Fuzzy Logic Toolbox

Provide feedback about this page

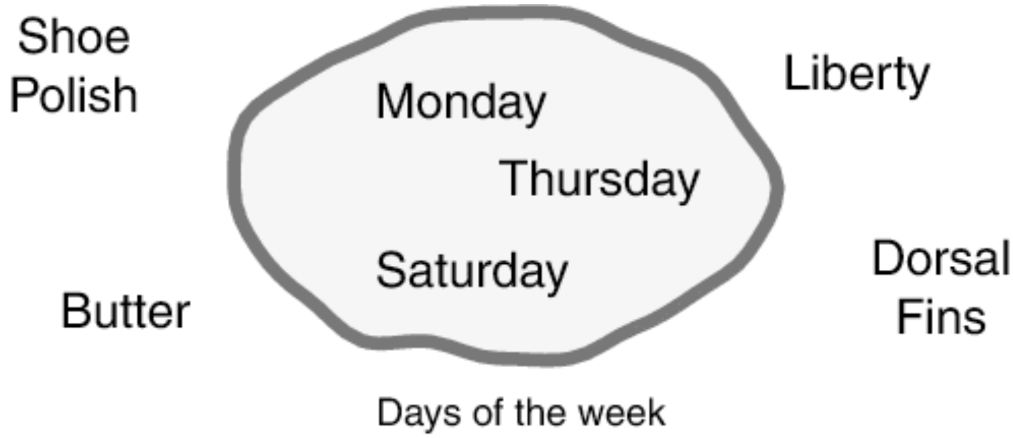
Foundations of Fuzzy Logic

On this page…

Fuzzy Sets

Fuzzy logic starts with the concept of a fuzzy set. A *fuzzy set* is a set without a crisp, clearly defined boundary. It can contain elements with only a partial degree of membership.

To understand what a fuzzy set is, first consider the definition of a *classical set*. A classical set is a container that wholly includes or wholly excludes any given element. For example, the set of days of the week unquestionably includes Monday, Thursday, and Saturday. It just as unquestionably excludes butter, liberty, and dorsal fins, and so on.

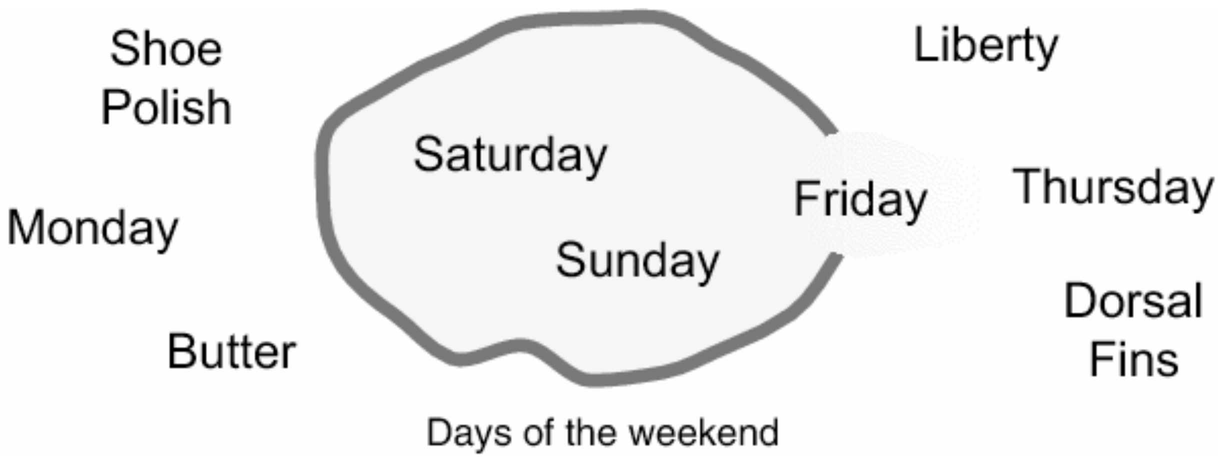


This type of set is called a classical set because it has been around for a long time. It was Aristotle who first formulated the Law of the Excluded Middle, which says X must either be in set A or in set not-A. Another version of this law is:

Of any subject, one thing must be either asserted or denied.

To restate this law with annotations: "Of any subject (say Monday), one thing (a day of the week) must be either asserted or denied (I assert that Monday is a day of the week)." This law demands that opposites, the two categories A and not-A, should between them contain the entire universe. Everything falls into either one group or the other. There is no thing that is both a day of the week and not a day of the week.

Now, consider the set of days comprising a weekend. The following diagram attempts to classify the weekend days.



Most would agree that Saturday and Sunday belong, but what about Friday? It feels like a part of the weekend, but somehow it seems like it should be technically excluded. Thus, in the preceding diagram, Friday tries its best to "straddle on the fence."

Classical or normal sets would not tolerate this kind of classification. Either something is in or it is out. Human experience suggests something different, however, straddling the fence is part of life.

Of course individual perceptions and cultural background must be taken into account when you define what constitutes the weekend. Even the dictionary is imprecise, defining the weekend as the period from Friday night or Saturday to Monday morning. You are entering the realm where sharp-edged, yes-no logic stops being helpful. Fuzzy reasoning becomes valuable exactly when you work with how people really perceive the concept *weekend* as opposed to a simple-minded classification useful for accounting purposes only. More than anything else, the following statement lays the foundations for fuzzy logic.

