HyperLogic

TECHNICAL NOTES

A series of informal papers on technical topics for customers and friends of HyperLogic Corporation

Number 1
Revision 5

Fuzzy Logic Inference Techniques in CubiCalc Version 2

Introduction

This note assumes you have some basic familiarity with fuzzy logic, but not much. It assumes you've heard of fuzzy logic and you know that it's a computing technique that somehow accommodates the ambiguity of everyday language. You've heard about fuzzy sets or membership functions (for our purposes synonymous) and you've seen graphs illustrating, for example, how a fuzzy air conditioner reasons in terms of words like warm, hot, and cold.

The purpose of this technical note is to explain the steps involved in performing fuzzy inferences from fuzzy rules and to describe options available in CubiCalc for performing the various steps.

Terminology

Suppose you have a set of fuzzy rules for selecting a computer as shown in Figure 1.

The fuzzy input variables in the rules are *price*, *speed*, and *reliability*. The single fuzzy output variable is *choice*. Each variable has a range, sometimes referred to as its *universe of discourse*.

The IF part of a rule is its *antecedent* and the THEN part is its *consequent*. Fuzzy input variables always appear in rule antecedents. A rule consequent refers to one or more fuzzy output variables.

The words like "typical,“ "low," and "good" are *adjectives* describing the fuzzy variables. You define an adjective by specifying a function that gives the degree to which each value of the variable is described by the adjective. These functions are also called *membership functions* because they represent degrees of membership in *fuzzy sets*.

In Boolean logic, something is either a member of a set or not, with nothing in between. A fuzzy set allows partial membership. A clock rate of 25 MHz might be a member of the set of low computer speeds to some degree and simultaneously a partial member of the set of high speeds.

Fuzzy set memberships range from zero to one, with zero representing no membership and one representing full membership. High membership means the adjective is a good description of the value; low membership means it is not.

The word "very" is an *adverb*, a modifier for an adjective. These words are also referred to as *hedges* in the technical literature.

```
If price is typical then choice is okay;
If speed is low or price is high then choice is bad;
If reliability is good or
  (reliability is okay and warranty is very long)
   then choice is good;
```

Figure 1. Example Rules

Evaluating the Antecedents

The first step in evaluating a set of fuzzy rules is to calculate the antecedents. For each rule, the *activation level* of the IF part is computed; this is the degree to which the rule is active. Computation of the fuzzy outputs uses the rule activation level in a way determined by the particular inference method.

The simplest antecedent is of the form "*var* is *adj*" where *var* is a variable name and *adj* is an adjective that applies to the variable *var*. To evaluate this antecedent, take the value of the membership function at the current value of *var*.

For example, suppose Figure 2 defines the word "typical" applied to the variable "price." The graph shows the membership (on the vertical axis) of each price under discussion (on the horizontal axis). If price is 1500, it is typical to a high degree; if price is zero or 3000, it is not typical at all.

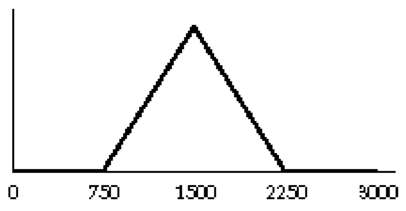

Figure 2. "Typical" Prices

A slightly more complex antecedent could use an adverb to modify the adjective. The phrase "warranty is very long" is an example of this.

When an adverb modifies an adjective, its effect is to change the shape of the membership function. The word *very* steepens the rise of the curve by squaring the original adjective function; *somewhat* softens the rise by using the square root.

If a warranty is very long, it is certainly also "long." So it seems reasonable that memberships in the fuzzy set "long" would be at least as great as those in "very long."

Another common modifier is *not*, which complements the membership. The graph of *not A* is the complement of the graph of *A*. An important feature of fuzzy logic is that both A and not A can be partially true simultaneously, unlike Boolean logic where complements are mutually exclusive.

Thus if 1450 is “typical” to degree 0.93, it is “very typical” to a lesser degree (0.87) but “somewhat typical” to a greater degree (0.96). The degree to which 1450 is “not typical” is 0.07, obtained by subtracting 0.93 from 1.0. Figure 3 illustrates the effects of *very*, *somewhat*, and *not* on a triangular fuzzy set graph.

