

Presented at:
NASA Workshop on Computational Intelligence and Its Impact on Design/Fabrication
and Operation of Future High Performance Engineering Systems
Hampton, Virginia, June 1995

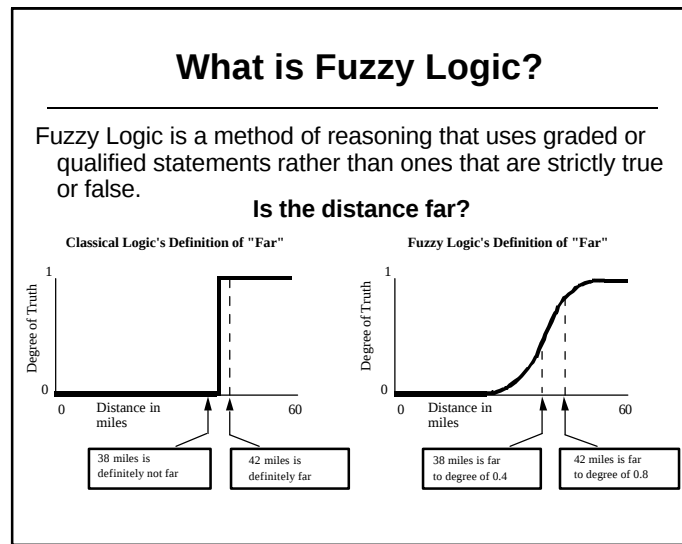
Trends and Issues in Fuzzy Control and Neuro-Fuzzy Modeling

Stephen Chiu

**Control & Information Management Dept.
Rockwell Science Center
1049 Camino Dos Rios
Thousand Oaks, CA 91360
Email: slchiu@rsc.rockwell.com**

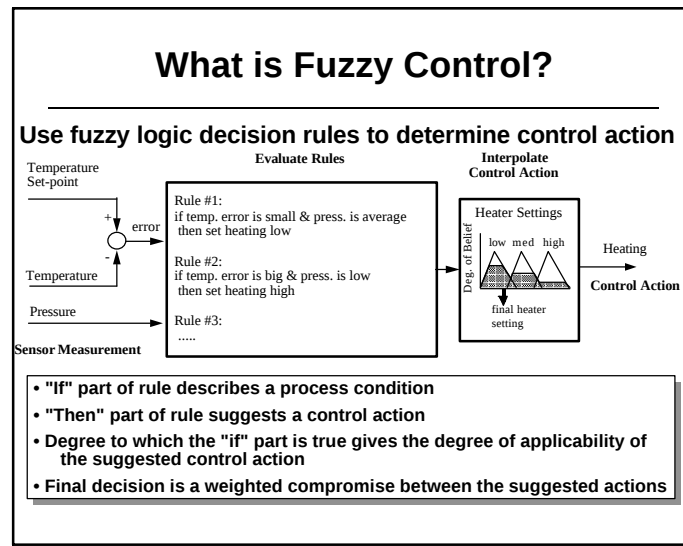
Introduction

Everyday experience in building and repairing things around the home have taught us the importance of using the right tool for the right job. Although we tend to think of a “job” in broad terms, such as “build a bookcase,” we understand well that the “right job” associated with each “right tool” is typically a narrowly bounded subtask, such as “tighten the screws”. Unfortunately, we often lose sight of this principle when solving engineering problems; we treat a broadly defined problem, such as controlling or modeling a system, as a narrow one that has a single “right tool” (e.g., linear analysis, fuzzy logic, neural network). We need to recognize that a typical real-world problem contains a number of different subproblems, and that a truly optimal solution (the best combination of cost, performance, and feature) is obtained by applying the right tool to the right subproblem. Here I share some of my perspectives on what constitutes the “right job” for fuzzy control and describe recent advances in neuro-fuzzy modeling to illustrate and to motivate the synergistic use of different tools.

