_{\mathbf{hot}}(t) * \mu_{\mathbf{low}}(p)$$

where $*$ is either the multiplication or minimum operator. This gives the degree to which that particular rule is selected. The output of the rule is the fuzzy set R defined by the function

$$\mu_R(v) = \mu_{\mathbf{medium}}(v) * s.$$

This procedure produces a fuzzy set for each rule ($R_1(v)$,$R_2(v)$,...). A single fuzzy set V is produced by taking the fuzzy union (or max): $V = R_1 \cup R_2 \cup \ldots$. A single value v is then produced by a defuzzification operator

$$v = \text{defuzzify}(V).$$

In practice, several defuzzification stategies have been used. A typical strategy is to take the centroid of area under the curve specified by the membership function of V. This is the strategy used in the example below.

Fuzzy logic systems (such as fuzzy controllers) thus allow conflicting rules to apply allowing consensus answers to be formulated. This effect can be acheived by using partial matching and confidence factors in standard expert systems, but the fuzzy approach is a less ad-hoc approach.

An FLC consists of three components: a *fuzzification interface* with sensors, a *fuzzy rule-base* for inferencing, and a *defuzzication interface* for activators (or decison variables). The role of fuzzification is to map a sensor or input signal x into fuzzy membership values – one for each fuzzy set in the universe of discourse of x. For low dimensional systems, the fuzzy control logic can be specified by a table as shown in Figure 4. Entries in the table can be blank indicating no fuzzy set output for the corresponding rule.

	NL	NM		ZE	PS	PM	PL
NL	NB	NM	PL		ZE	PS	ZE
NM	NB		PL	NM	ZE	PL	ZE
NS	NB	NM	PL		NL	PM	PS
	NM	PS	NS	ZE	NL	ZE	PM
PS		PS		ZE	NS	NL	PL
PM	ZE	PS	ZE	PS	NS	NL	ZE
	ZE	NS	ZE	PL	PM	NM	ZE

Figure 4: Fuzzy decision table

3 LEARNING FUZZY RULES AND MEMBERSHIP FUNCTIONS

There are a number of previous approaches to learning fuzzy rule sets for implementing control strategies. These are referenced in [9], as well as a discussion of their general concept. A genetic algorithm approach is presented in [4]. In [5], a genetic algorithm is used to find both the memberships functions (parameterized by the endpoints of the triangular shapes) as well as the fuzzy rules. In this paper only the learning of fuzzy rules are considered.

4 FLC SYNTHESIS WITH A GA

One of the main applications of genetic algorithms and genetics-based machine learning (GBML) systems (such as classifier systems) is control. The use of GAs in controller synthesis has ranged from using low level binary pattern languages [2] to high level rule languages as the expression of control stratgies [3,7]. The use of fuzzy rule expressions is somewhat of a middle ground between these two.

We shall only consider fuzzy control synthesis for decision table forms. We will consider a table as a genotype with alleles that are fuzzy set indicators over the output domain. The phenotype is produced by the behavior produced by the fuzzification, max-$*$ composition, and defuzzification operations. The eight gene values can be written {**NL, NM, NS, ZE, PS, PM, PL, _**}. The _ symbol indicates there is no fuzzy set entry at a position that it appears. A chromosome (genotype) is formed from the decision table by going rowwise and producing a string of numbers from the code set. Standard crossover and mutation operators can act on these strings. In the example below we will only take a subset of the fuzzy partition.

5 EXAMPLE

We consider the example of cart centering. This problem involves a cart with mass m that moves on a one dimensional track, as indicated in Figure 5. The state variable for this system are the **position** (x), and **velocity** (v). We assume in the dynamics of the model that the track is frictionless. The output of the controller is a **force** (F). The objective is, given an initial position and speed, to move the cart to zero position and velocity in minimum time.

The simulation for the cart is given by

$$x(t+\tau) \;=\; x(t) + \tau v(t)$$

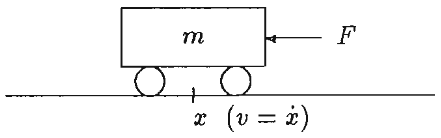

Figure 5: Cart centering example

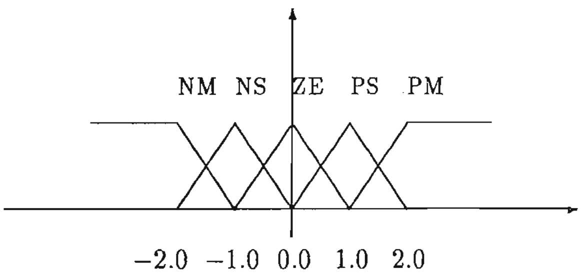

Figure 6: Fuzzy partitions for **position**, **velocity**, **force** (in meters, meters/sec, and newtons respectively)

$$v(t+\tau) \;=\; v(t) + \tau\frac{F(t)}{m}$$

Following [7], $\tau = .02$ sec, $m = 2.0$ kilograms. Position (velocity) is chosen randomly between -2.5 and 2.5 meters (meters/sec). For the first experiment, only five fuzzy partitions of each parameter were chosen. The fuzzy partitions for all three parameters (**position, velocity, force**) are chosen as indicated in Figure 6. The individual control strategies are are 5×5 tables, coded as chromosomes with alleles in $\{0,1,2,3,4,5\}$. (corresponding to **NM,NS,ZE,PS,PM**, and _ respectively. Based on the sup-$*$ algorithm above, a force F can be computed for a given (x, v). The defuzzification used is a simpilification of the centroid operator which computes a weighted average of the central points of the output fuzzy sets.