In fuzzy logic, the truth of any statement becomes a matter of degree.

Any statement can be fuzzy. The major advantage that fuzzy reasoning offers is the ability to reply to a yes-no question with a not-quite-yes-or-no answer. Humans do this kind of thing all the time (think how rarely you get a straight answer to a seemingly simple question), but it is a rather new trick for computers.

How does it work? Reasoning in fuzzy logic is just a matter of generalizing the familiar yes-no (Boolean) logic. If you give true the numerical value of 1 and false the numerical value of 0, this value indicates that fuzzy logic also permits in-between values like 0.2 and 0.7453. For instance:

Q: Is Saturday a weekend day?

A: 1 (yes, or true)

Q: Is Tuesday a weekend day?

A: 0 (no, or false)

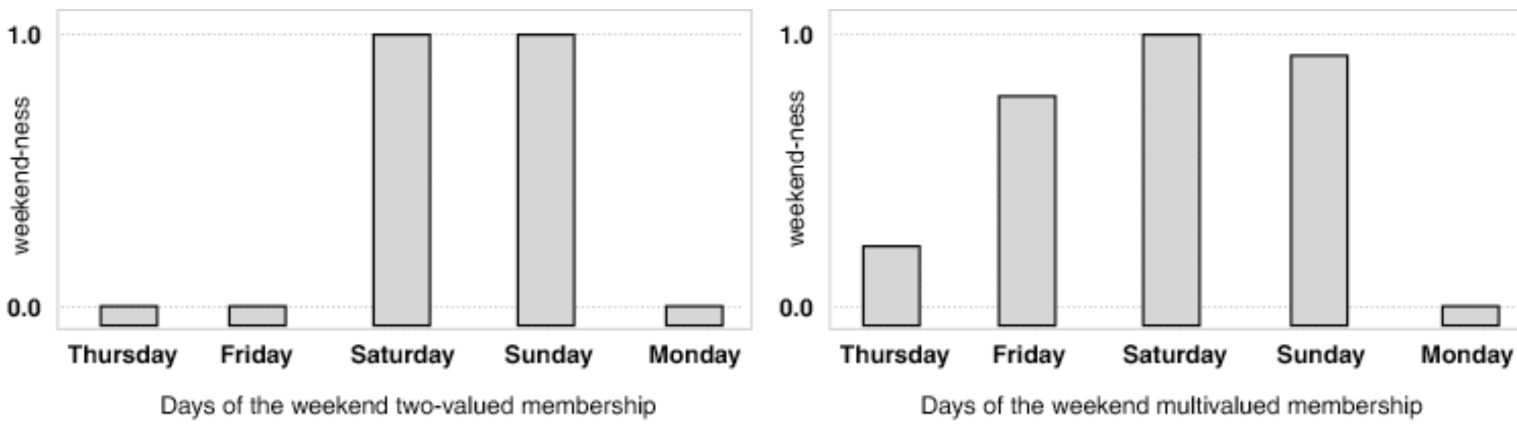
Q: Is Friday a weekend day?

A: 0.8 (for the most part yes, but not completely)

Q: Is Sunday a weekend day?

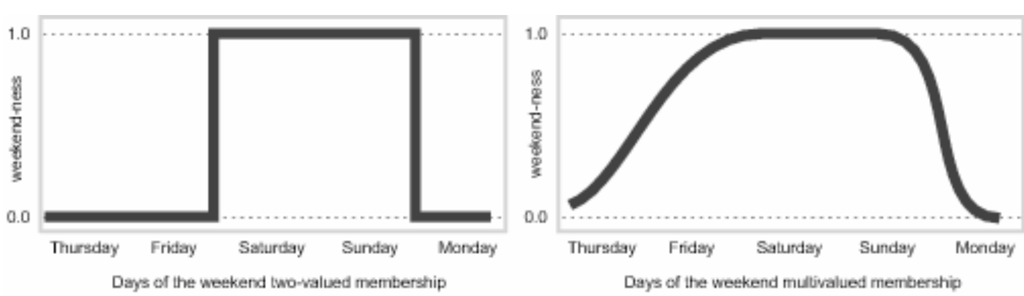
A: 0.95 (yes, but not quite as much as Saturday).

The following plot on the left shows the truth values for weekend-ness if you are forced to respond with an absolute yes or no response. On the right, is a plot that shows the truth value for weekend-ness if you are allowed to respond with fuzzy in-between values.



Technically, the representation on the right is from the domain of *multivalued logic* (or multivalent logic). If you ask the question "Is X a member of set A?" the answer might be yes, no, or any one of a thousand intermediate values in between. Thus, X might have partial membership in A. Multivalued logic stands in direct contrast to the more familiar concept of two-valued (or bivalent yes-no) logic.