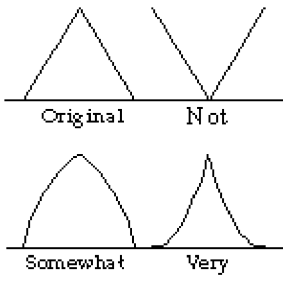

Figure 3. Adverbs

Even more complex antecedents can be created using AND and OR. If the antecedent of a rule connects two simple antecedents with AND, the two are combined by taking the minimum of the two. If OR connects them, the result of two simple antecedents is the maximum of the two.

A	B	A OR B max(A,B)	A AND B min(A,B)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1
0.9	0.1	0.9	0.1

Figure 4. Logical Operators

These operations are consistent with Boolean logic, as illustrated by the table in Figure 4.

The minimum and maximum operators are limiting cases of more general operators called *t-norms* and *t-conorms*, or just “conorms.” Another pair that has been used effectively is multiplication and its conorm ($x + y - xy$).

CubiCalc observes standard rules of operator precedence for AND and OR. Parenthesized phrases are evaluated first, but if no parentheses are present, AND is given higher precedence than OR. This means that “If C is medium or A is small and B is large” is the same as “If C is medium or (A is small and B is large).”

To evaluate an antecedent involving AND or OR, CubiCalc first evaluates the constituent simple antecedent phrases, applies adverbs, then combines antecedents with maximum and minimum in an order determined by parentheses and the operator precedence rules. Figure 5 shows intermediate values of antecedent computations for a sample rule.

Once the antecedents have been evaluated, inference options determine the methods used for the remaining steps: scaling the consequent adjectives, combining the scaled consequent adjectives to create a resultant adjective, and defuzzifying the combined resultant to obtain a single numeric result. CubiCalc lets you choose how to do each of these remaining steps.

Scaling the Consequents

With CubiCalc you can choose whether to scale the consequent adjectives by multiplying pointwise by the activation level or by taking the pointwise minimum of the adjective and the activation level.

The minimum method has been used in many fuzzy systems, but there are computational and analytic advantages to scaling by product instead. Multiplying a function by a positive value does not change the horizontal positions of its center of gravity and its region of maximum height. Figure 6 illustrates these two methods.

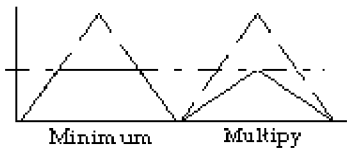

Figure 6. Scaling Consequents

Combining Consequents

After scaling, CubiCalc gives three choices for combination of consequent adjectives: You can add them, take the pointwise maximum over all of them, or select the single best as the resultant, ignoring the others. The “best” is the one most active, the one appearing in the rule with the highest activation. Figure 7 shows the effects of these actions.

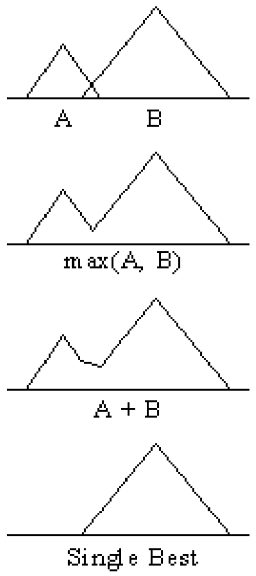

Figure 7. Consequent Combination

Reliability is good	Reliability is okay	Warranty is long	Warranty is very long	Reliability is okay and Warranty is very long	Reliability is good or (Reliability is okay and Warranty is very long)
0.5	0.5	0.5	0.25	0.25	0.5
0.75	0.5	0.5	0.25	0.25	0.75
0.5	.9	0.9	0.81	0.81	0.81
0.5	0.9	1.0	1.0	0.9	0.9

Figure 5. Evaluating Antecedents

CubiCalc offers another option for the combination step when you have more than one rule with the same consequent: "If A is large then C is small" and "If B is small then C is small."

One method is to treat these rules as if all the antecedents had been connected with OR in a single rule. In this case, the maximum activation over all the rules for that consequent is used. This is the same as writing one rule, "If A is large or B is small then C is small."

The other method is to add the activations of rules with the same consequent. This has the effect of putting more emphasis on any consequent mentioned in multiple rules.