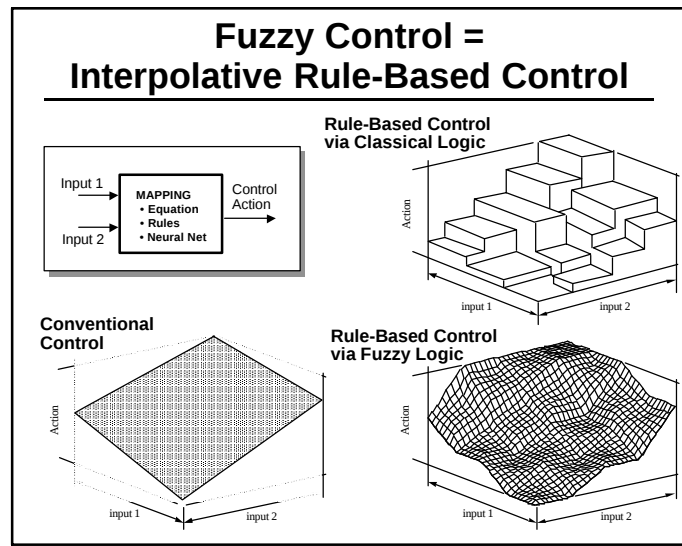


Because the reader may not be familiar with fuzzy control, I will start with a brief introduction to fuzzy control.

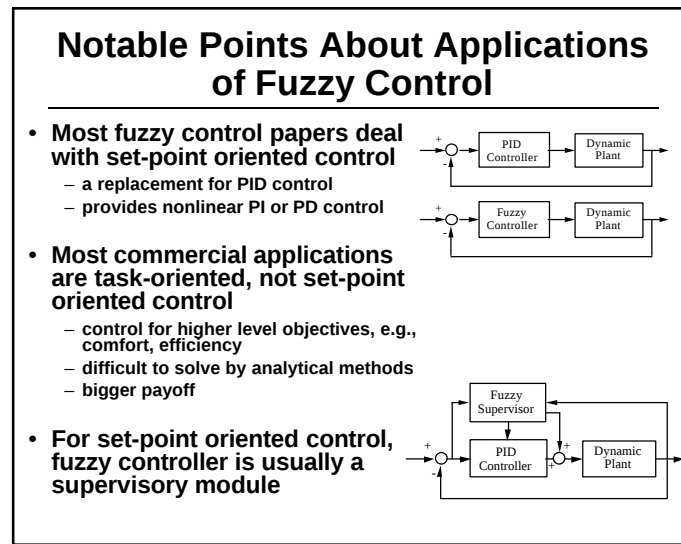
Fuzzy logic was developed to compensate for the limitations of classical logic. In classical logic, a statement can be either true or false, but nothing in between. For example, if we were to ask the question “is the distance to the market far?”, the answer can be only yes or no, based on a very precise definition of “far”. In classical logic, the definition of “far” would look something like the step function shown above. Here “far” is defined as any distance greater than or equal to 40 miles; hence, a distance of 39.999 miles would still be considered as definitely not far, but a location becomes definitely far as soon as it crosses over the 40 mile line. Fuzzy logic, on the other hand, allows a smooth, graded transition between totally true and totally false states. At the heart of fuzzy logic is the use of “membership functions” to define terms such as “far”, as shown above. Using this fuzzy definition of “far”, we can say that 38 miles is far to some degree (e.g., degree of 0.4), and 42 miles is also far, but to a greater degree (e.g., degree of 0.8).



Currently the most popular application of fuzzy logic is in automatic control. The basic idea of fuzzy control is to express the control knowledge of human experts as fuzzy "if-then" rules, such as "if the temperature error is small and pressure is average then set the heating to low." The "if" part of the rule describes a process condition, and the "then" part of the rule describes the control action for handling that particular condition. The degree to which the "if" part of the rule is true indicates the degree of applicability of the corresponding control action. Given a measured process condition, there are usually several control actions that are applicable, each with a different degree of applicability. The applicable control actions are combined to determine the final control action, usually by weighting each control action by its degree of applicability and computing a weighted-average type value.