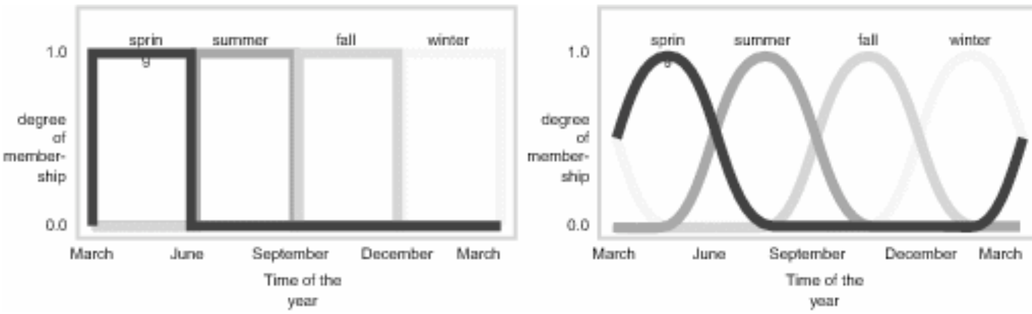
To return to the example, now consider a continuous scale time plot of weekend-ness shown in the following plots.



By making the plot continuous, you are defining the degree to which any given instant belongs in the weekend rather than an entire day. In the plot on the left, notice that at midnight on Friday, just as the second hand sweeps past 12, the weekend-ness truth value jumps discontinuously from 0 to 1. This is one way to define the weekend, and while it may be useful to an accountant, it may not really connect with your own real-world experience of weekend-ness.

The plot on the right shows a smoothly varying curve that accounts for the fact that all of Friday, and, to a small degree, parts of Thursday, partake of the quality of weekend-ness and thus deserve partial membership in the fuzzy set of weekend moments. The curve that defines the weekend-ness of any instant in time is a function that maps the input space (time of the week) to the output space (weekend-ness). Specifically it is known as a *membership function*. See Membership Functions for a more detailed discussion.

As another example of fuzzy sets, consider the question of seasons. What season is it right now? In the northern hemisphere, summer officially begins at the exact moment in the earth's orbit when the North Pole is pointed most directly toward the sun. It occurs exactly once a year, in late June. Using the astronomical definitions for the season, you get sharp boundaries as shown on the left in the figure that follows. But what you experience as the seasons vary more or less continuously as shown on the right in the following figure (in temperate northern hemisphere climates).