Defuzzification

The resultant fuzzy set computed by scaling and combining output adjectives could be described as a membership function for the degree to which each value is "the answer." What remains is to *defuzzify* — to pick a single value as the final result from the rule base. CubiCalc provides several ways to do this: centroid and three variations of maximum height.

The centroid is the "center of gravity" of the resultant shape. It is the point at which, if solid, it would balance. This takes into account the entire shape. Compare the centroid results from the shapes in Figure 8 to see how this works.

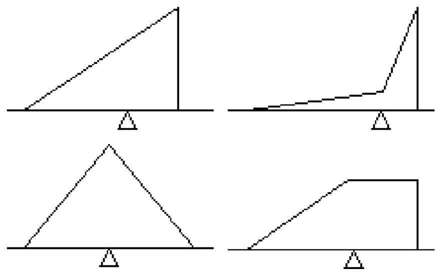

Figure 8. Centroids

Max-height defuzzification uses the *x*-coordinate of the point of maximum height as the result. But since the maximum height can be achieved at multiple points or even over regions, a tie-breaking method is needed. With CubiCalc, you can choose to take either the left-most or right-most point of maximum height or to use the point midway between these two.

Figure 9 repeats the shapes from Figure 8, with the midpoint of the region of maximum height indicated. For all but the last figure, the left and right maximum are the same as the midpoint.

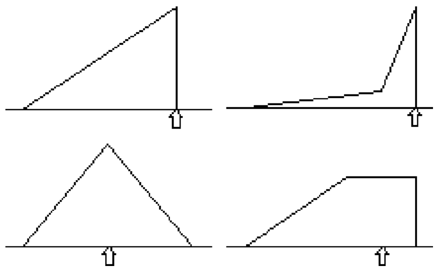

Figure 9. Midpoint of Maximum

Centroid defuzzification tends to give smoother results than max-height methods. The point or region of maximum height can change quite suddenly as different rules activate, as illustrated by Figure 10. The left triangle is slightly higher than the right triangle in the first example, but slightly lower in the second. Note that the centroids of the two shapes are very close, but the points of maximum height are widely separated. This can result in choppy response as the inputs vary slightly.

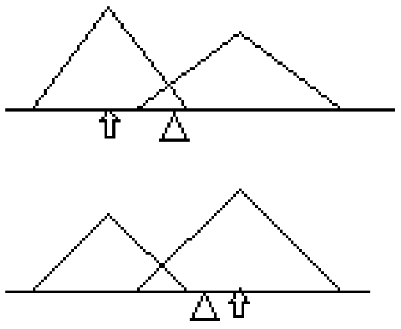

Figure 10. Max-height vs. Centroid

For product scaling and sum combination, the centroid method is very efficient due to computational shortcuts available. But for applications in which the problem is to select one from among a set of discrete choices, the left or right maximum with product scaling is generally best. In discrete decision problems, the "choppiness" of max-height defuzzification is precisely what is needed — a result that is halfway between two discrete options is unacceptable.

Run-time Options

CubiCalc RTC provides additional options for evaluating fuzzy rules. These are more a matter of implementation method than actual inference techniques, but they can affect the final results.

With CubiCalc RTC's run-time generator, you can choose whether to execute the fuzzy rule evaluation using integer math or floating point. Integer is usually faster and smaller but less precise than floating point. Eight-bit integer math is used in the precompiled object code libraries and the Dynamic Link Library for Windows, but the run-time source code supports other integer types.

CubiCalc RTC also lets you choose whether integer input memberships are evaluated using a fast lookup table of samples or by interpolating from a set of control points that define the adjective shape. When tables are used, you have control over the size, hence the precision, of the table.

Another option in CubiCalc RTC is to use singleton values to represent output adjectives. This is meaningful for only a few of the inference methods but can produce smaller data structures with slightly faster execution.

Effects of Inference Methods

First-time users of fuzzy logic are sometimes surprised by the results of their rules. By understanding the computation process, it is easier to see what is causing the effects. In this section we cover a few issues that might not be obvious at first glance.

First we'll set up a test bed for CubiCalc experiments as follows: Define a fuzzy input variable called X with range 0 to 100. Make a single membership function called "large" that is a straight line from (0, 0) to (100, 1). Clone X to create Y, a fuzzy output variable. Write a single rule, "If X is large then Y is large;" Create a scatter chart of X versus Y. Then enter the following instructions in Preprocessing:

```
IF (X < 100) X = X + 1;
ELSE X = 0;
END
```

As the scenario runs, X steps through its range. When the range is exceeded, it starts over again at the lower limit of its range.