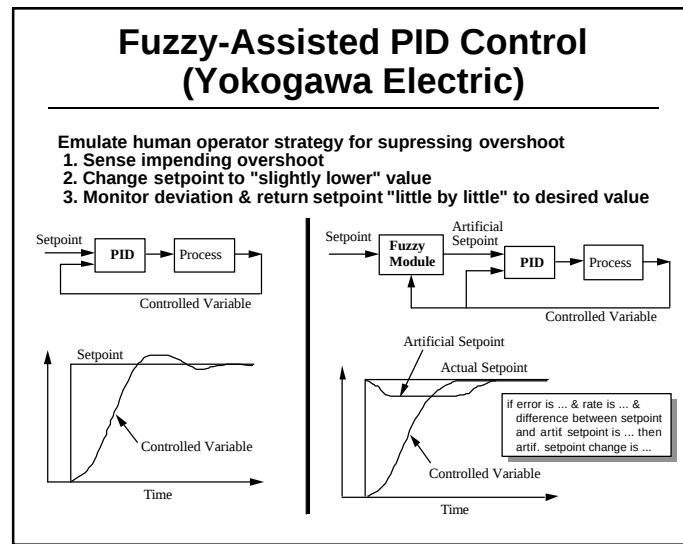


A control algorithm is basically a mapping that maps the controller's input values into output values. This mapping can be specified by analytical equations, decision trees, fuzzy rules, neural networks, or a combination thereof. Conventional linear control such as PID control produces a continuous, simple linear mapping; rule-based control employing classical logic produces a discontinuous mapping (due to step-like transitions), but a highly complex mapping can be easily specified by the designer through rules. Fuzzy control can be viewed as a cross between these two previous methods; the rule-based nature of fuzzy control allows a designer to easily specify a complex mapping, and the smooth transition and interpolation between fuzzy rules make the mapping continuous.



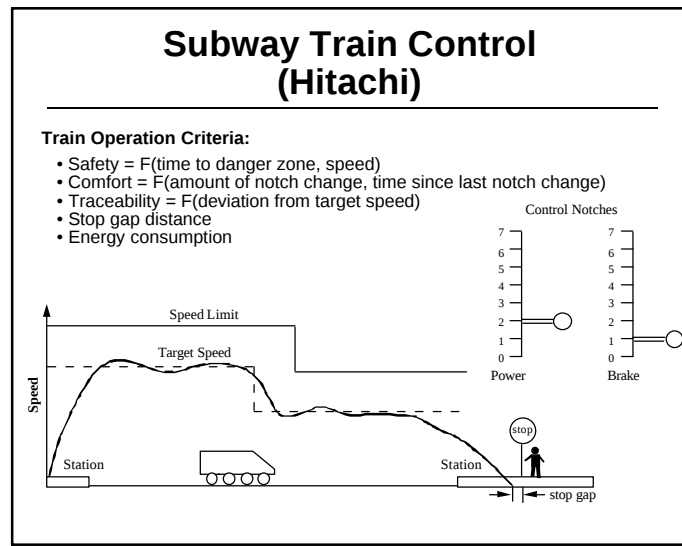
The fuzzy controllers described in most technical papers deal with the set-point regulation problem, where the control objective is to drive a process variable (e.g., motor shaft position, oven temperature) to a commanded set-point. If you read a paper on fuzzy control, chances are you will see that the inputs to the fuzzy controller are the set-point error and the rate of change of error, and the output is an actuator command or a change in actuator command. When used in this way, fuzzy control is not much different from conventional PID control - it is solving the same problems addressed by PID control and solving them in essentially the same way as PID control, except that fuzzy control provides a nonlinear input/output mapping. Hence, fuzzy control is often viewed as a form of nonlinear PD or PI control, and comparisons of fuzzy control versus conventional PID control abound in literature. However, when we look at commercial products where fuzzy control is said to be incorporated, we rarely see fuzzy logic being used to specify nonlinear PD or PI control; it is used mostly to handle high-level, task-oriented control functions that analytical control methods do not address (e.g., select the cycle time for a washing machine, select the gear for automatic transmission).

While fuzzy control can outperform conventional PID control to some degree, pursuing the use of fuzzy control as a form of nonlinear PID control should be limited to only a few special applications. The reason is that PID control is well-established and can satisfy the performance requirements of most set-point regulation problems at minimal cost. Because of the “establishment”, cost, and personnel training issues, there is little incentive to switch from conventional PID control to a more complex, nonlinear form of PID control unless the conventional controller is doing an unsatisfactory job. Therefore, commercial applications of fuzzy control are largely focused on task-oriented control rather than set-point regulation. For the instances where fuzzy logic is applied to set-point regulation, it is typically used in a high-level module that supervises or tunes a conventional PID controller.