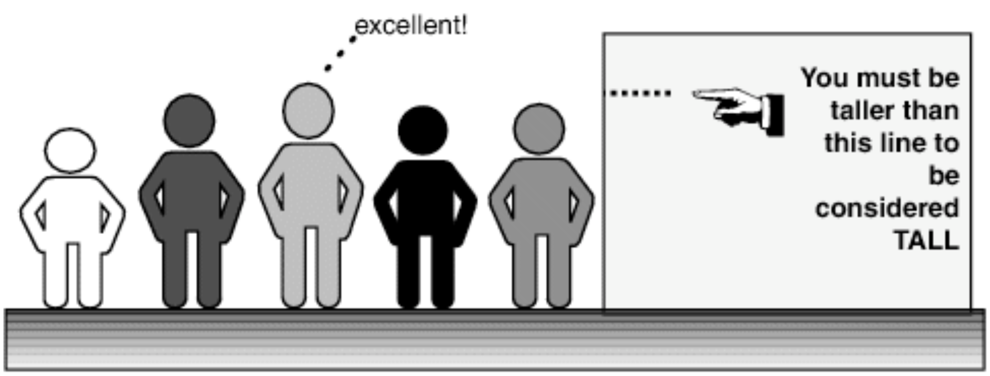


Back to Top

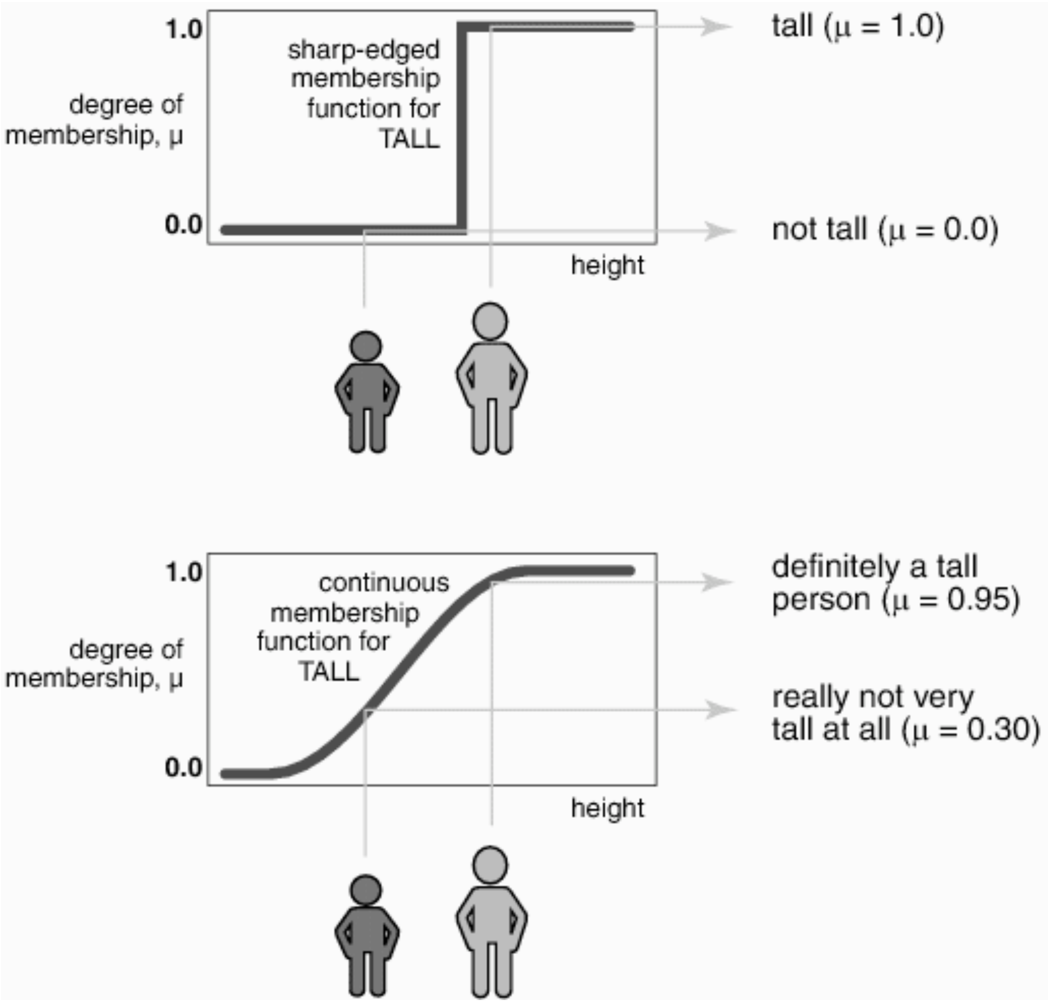
Membership Functions

A *membership function* (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The input space is sometimes referred to as the *universe of discourse*, a fancy name for a simple concept.

One of the most commonly used examples of a fuzzy set is the set of tall people. In this case, the universe of discourse is all potential heights, say from 3 feet to 9 feet, and the word tall would correspond to a curve that defines the degree to which any person is tall. If the set of tall people is given the well-defined (crisp) boundary of a classical set, you might say all people taller than 6 feet are officially considered tall. However, such a distinction is clearly absurd. It may make sense to consider the set of all real numbers greater than 6 because numbers belong on an abstract plane, but when we want to talk about real people, it is unreasonable to call one person short and another one tall when they differ in height by the width of a hair.



If the kind of distinction shown previously is unworkable, then what is the right way to define the set of tall people? Much as with the plot of weekend days, the figure following shows a smoothly varying curve that passes from not-tall to tall. The output-axis is a number known as the membership value between 0 and 1. The curve is known as a *membership function* and is often given the designation of μ. This curve defines the transition from not tall to tall. Both people are tall to some degree, but one is significantly less tall than the other.



Subjective interpretations and appropriate units are built right into fuzzy sets. If you say "She's tall," the membership function tall should already take into account whether you are referring to a six-year-old or a grown woman. Similarly, the units are included in the curve. Certainly it makes no sense to say "Is she tall in inches or in meters?"

Membership Functions in Fuzzy Logic Toolbox

The only condition a membership function must really satisfy is that it must vary between 0 and 1. The function itself can be an arbitrary curve whose shape we can define as a function that suits us from the point of view of simplicity, convenience, speed, and efficiency.

A classical set might be expressed as

$$A = \{x \mid x > 6\}$$

A fuzzy set is an extension of a classical set. If X is the universe of discourse and its elements are denoted by *x*, then a fuzzy set A in X is defined as a set of ordered pairs.

$$A = \{x, \mu_A(x) \mid x \in X\}$$

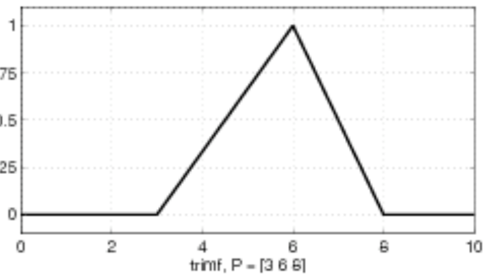
$\mu_A(x)$ is called the membership function (or MF) of *x* in A. The membership function maps each element of X to a membership value between 0 and 1.

Fuzzy Logic Toolbox includes 11 built-in membership function types. These 11 functions are, in turn, built from several basic functions:

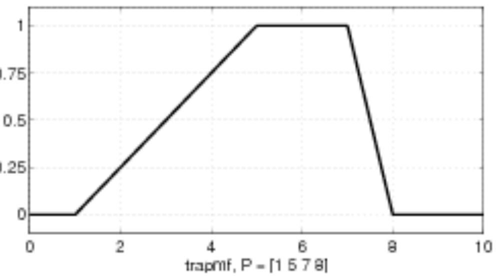
- piecewise linear functions
- the Gaussian distribution function
- the sigmoid curve
- quadratic and cubic polynomial curves

For detailed information on any of the membership functions mentioned next, turn to Functions — Alphabetical List. By convention, all membership functions have the letters `mf` at the end of their names.

The simplest membership functions are formed using straight lines. Of these, the simplest is the *triangular* membership function, and it has the function name `trimf`. This function is nothing more than a collection of three points forming a triangle. The *trapezoidal* membership function, `trapmf`, has a flat top and really is just a truncated triangle curve. These straight line membership functions have the advantage of simplicity.