6 RESULTS

The fitness for an individual is determined by running a simulation of the cart for 500 steps (corresponding to 10 seconds for $\tau = 0.02$) with starting points (x_0, v_0) selected from 25 equally spaced positions. The fitness of an individual control strategy is measured by $500-T$, where T is average time required to be sufficiently close to $(0,0)$ in (x, v) coordinates (chosen as $\max(|x|,|v|) < .5$). If, for a given starting point (x_0, v_0), more than 500 steps are required, the process "times out", recording 500 steps.

These are the particular features of the genetic al-

				v		
		NM	NS	ZE	PS	PM
	NM	PM	PM	PM		
	NS	PM	PM	PM		
x	ZE	PM	PM		NM	NM
	PS	PM		NS		NM
	PM	PM	NM	NM	NM	NM

Figure 7: Fuzzy control strategy found in generation 100

gorithm based on the coding strategy described above. A mutation operator changes a fuzzy code either up a level or down a level, or to the blank code. (if it is already blank, then it chooses a non-blank code at random). The crossover operator is the standard two-point crossover [1]. A mutation rate of 0.01 and a crossover (two-point) rate of 0.7 were chosen. A elite strategy was taken whereby the best solution at a given generation is promoted directly to the next.

A simulation of 100 generations with a population of 31 individuals produces a solution which translates into the fuzzy decision table shown in Figure 7. This rule set compares well with the optimal "bang-bang" control rule [7], which is defined as follows. If $F(t)$ is chosen to be either F or $-F$, where F is some positive constant, then choose $F(t)$ to be F if

$$\frac{v^2 \mathrm{sgn}(v)}{2|\frac{F}{m}|} \quad < \quad -x$$

and $-F$ otherwise. In this simulation $F = m = 2.0$.

The control strategy in Figure 7 was compared with the optimal control strategy over 100 runs with random starting points in $-2.5 < x < 2.5, -2.5 < v < 2.5$. The average number of time steps (< 500) were 164 (corresponding to 3.28 sec) for the fuzzy controller and 143 (2.86 sec) for the optimal controller. A strategy succeeds when the cart is within the tolerance as specified above (0.5 in x and v). Keeping the same fuzzy control strategy and reducing the tolerance to 0.2 resulted and allowing 2000 steps before time-out resulted in an average of 242 (4.8 sec) for the fuzzy controller and 161 (3.2 sec) for the optimal controller. The controller could be further optimized by using a GA to place the endpoints of the triangular membership functions shown in Figure 6.

7 CONCLUSION

In this paper we have examined one aspect of using GAs and GBML in fuzzy systems, namely to design a simple fuzzy controller. Fuzzy systems research and applications have been experiencing a recent acceleration [10], so the usefulness of genetic approaches as these systems become more complex could become more apparent. Future directions remain: more complex rule structures with modifiers and quantification, the relation of fuzzy logic and classifier systems, and adaptive generation of the shapes of fuzzy membership functions themselves. In regards to classifier systems, a classifer of the form 751 / 3 could code a rule of the form

$$\text{if } x_2 \text{ is } \mathbf{PM} \text{ and } x_3 \text{ is } \mathbf{NM} \text{ then } u \text{ is } \mathbf{ZE}$$

Rule activation would be by fuzzification and defuzzification instead of the standard pattern matching. Also fuzzy algorithms [11] expressed in a programing language form could be more robust than with non-fuzzy programs (the problem raised in [8] notes the radical change of phenotype due to minor change in genotypes based on standard programs with non-fuzzy decisions). The interface between genetic algorithms and fuzzy systems should prove to produce useful results.

In the languange of genetic algorithms, the alleles are fuzzy set indicators. This allows discrete specification to have continuous and robust interactions with the environment via the *fuzzy phenotype*.

References

[1] Davis, L. ed. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold 1991.

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

[3] Grefenstette, J.J. A System for Learning Control Strategies with Genetic Algorithms. *International Conference on Genetic Algorithms.* 1989.

[4] Karr, C. Genetic Algorithms for Fuzzy Controllers. *AI Expert.* February 1991.

[5] Karr, C. Applying Genetics to Fuzzy Logic. *AI Expert.* March 1991.

[6] Kong, S-G. Kosko, B. Comparison of Fuzzy and Neural Truck Backer-Upper Control Systems. *IJCNN* 1990. III:349-358.

[7] Koza, J.R. Keane, M.A. Cart Centering and Broom Balancing by Genetically Breeding Populations of Control Strategy Programs. *IJCNN-90.* 1990.

[8] Koza, J.R. Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Populations of Computer Programs to Solve Problems. Stanford University Department of Computer Science Report No. STAN-CS-90-1314. 1990.

[9] Lee, C.C. Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Parts I & II. *IEEE Transaction on Systems, Man, and Cybernetics.* **20**:2. 1990. pages 404-435.

[10] Self, K. Designing with fuzzy logic. *IEEE Spectrum.* November 1990.

[11] Zadeh, L.A. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. *IEEE Transactions on Systems, Man, and Cybernetics* Vol. SMC-3, No. 1, January 1973.