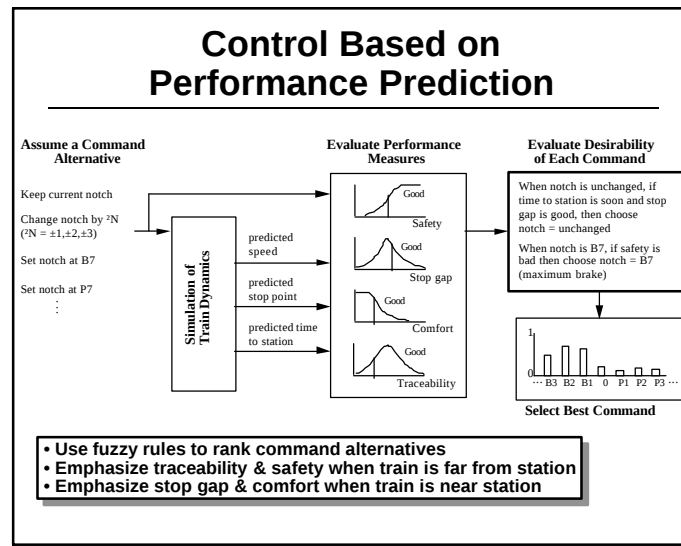
The temperature controller from Yokogawa Electric is a good example of how fuzzy logic is used commercially for set-point regulation. Temperature control usually involves processes that have a long time delay; for many processes, it is also imperative that the temperature do not overshoot the desired set-point. However, it is difficult to avoid overshoot when a process have a long time delay. For example, consider boiling milk on an electric stove; it is difficult to turn off the heat at just the right point to prevent the milk from boiling over. To ensure that the milk does not boil over, you will need to heat the milk very slowly at low heat. Similarly, PID controllers must employ very low gains to ensure that the temperature do not overshoot, and this leads to very slow time response.

In Yokogawa Electric's temperature controller, fuzzy logic is used to determine artificial set-points that are fed to a conventional PID controller. The PID controller is allowed to have high gains for fast time response. As the fuzzy module detects impending overshoot, it "fools" the PID controller by telling the PID controller to aim for a temperature value that is somewhat lower than the actual set-point. As the temperature rises to (and overshoots) the artificial set-point, the fuzzy module gradually raises the artificial set-point toward the actual set-point. In this way, the fuzzy module leads the PID controller along a temperature trajectory that can quickly reach the actual set-point without overshoot.



The subway train control system developed by Hitachi for the city of Sendai in Japan was among the first commercial applications of fuzzy logic; its success helped to galvanize the field fuzzy control and is often cited by proponents of fuzzy control. However, if we take a look at how fuzzy logic was actually used in this control system, we will see that its operation departs significantly from the input/output mapping notion of control, but is more closer to the notion of an expert system.

For this particular system, the train's acceleration/deceleration is controlled by setting a power lever and a brake lever at different notch positions. Changing the notch position frequently or in large increments creates an uncomfortable ride. In addition to riding comfort, a train operator must consider safety, on-time arrival, energy consumption, and stopping the train accurately at a specified position along the station platform. Here fuzzy logic was used to select the notch position that will best satisfy these multiple, often conflicting objectives.



The control method is based on predicting the outcome of each possible control action and then choosing the action that corresponds to the most desirable outcome. A simple simulation of the train dynamics is used to predict the resultant speed, stopping position, and time of arrival for each possible choice of notch position. Fuzzy rules then rank the desirability of each notch position based on the predicted outcome, taking into account factors such as the safeness of the resultant speed, arrival time, stopping position, amount of notch change, and the elapsed time since the last notch change. The notch position that received the highest ranking is selected as the notch command.

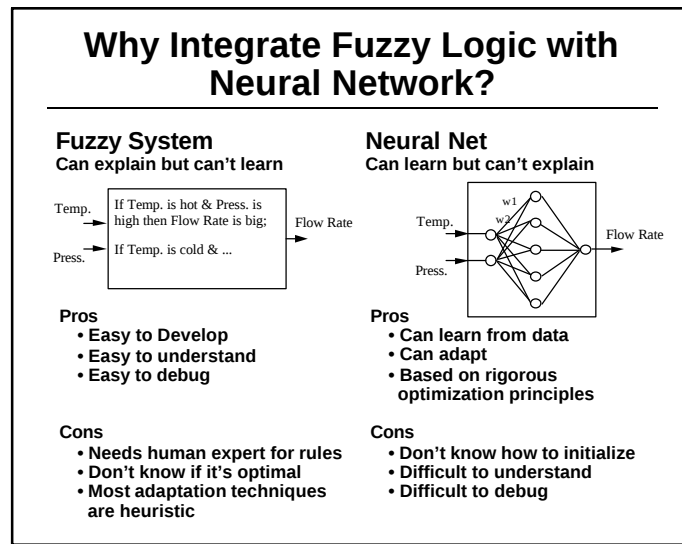
Here fuzzy rules are not being used to specify the familiar condition-action mapping, but to rank the different control outcomes in a way that reflects human's sensibility of "optimal". It would be difficult to solve this type of task-oriented control problem within an analytical framework.