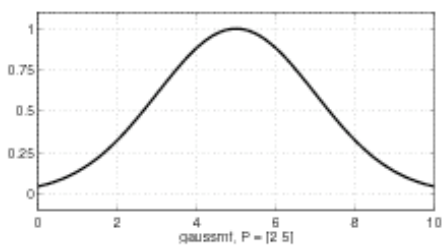
trimf



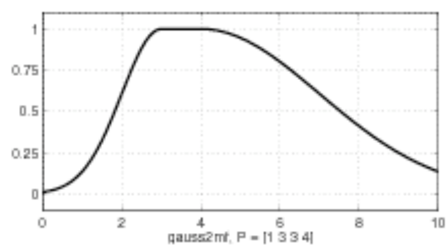
trapmf

Two membership functions are built on the *Gaussian* distribution curve: a simple Gaussian curve and a two-sided composite of two different Gaussian curves. The two functions are `gaussmf` and `gauss2mf`.

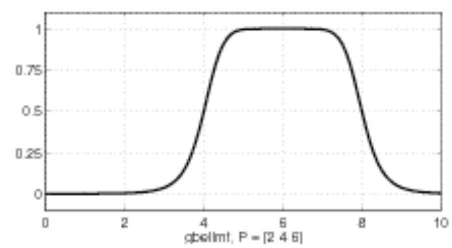
The *generalized bell* membership function is specified by three parameters and has the function name `gbellmf`. The bell membership function has one more parameter than the Gaussian membership function, so it can approach a non-fuzzy set if the free parameter is tuned. Because of their smoothness and concise notation, Gaussian and bell membership functions are popular methods for specifying fuzzy sets. Both of these curves have the advantage of being smooth and nonzero at all points.



gaussmf

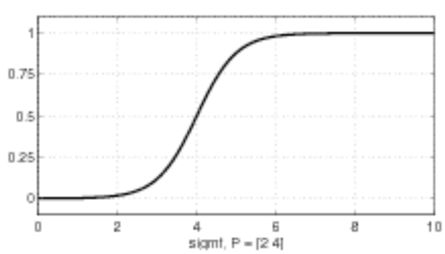


gauss2mf

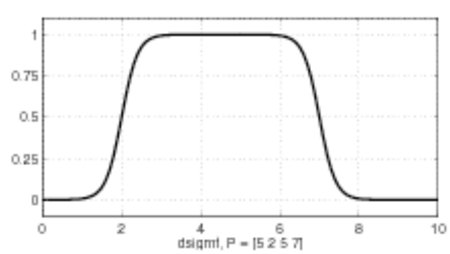


gbellmf

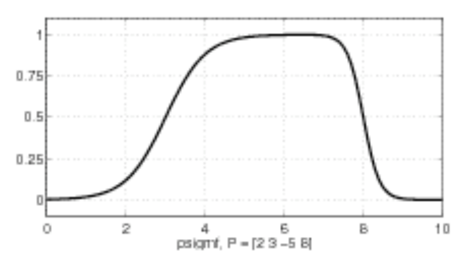
Although the Gaussian membership functions and bell membership functions achieve smoothness, they are unable to specify asymmetric membership functions, which are important in certain applications. Next, you define the *sigmoidal* membership function, which is either open left or right. Asymmetric and closed (i.e. not open to the left or right) membership functions can be synthesized using two sigmoidal functions, so in addition to the basic `sigmf`, you also have the difference between two sigmoidal functions, `dsigmf`, and the product of two sigmoidal functions `psigmf`.



sigmf

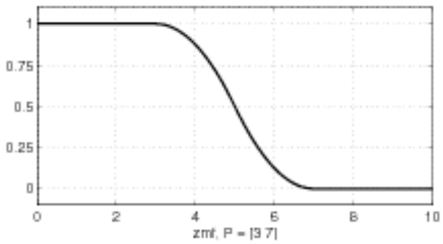


dsigmf

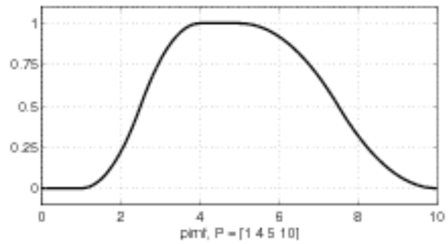


psigmf

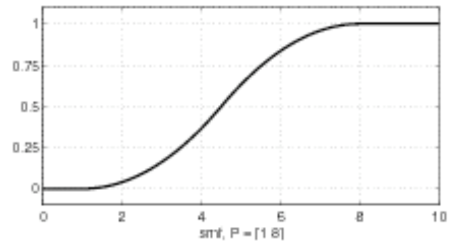
Polynomial based curves account for several of the membership functions in the toolbox. Three related membership functions are the *Z, S*, and *Pi curves*, all named because of their shape. The function `zmf` is the asymmetrical polynomial curve open to the left, `smf` is the mirror-image function that opens to the right, and `pimf` is zero on both extremes with a rise in the middle.



zmf



pimf



smf

There is a very wide selection to choose from when you're selecting your favorite membership function. Fuzzy Logic Toolbox also allows you to create your own membership functions if you find the list too restrictive. However, if a list based on expanded membership functions seems too complicated, just remember that you could probably get along very well with just one or two types of membership functions, for example the triangle and trapezoid functions. The selection is wide for those who want to explore the possibilities, but expansive membership functions are not necessary for good fuzzy inference systems. Finally, remember that more details are available on all these functions in the reference section.

Summary of Membership Functions

- Fuzzy sets describe vague concepts (e.g., fast runner, hot weather, weekend days).
- A fuzzy set admits the possibility of partial membership in it. (e.g., Friday is sort of a weekend day, the weather is rather hot).
- The degree an object belongs to a fuzzy set is denoted by a membership value between 0 and 1. (e.g., Friday is a weekend day to the degree 0.8).
- A membership function associated with a given fuzzy set maps an input value to its appropriate membership value.