Select product-sum-centroid inference (*i.e.* scale by product, combine by sum, defuzzify by centroid) and run the project.

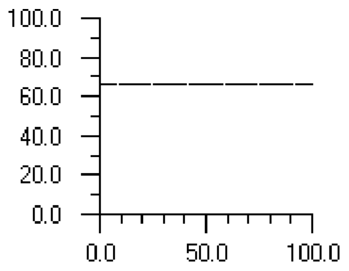

Figure 12. Constant Output

The result is constant! As it turns out, the centroid does not change when a membership function is scaled by multiplication. This can be seen by examining the form of the equation for the centroid, shown in Figure 13. If both the numerator and denominator are multiplied by the same constant (the rule activation level), the result does not change.

$$\frac{\int y\, f(y)\, dy}{\int f(y)\, dy}$$

Figure 13. Centroid Formula

As a result, if only a single rule is active, the output resultant membership function is just a scaled version of the named consequent membership function. Its centroid is always the same, regardless of the activation level (unless the activation is zero.) Thus with product-sum-centroid inference, your rules must cause multiple consequents to be active if you want to avoid regions of constant output. Using minimum scaling avoids this, but only if the output adjectives are asymmetric.

Now change the scaling to minimum instead of product and run again. The result looks like Figure 14. Although this is a little more satisfying, no practical fuzzy system uses only one rule. But when there are regions in which only a single rule is active it's good to know about this effect.

Another situation that can arise in product-sum-centroid inference is that effects of same-size or symmetric membership functions can cancel out. A careful study of the inference process shows how this can happen but it's a little too complicated to explain in this short article.

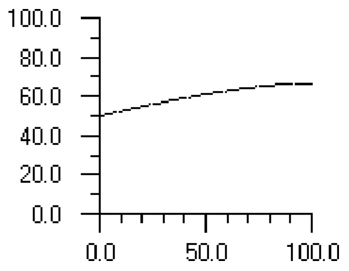

Figure 14. Scale by Minimum

To see the effect, use the Adjective Editor to automatically create five evenly-spaced membership functions for X, each with a base width of 50. (The outer two adjectives will be only half the width of the inner ones.) Discard Y and clone X to create a new version of Y. Write five rules like "If X is Small then Y is Small;"

When you run the scenario, the initial plot looks like Figure 15. Notice the linear region in the center, from X = 25 to X = 75.

Removing symmetries on the input side can eliminate the linearity, as can changing the output adjectives so they do not all have the same area. Figure 16 incorporates both changes. It uses three adjectives instead of five.

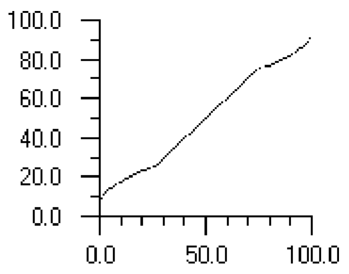

Figure 15. Linear Region 25 to 75

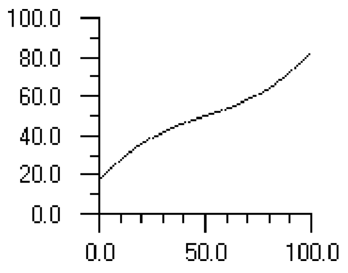

Figure 16. Linearities Removed

Another issue that arises with centroid defuzzification is that the centroid itself is undefined when the denominator is zero. This occurs when the resultant fuzzy set obtained by scaling and combining various consequents has a zero integral. Since the membership functions are all positive, the only way this can happen is if no rules are active at all. When you use centroid defuzzification, you must be prepared for a default action in case no rule is active — the standard computations just don't go through.

The issues for maximum-height defuzzification methods are different — perhaps a little more obvious but not so easy to accommodate. The points of maximum height can change drastically with small input changes. With max-height defuzzification, exhaustive testing is a good idea.

Summary

Fuzzy methods can require some experience to learn how to achieve desired effects. The technique described above is a good way to get a feel for them with CubiCalc. As the experimental project runs, you can make changes and see immediate effects in the plots you've set up. This type of experimentation is really the best way to see how fuzzy logic works.