Notable Points of Application Examples

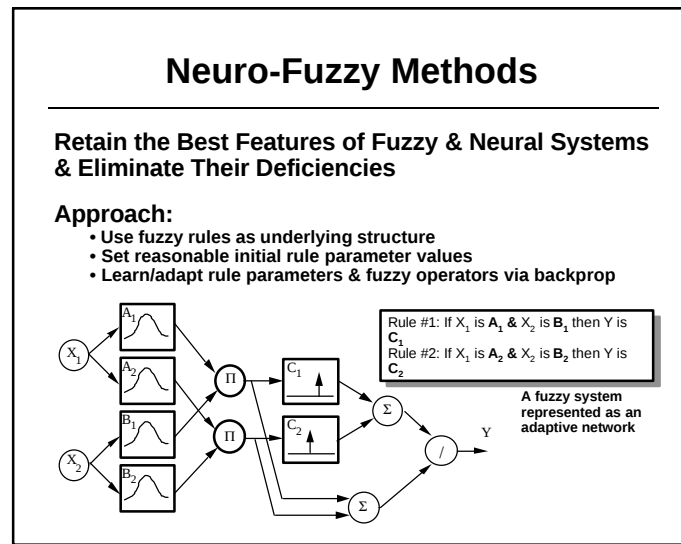
- Many different ways to use fuzzy logic for control applications
- Two classes of fuzzy controllers
 - Direct condition/action control
 - Ranking of discrete control options
- Two levels of fuzzy control
 - Low level, set-point oriented
 - High level, task oriented (biggest payoff)
- Commercial applications aimed at task-oriented control & more "human-friendly" machines
 - Increased convenience
 - Increased match with human intentions/objectives

We have been conditioned by education to have a narrow view of what constitutes a control problem (i.e., set-point regulation via state feedback), which engenders a narrow view of how fuzzy logic fits into the control big picture. The two previous example applications serve to illustrate several important points about fuzzy control, which are listed in the figure above. The main point is that fuzzy control is not just a form of nonlinear PID control; the number of ways that fuzzy logic can be used for control is only limited by your creativity. The essence of fuzzy logic is that **it lets you express what's on your mind**. It's no wonder that the biggest payoff of fuzzy control has been for high level, task-oriented control, where there are no standard solutions but ample human intuition.

In the commercial market, the motivation for applying fuzzy logic is not so much in improving the performance of a product, but to make a product easier to use. For both consumer and industrial applications, fuzzy logic is largely used to automate functions that had previously required the attention of a skilled human. To appreciate this business strategy, consider the commercial success of the auto-focus camera. Auto-focus cameras do not necessarily take better pictures than the traditional, fully mechanical cameras, but resources spent on developing auto-focus reaped much greater payoff than any incremental improvements on the performance of mechanical cameras. To derive maximum benefit from fuzzy control, we must be creative and look for new opportunities - a better control system is not simply one with a better state feedback law.