Back to Top

Logical Operations

Now that you understand the fuzzy inference, you need to see how fuzzy inference connects with logical operations.

The most important thing to realize about fuzzy logical reasoning is the fact that it is a superset of standard Boolean logic. In other words, if you keep the fuzzy values at their extremes of 1 (completely true), and 0 (completely false), standard logical operations will hold. As an example, consider the following standard truth tables.

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

OR

A	not A
0	1
1	0

NOT

Now, because in fuzzy logic the truth of any statement is a matter of degree, can these truth tables be altered? The input values can be real numbers between 0 and 1. What function preserves the results of the AND truth table (for example) and also extend to all real numbers between 0 and 1?

One answer is the *min* operation. That is, resolve the statement *A* AND *B*, where *A* and *B* are limited to the range (0,1), by using the function *min*(*A*,*B*). Using the same reasoning, you can replace the OR operation with the *max* function, so that *A* OR *B* becomes equivalent to *max*(*A*,*B*). Finally, the operation NOT *A* becomes equivalent to the operation $1-A$. Notice how the previous truth table is completely unchanged by this substitution.

A	B	min(A,B)
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	max(A,B)
0	0	0
0	1	1
1	0	1
1	1	1

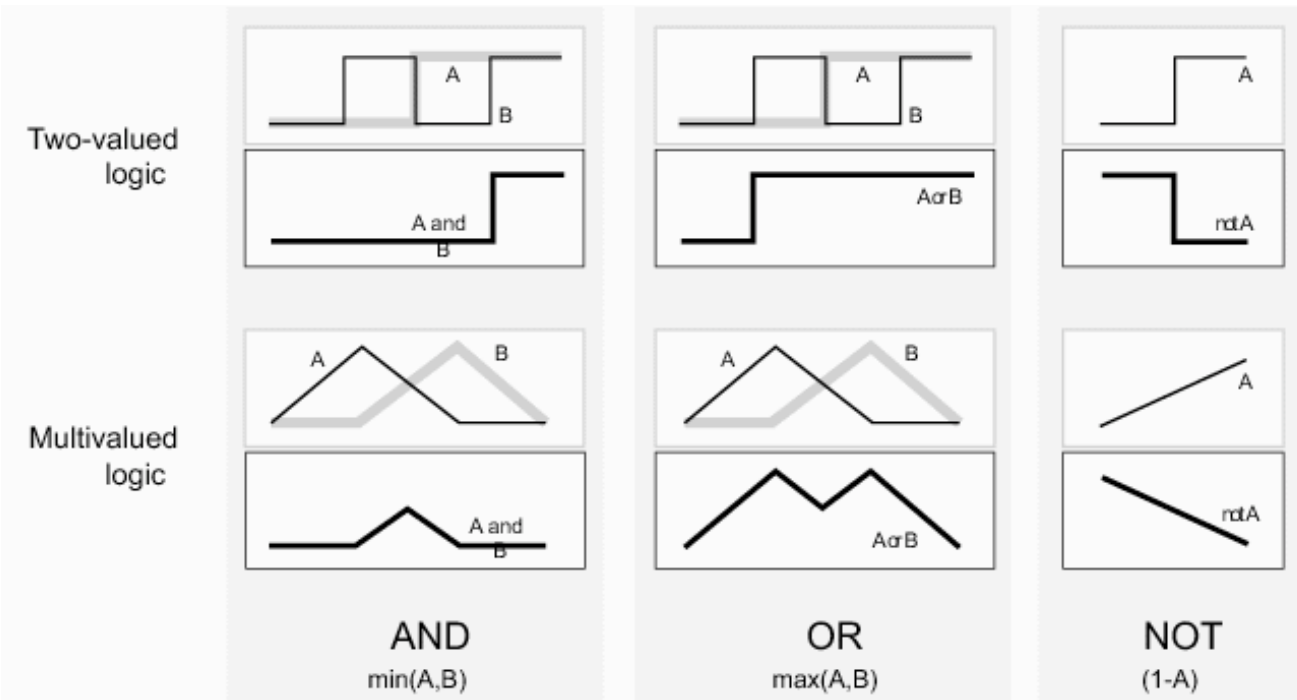
OR

A	1 - A
0	1
1	0

NOT

Moreover, because there is a function behind the truth table rather than just the truth table itself, you can now consider values other than 1 and 0.

The next figure uses a graph to show the same information. In this figure, the truth table is converted to a plot of two fuzzy sets applied together to create one fuzzy set. The upper part of the figure displays plots corresponding to the preceding two-valued truth tables, while the lower part of the figure displays how the operations work over a continuously varying range of truth values *A* and *B* according to the fuzzy operations you have defined.



Given these three functions, you can resolve any construction using fuzzy sets and the fuzzy logical operation AND, OR, and NOT.

Additional Fuzzy Operators

In this case, you defined only one particular correspondence between two-valued and multivalued logical operations for AND, OR, and NOT. This correspondence is by no means unique.

