HyperLogic
TECHNICAL NOTES
A series of informal papers on technical topics for customers and friends of HyperLogic Corporation
Number 8
Revision 2

Is This a Fuzzy Problem?

This note is to help you decide whether a fuzzy system would be effective in a problem you face.

"Fuzzy" has become quite a broad term. It is used to describe techniques as diverse as "sound-alike" text matching, fuzzy arithmetic, cognitive maps, and many others. This document concentrates specifically on fuzzy rule-based systems and the inferencing methods used in HyperLogic's CubiCalc fuzzy shell. See our Tech Note #1 for detail about CubiCalc's inferencing methods.

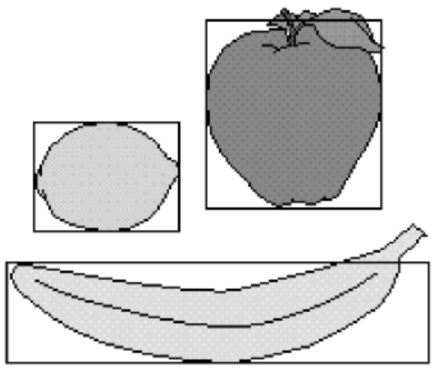
Is the problem numeric?

Fuzzy rule-based systems are computational elements. They are not symbolic reasoning components. The fuzzy system takes numbers as inputs and produces numbers as outputs. If your problem involves text or other symbolic data, you'll have to represent it as a numeric problem to apply fuzzy rules.

Here is an example from the PICKER project used as a tutorial in the CubiCalc "working model" demo. Given a description in terms of *weight*, *shape*, and *color*, PICKER chooses which of several fruits is being described.

Weight is clearly numeric. *Color* representation in numeric terms often needs multiple dimensions. But the *shape* input is a good example for discussion.

You might think of shape in terms of words like "oval" or "round," but those aren't suitable inputs for a fuzzy rule-based system. Aspect ratio (the ratio of width to height) is a single numeric description of "roundness." Then the descriptive adjectives *thin*, *oval*, and *round* become the fuzzy sets describing the aspect ratio.


Aspect Ratio

So the inputs to the fuzzy system are not words but numbers. "What about distributions?" you might ask. "I have a distribution that defines 'tall.' Can't I use that as an input? If so, isn't that the same as using a word as an input?"

There *is* a conceptually simple approach that lets you use a distribution as input to a fuzzy rule-based system. Assuming sufficient regularity conditions, if F is the fuzzy system and p is the distribution (technically, a probability density), the response of F to the distribution p is:

$$\frac{\int F(x)\, p(x)\, dx}{\int p(x)\, dx}\ .$$

In practice, you would probably just compute a weighted average instead. Nonetheless, a fuzzy system always defines a numeric function.

Now suppose you wanted to take as input any one of a collection of words, with each word defined by a distribution. You could use the technique just described to apply your fuzzy rule base to "concept" inputs instead of numeric inputs.

That you *can* does not imply that you *should*! A simplified example illustrates: Suppose you have predefined distributions for heights called *tall*, *medium*, and *short* and your rules relate these in a natural way to shoe sizes called *large*, *medium*, and *small*. If those three specific distributions are the only possible inputs to the system, then there are only three possible results!

Your own problem is certainly not so simple, but it may be similar. So if your inputs are words from a predefined list, consider a simple lookup table instead of fuzzy rules.

On the other hand, if you will continually produce new distributions — so the results can be different every time — then you no longer have a fixed predefined set of input cases. When arbitrary distributions are possible the integral method is useful. But in that case, you probably don't have *names* for all the distributions; your input is a shape, not a word.

Is the input-output relationship vague but understandable? Or is it the statement of the problem that's unclear?

Fuzzy rule-based systems use English-like rules to say how the desired output relates to the input. Fuzzy rules use indefinite terms, like "large" or "warm" or "fast" so you don't need to write an explicit formula — you just need to describe general relationships.

To use a fuzzy rule-based system, either you must understand the problem well enough to write the rules or you need to have sufficient data examples of the input-output relationship. In any case, you *must* be able to identify specific numeric input and output variables.