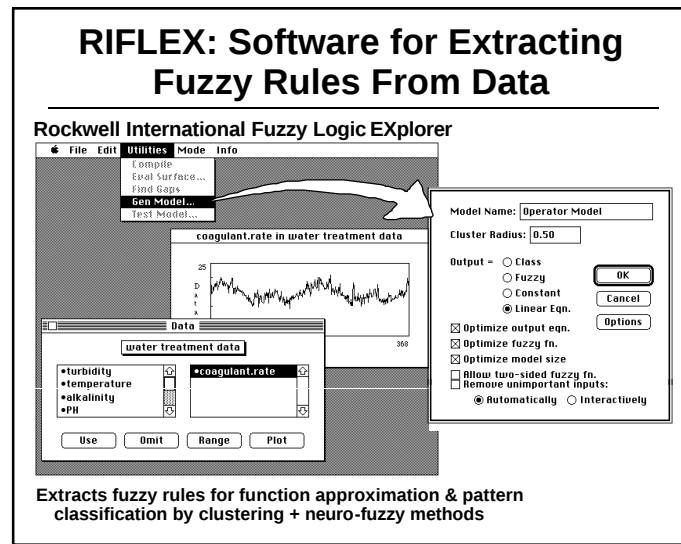


A fuzzy system is easy to develop, easy to understand, and easy to debug. However, traditionally a human expert is needed to specify the fuzzy rules, and the rules are fine tuned by trial and error. A neural network, on the other hand, uses well-grounded optimization methods to automatically learn from training data, but the resultant neuron connection weights have no physical meaning, thus making the neural network difficult to understand and debug. In short, a fuzzy system can explain the knowledge it encodes but can't learn or adapt its knowledge from training examples, while a neural network can learn from training examples but can't explain what it has learned. Because fuzzy systems and neural networks have complementary strengths and weaknesses, there is increasing interest in finding ways to integrate the two methodologies, to create hybrid systems that can learn from training examples as well as explain what has been learned.



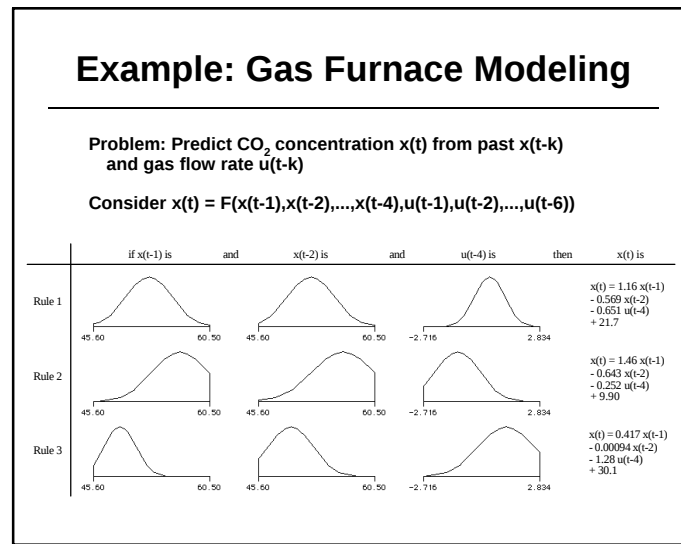
The integration of fuzzy logic with neural network techniques has resulted in what is commonly referred to as neuro-fuzzy systems. These systems use fuzzy rules as the underlying structure and then apply neural techniques to learn the rule parameters, e.g., the input region covered by each rule and the output value of each rule. A popular architecture is the ANFIS (Adaptive Network-based Fuzzy Inference System) architecture shown above, where fuzzy rules are represented as an equivalent adaptive network (a generalization of neural network); backpropagation can then be applied to learn the membership functions, output values, and parameterized logical operators (i.e., optimizing the “AND” operator). A few fuzzy system development software now incorporate neuro-fuzzy algorithms to help the user generate fuzzy rules from data. The ANFIS architecture, for example, is supported in the Fuzzy Logic Toolbox for MATLAB.

When considering the different techniques and how they fit together, we must keep in mind that the user is not interested in the techniques themselves, but in the fastest, most cost-effective way to achieve an end. In modeling, the greatest cost is in acquiring training data and the desired end is a simple model that can predict a system’s behavior accurately in the future. Model accuracy with respect to the available data is meaningless beyond a certain point because real data is invariably incomplete and noisy. Choosing the “right” modeling tool means matching the characteristics of the modeling method (e.g., ability to capture various nonlinearity, the number of fitting parameters needed to capture the nonlinearity, need for trial-and-error iterations) with the characteristics of the problem (e.g., the nonlinearity of the system, the amount of training data available/acquirable, amount of time you can spend on the problem).