In more general terms, you are defining what are known as the fuzzy intersection or conjunction (AND), fuzzy union or disjunction (OR), and fuzzy complement (NOT). The classical operators for these functions are: AND = *min*, OR = *max*, and NOT = additive complement. Typically, most fuzzy logic applications make use of these operations and leave it at that. In general, however, these functions are arbitrary to a surprising degree. Fuzzy Logic Toolbox uses the classical operator for the fuzzy complement as shown in the previous figure, but also enables you to customize the AND and OR operators.

The intersection of two fuzzy sets *A* and *B* is specified in general by a binary mapping *T*, which aggregates two membership functions as follows:

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

For example, the binary operator *T* may represent the multiplication of $\mu_A(x)$ and $\mu_B(x)$. These fuzzy intersection operators, which are usually referred to as *T*-norm (Triangular norm) operators, meet the following basic requirements:

A *T-norm* operator is a binary mapping *T*(. , .) satisfying

boundary: *T*(0, 0) = 0, *T*(*a*, 1) = *T*(1, *a*) = *a*
monotonicity: *T(a, b)* <= *T(c, d)* if *a* <= *c* and *b* <= *d*
commutativity: T(a, b) = T(b, a)
associativity: T(a, T(b, c)) = T(T(a, b), c)

The first requirement imposes the correct generalization to crisp sets. The second requirement implies that a decrease in the membership values in *A* or *B* cannot produce an increase in the membership value in *A* intersection *B*. The third requirement indicates that the operator is indifferent to the order of the fuzzy sets to be combined. Finally, the fourth requirement allows us to take the intersection of any number of sets in any order of pairwise groupings.

Like fuzzy intersection, the fuzzy union operator is specified in general by a binary mapping *S*:

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

For example, the binary operator S can represent the addition of $\mu_A(x)$ and $\mu_B(x)$. These fuzzy union operators, which are often referred to as *T-conorm (or S-norm)* operators, must satisfy the following basic requirements:

A *T-conorm* (or *S-norm*) operator is a binary mapping *S*(. , .) satisfying

boundary: *S*(1, 1) = 1, *S*(*a*, 0) = *S*(0, *a*) = *a*
monotonicity: *S(a, b)* <= S(c, d) if *a* <= *c* and *b* <= *d*
commutativity: *S(a, b) = S(b, a)*
associativity: *S(a, S(b, c)) = S(S(a, b), c)*

Several parameterized *T*-norms and dual *T*-conorms have been proposed in the past, such as those of Yager[19], Dubois and Prade [3], Schweizer and Sklar [14], and Sugeno [15], found in the Bibliography. Each of these provides a way to vary the gain on the function so that it can be very restrictive or very permissive.

Back to Top

If-Then Rules

Fuzzy sets and fuzzy operators are the subjects and verbs of fuzzy logic. These if-then rule statements are used to formulate the conditional statements that comprise fuzzy logic.

A single fuzzy if-then rule assumes the form

if *x* is *A* then *y* is *B*

where *A* and *B* are linguistic values defined by fuzzy sets on the ranges (universes of discourse) X and Y, respectively. The if-part of the rule "*x* is *A*" is called the *antecedent* or premise, while the then-part of the rule "*y* is *B*" is called the *consequent* or conclusion. An example of such a rule might be

If service is good then tip is average

The concept *good* is represented as a number between 0 and 1, and so the antecedent is an interpretation that returns a single number between 0 and 1. Conversely, *average* is represented as a fuzzy set, and so the consequent is an assignment that assigns the entire fuzzy set B to the output variable *y*. In the if-then rule, the word *is* gets used in two entirely different ways depending on whether it appears in the antecedent or the consequent. In MATLAB terms, this usage is the distinction between a relational test using "==" and a variable assignment using the "=" symbol. A less confusing way of writing the rule would be

If service == good then tip = average

In general, the input to an if-then rule is the current value for the input variable (in this case, *service*) and the output is an entire fuzzy set (in this case, *average*). This set will later be *defuzzified,* assigning one value to the output. The concept of defuzzification is described in the next section.

Interpreting an if-then rule involves distinct parts: first evaluating the antecedent (which involves *fuzzifying* the input and applying any necessary *fuzzy operators*) and second applying that result to the consequent (known as *implication*). In the case of two-valued or binary logic, if-then rules do not present much difficulty. If the premise is true, then the conclusion is true. If you relax the restrictions of two-valued logic and let the antecedent be a fuzzy statement, how does this reflect on the conclusion? The answer is a simple one. if the antecedent is true to some degree of membership, then the consequent is also true to that same degree.

Thus:

in binary logic: $p \rightarrow q$ (*p* and *q* are either both true or both false.)
in fuzzy logic: $0.5\,p \rightarrow 0.5\,q$ (Partial antecedents provide partial implication.)