There is no AI or machine intelligence method known today that will perform a system analysis for you. If the problem is completely nebulous, maybe what you need is an initial partitioning of the problem into manageable pieces. If you don't know how to do this, perhaps you should look for outside help. Some consultants specialize in this type of analysis.

But if you understand and can describe the general input-output relationship, try a fuzzy approach.

Are there gradual transitions between different cases?

The hallmark of a fuzzy system is a smooth transition between the cases governed by different rules. A "speed" input in a fuzzy system doesn't have a specific dividing line between "slow" and "fast" — it gradually changes from one to the other. For certain speeds, both terms apply to a degree.

Consequently, fuzzy sets serving as descriptive adjectives for input variables almost always overlap. Wherever overlap appears, the situation is "somewhat this and somewhat that." Multiple rules apply in the overlap.

Is an approximate solution acceptable?

People communicate quite well with approximate language: "A little higher," instead of "up by 2.73 centimeters." In many day-to-day situations we instruct others in approximate terms: maneuvering vehicles, cooking, taking photographs, adjusting all kinds of equipment.

Some automation problems don't need an exact solution — "close enough" is good enough. For this type of problem, a continuous fuzzy system lets you *describe* the solution instead of *specifying* it.

The fuzzy truck backer is an excellent example with a simple approximate solution. The goal is to back a truck into a loading dock by turning the steering wheel. To lay out a fuzzy solution, just consider representative situations. For each one, decide which direction to turn and whether it's a small turn or a large turn. Write these instructions as a list of rules. Another approach is to mathematically solve the problem. Try it both ways and decide for yourself.

If you prefer an approximate solution today over an exact solution next year or never, try a fuzzy approach.

But even if you know exactly what the numeric results should be in every situation, consider *how* you know. The next two topics show why it matters.

Do you have an equation but not enough horsepower to evaluate it?

If you already know the answer for every case because you have a formula, one reason to use a fuzzy method is for more efficient computation or storage.

Microcontrollers are good at integer operations and not so good at floating point. Many equations are not amenable to 8-bit or integer arithmetic. For that case, try a fuzzy rule-based system as a function approximator.

Fit fuzzy sets to the peaks and valleys of your function and let the fuzzy system interpolate between exact points identified by rules. You can do all the tuning at your desk instead of in the lab. You can make trade-offs between speed, space, and precision without ever changing your code by hand. (Read our Tech Note #4 on embedded systems to see how.)

Is there a table that gives the right answers but there's not enough storage space to hold it?

If the input-output relationships are embodied in a collection of data, but you can't carry the data along in the target system, maybe a function approximation is what you need.

If you have a database in which the answer appears for each possible input, and that database is both available and rapidly accessible, then when you need an answer you can just look it up in the table. But if the table is too big, a fuzzy system can be a good substitute.

Recall that trigonometric and logarithmic tables were once quite common, but portable calculators made computation sufficiently accurate and inexpensive that the tables were no longer needed. A fuzzy rule-based system is a good way to approximate what's in *your* table. (This type of problem is also a good candidate for a neural network, another function approximation technique.)

Whether your table can be compressed into a fuzzy system depends very much on the nature of the function it represents. You might have to use multiple fuzzy systems, deciding which to use with Boolean decisions based on one of the inputs. With a large number of inputs, you might need to partition the table to get the fuzzy system down to a reasonable size. You might want to use data analysis tools and automatic rule-building methods, rather than describing the input-output relationship with your own rules.

In Brief ...

Fuzzy logic can solve problems that are numeric and approximate, with gradual transitions between cases. The problem can be vaguely described, but must still be clearly understood. Even if an exact solution is available, a fuzzy implementation is sometimes more economical.

Still not sure?

With HyperLogic's CubiCalc, you can create a fuzzy rule-based system and use it interactively. When you're finished reading this note, try out the CubiCalc "working model" demo to make your own rules and see how they work. The demo comes with several examples that illustrate the behavior of fuzzy rule-based systems.

By working through your own examples, you can get a much better feel for the behavior of fuzzy systems and when to use them.