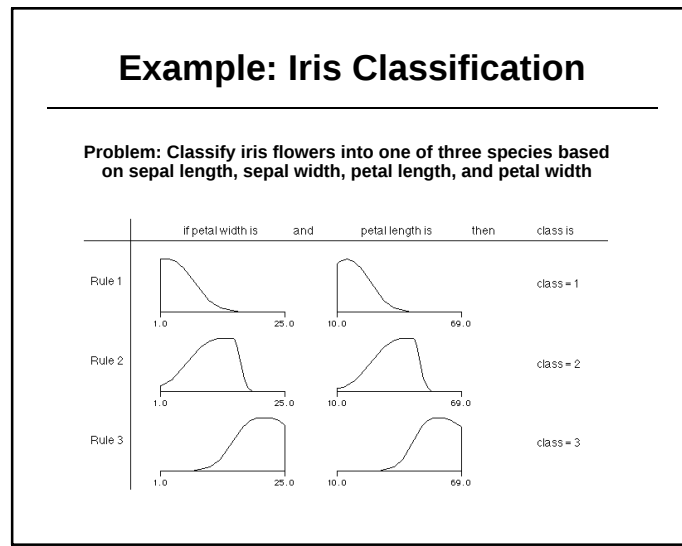
At Rockwell, we have developed a Macintosh software called RIFLEX that uses a combination of clustering and neuro-fuzzy techniques to automatically extract fuzzy rules from data. RIFLEX can extract fuzzy rules for both function approximation and pattern classification problems. The picture above shows the user interface. To extract rules, the user needs only to enter an initial cluster radius; the software uses clustering to determine the number of rules and initial rule parameter values, and then optimizes the rule parameters by backpropagation. The software can also automatically eliminate unimportant input variables by searching a tree of possible combinations of input variables. The key point here is that the user interface is very simple and gives no hint of the underlying relationships between clustering, fuzzy rules, neural networks, and tree search. To the user, the pieces are indistinguishable and appear as a single tool.

The indistinctness of the different methodologies is not just cosmetic, but real in many hybrid architectures. For example, it is difficult to categorize ANFIS as either a fuzzy system or a neural network (fortunately, we now have the new category called neuro-fuzzy system). The integration of fuzzy logic and neural network in ANFIS goes beyond having a separate fuzzy box that interacts with a separate neural box; it integrates fuzzy logic and neural network at a more fundamental level. Similarly, the distinction between existing methodologies will be blurred as one methodology borrows the best ideas from another to remedy its own weaknesses.



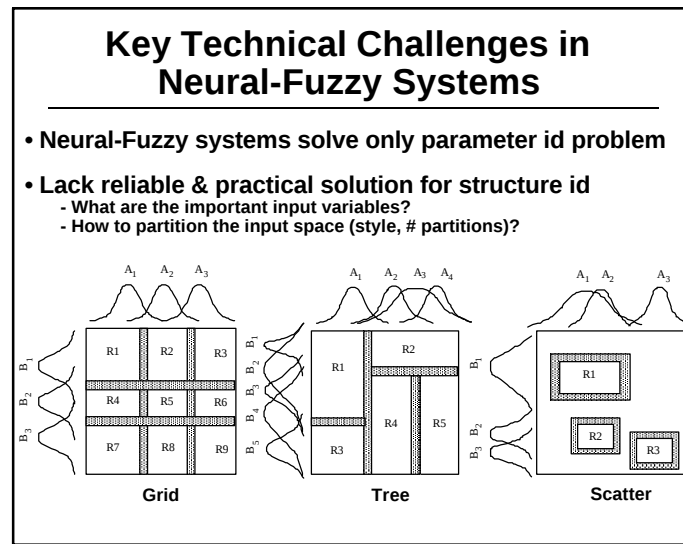
To illustrate the benefits of neural fuzzy systems, we consider a benchmark problem that involves modeling the dynamics of a gas furnace (the Box and Jenkins gas furnace data). The task is to predict the CO₂ concentration from past measurements of CO₂ concentration and gas flow rate. We consider 10 possible input variables: the CO₂ concentration at the past 4 sample times $\{x(t-1), x(t-2), \dots, x(t-4)\}$ and the gas flow rate at the past 6 sample times $\{u(t-1), u(t-2), \dots, u(t-6)\}$. The RIFLEX software determined that only 3 of these input variables are important and generated a model composed of only 3 rules. The rules are shown in the figure above. Here we have elected to use the Takagi-Sugeno type of rules where the consequent is a linear equation in the input variables; hence, an aspect of linear modeling was also involved. Because of the additional fitting parameters in the consequent, the Takagi-Sugeno type of rules can significantly increase the accuracy of a model when compared to the same number of conventional fuzzy rules.