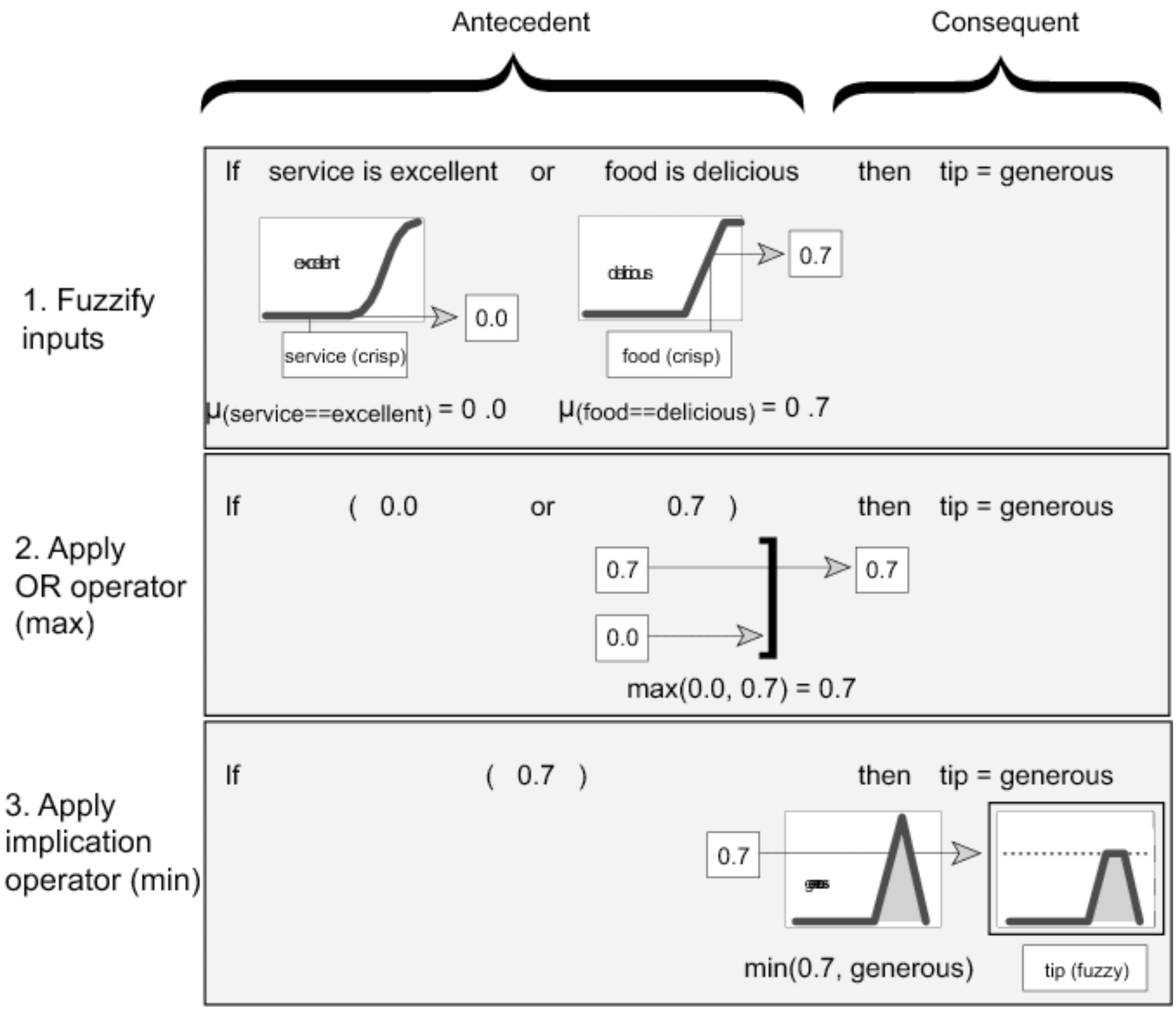
The antecedent of a rule can have multiple parts.

if sky is gray and wind is strong and barometer is falling, then ...

in which case all parts of the antecedent are calculated simultaneously and resolved to a single number using the logical operators described in the preceding section. The consequent of a rule can also have multiple parts.

if temperature is cold then hot water valve is open and cold water valve is shut

in which case all consequents are affected equally by the result of the antecedent. How is the consequent affected by the antecedent? The consequent specifies a fuzzy set be assigned to the output. The *implication function* then modifies that fuzzy set to the degree specified by the antecedent. The most common ways to modify the output fuzzy set are truncation using the `min` function (where the fuzzy set is chopped off as shown in the following figure) or scaling using the `prod` function (where the output fuzzy set is squashed). Both are supported by Fuzzy Logic Toolbox, but you use truncation for the examples in this section.



Summary of If-Then Rules

Interpreting if-then rules is a three-part process. This process is explained in detail in the next section:

1. **Fuzzify inputs**: Resolve all fuzzy statements in the antecedent to a degree of membership between 0 and 1. If there is only one part to the antecedent, then this is the degree of support for the rule.
2. **Apply fuzzy operator to multiple part antecedents**: If there are multiple parts to the antecedent, apply fuzzy logic operators and resolve the antecedent to a single number between 0 and 1. This is the degree of support for the rule.
3. **Apply implication method**: Use the degree of support for the entire rule to shape the output fuzzy set. The consequent of a fuzzy rule assigns an entire fuzzy set to the output. This fuzzy set is represented by a membership function that is chosen to indicate the qualities of the consequent. If the antecedent is only partially true, (i.e., is assigned a value less

than 1), then the output fuzzy set is truncated according to the implication method.

In general, one rule alone is not effective. Two or more rules that can play off one another are needed. The output of each rule is a fuzzy set. The output fuzzy sets for each rule are then aggregated into a single output fuzzy set. Finally the resulting set is defuzzified, or resolved to a single number. Fuzzy Inference Systems shows how the whole process works from beginning to end for a particular type of fuzzy inference system called a *Mamdani type*.

Back to Top

Provide feedback about this page

Overview Fuzzy Inference Systems