The model tells us that the important input variables are $x(t-1)$, $x(t-2)$, and $u(t-4)$, which indicates the gas furnace dynamics is approximately second order with a dead time of 4 sample periods. From the rules, we also get an intuitive understanding of the relationships among system variables. The first rule tells us that medium levels of CO₂ concentration is associated with medium gas flow rate; the second rule tells us that high levels of CO₂ concentration is associated with low gas flow rate; and the third rule tells us that low levels of CO₂ concentration is associated with high gas flow rate. The rules also tell us that CO₂ concentrations do not change drastically over one sampling period, e.g., input combinations such as low $x(t-2)$ and high $x(t-1)$ do not exist.



Now we will look at a benchmark problem in pattern classification (Fisher's iris data). The task is to classify an iris flower into one of three species based on four input features: sepal length, sepal width, petal length, and petal width. The RIFLEX software determined that only petal width and petal length are important inputs and generated a classifier composed of 3 rules. The rules are shown in the figure above.

This classifier's performance is similar to those of pure neural-based classifiers reported in literature, but this classifier is much simpler and lets us understand how to classify iris flowers. The rules tell us that species #1 is marked by small sized petals, species #2 is marked by medium sized petals, and species #3 is marked by large sized petals. The rules also tell us that there are no species with disproportionate petal shapes, e.g., combinations such as long petal width and short petal length do not exist.



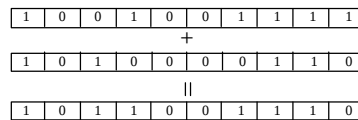
A neuro-fuzzy system by itself solves only the parameter identification problem, but do not address the structure identification problem (i.e., determine what input variables are useful, the number of rules, and how to partition the input space). In the RIFLEX software, for example, input variables are selected by tree search and the number of rules is determined by clustering (which necessarily leads to a scattered input partition). Other past methods have used tree search to simultaneously partition the input space and select input variables. Although these methods can provide reasonable solutions, there are better alternatives waiting to be explored. For example, tree search is necessarily heuristic for practical applications and can not guarantee finding the optimal input and partition combination. Searching by genetic algorithm is computationally more demanding than tree search, but genetic algorithm may provide the best trade-off between computational tractability and increased likelihood of finding the optimal solution.

Using Genetic Algorithm to Solve Structure ID Problems

- Encode possible input variable combinations & partitions as “chromosomes”



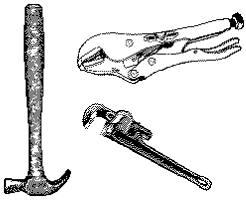
- Generate a population of solutions
- Select good performers for further “breeding”



Genetic algorithm can be used for structure identification by encoding possible input variable and partition combinations as chromosome strings. In the figure above, possible combinations of 10 input variables are encoded as chromosome strings, where a 1 represents the presence of an input variable and a 0 represents absence. Genetic algorithm generates a population of different solutions and retains the best solutions to further “breed” a new population containing better solutions. Because genetic algorithm searches along multiple paths simultaneously and introduces some randomness into the search, the search process is unlikely to paint itself into a bad corner. However, search by genetic algorithm is much more computationally demanding than conventional tree search because more potential solutions need to be evaluated. The choice of search algorithm should depend on the degree of difficulty in evaluating a potential solution, the importance of finding the best solution versus just a good solution, how well the selection criterion reflects real needs, the computational resource, and how much time can be spent on the problem.

Summary

- **Complex problems must be attacked by using a collection of different tools**
- **Matching the right part of the problem w/ the right technique results in most effective solution**



In today's competitive economic environment, solving engineering problems requires a careful balance between the quality of the solution, time-to-market, and the cost of the solution. To solve problems in the most effective way, an engineer must be aware of the different tools available, understand the strengths and weaknesses of each tool, and apply the right tool to the right job.

The "right job" for a particular tool is typically a narrow subtask within a broader problem; hence, each tool is often just a small piece of the overall solution. Using the different tools in combination is not an academic fancy, but a necessity to achieve an optimal overall solution. Using the tools in combination can involve attacking each subtask with the most appropriate tool (e.g., linear analysis, fuzzy logic, neural network, genetic algorithm), or creating a more effective hybrid tool such as neuro-fuzzy systems.

There is no good tool or bad tool, but there are definitely appropriate and inappropriate uses for a tool. We can tell that a tool is inappropriate for a job when using the tool for that job is clumsy. If all existing tools are clumsy for an important job, then chances are we need to invent a new tool.