*Ersch. in: Intelligent data analysis : an introduction ; with 39 tables / ed. Michael Berthold ; David J. Hand. - Berlin [u.a.] : Springer, 1999. - S. 269-298. - ISBN 3-540-65808-4*

# Chapter 8
# Fuzzy Logic

Michael Berthold
University of California, Berkeley, USA

## 8.1. Introduction

In the previous chapters a number of different methodologies for the analysis of datasets have been discussed. Most of the approaches presented, however, assume precise data. That is, they assume that we deal with exact measurements. But in most, if not all real-world scenarios, we will never have a precise measurement. There is always going to be a degree of uncertainty. Even if we are able to measure a temperature of 32.42 degrees with two significant numbers, we will never know the exact temperature. The only thing we can really say is that a measurement is somewhere in a certain range, in this case $(32.41, 32.43)$ degrees. In effect, all recorded data are really intervals, with a width depending on the accuracy of the measurement. It is important to stress that this is different from probability, where we deal with the likelihood that a certain crisp measurement is being obtained [438]. In the context of uncertainty we are interested in the range into which our measurement falls. Several approaches to handle information about uncertainty have already been proposed, for example interval arithmetic allows us to deal and compute with intervals rather than crisp numbers [310], and also numerical analysis offers ways to propagate errors along with the normal computation [19].

This chapter will concentrate on presenting an approach to deal with imprecise concepts based on *fuzzy logic*. This type of logic enables us to handle uncertainty in a very intuitive and natural manner. In addition to making it possible to formalize imprecise numbers, it also enables us to do arithmetic using such *fuzzy numbers*. Classical set theory can be extended to handle partial memberships, thus making it possible to express vague human concepts using *fuzzy sets* and also describe the corresponding inference systems based on *fuzzy rules*.

Another intriguing feature of using fuzzy systems is the ability to granulate information. Using fuzzy clusters of similarity we can hide unwanted or useless information, ultimately leading to systems where the granulation can be used to focus the analysis on aspects of interest to the user.

The chapter will start out by explaining the basic ideas behind fuzzy logic and fuzzy sets, followed by a brief discussion of fuzzy numbers. We will then concentrate on fuzzy rules and how we can generate sets of fuzzy rules from data. We will close with a discussion of Fuzzy Information Theory, linking this chapter to chapter 6 by showing how Fuzzy Decision Trees can be constructed.

## 8.2. Basics of Fuzzy Sets and Fuzzy Logic

Before introducing the concept of fuzzy sets it is beneficial to recall classical sets using a slightly different point of view. Consider for example the set of "young people", assuming that our perception of a young person is someone with an age of not more than 20 years:

$$\text{young} = \{x \in P \mid \text{age}(x) \leq 20\}$$

over some domain $P$ of all people and using a function *age* that returns the age of some person $x \in P$ in years. We can also define a characteristic function:

$$m_{\text{young}}(x) = \begin{cases} 1 & : \quad \text{age}(x) \leq 20 \\ 0 & : \quad 20 < \text{age}(x) \end{cases}$$

which assigns to elements of $P$ a value of 1 whenever this element belongs to the set of young people, and 0 otherwise. This characteristic function can be seen as a *membership function* for our set *young*, defining the set *young* on $P$.

Someone could now argue with us that he, being just barely over 20 years old, still considers himself young to a very high degree. Defining our set *young* using such a sharp boundary seems therefore not very appropriate. The fundamental idea behind fuzzy set theory is now a variable notion of membership; that is, elements can belong to sets to a certain degree. For our example we could now specify that a person with an age of, let's say, 21 years, still belongs to the set of *young* people, but only to a degree of less than one, maybe 0.9. The corresponding membership function would look slightly different:

$$\mu_{\text{young}}(x) = \begin{cases} 1 & : \quad \text{age}(x) \leq 20 \\ 1 - \frac{(\text{age}(x) - 20)}{10} & : \quad 20 < \text{age}(x) \leq 30 \\ 0 & : \quad 30 < \text{age}(x) \end{cases}$$

Now our set *young* contains people with ages between 20 and 30 with a linearly decreasing *degree of membership*, that is, the closer someone's age approaches 30, the closer his degree of membership to the set of young people approaches zero (see Figure 8.1).

The above is a very commonly used example for a *fuzzy set*. In contrast to classical sets, where an element can either belong to a set or lies completely

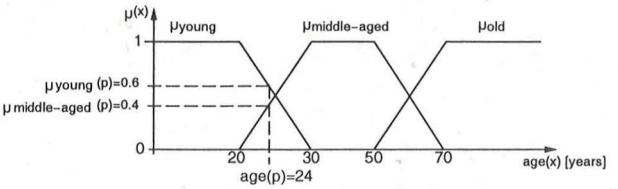


**Fig. 8.1.** A linguistic variable *age* with three fuzzy sets, and degrees of memberships for a certain age $a$.

outside of this set, fuzzy sets allow also partial memberships. A fuzzy set $A$ is thus defined through specification of a *membership function* $\mu_A$ that assigns each element $x$ a *degree of membership* to $A$: $\mu_A(x) \in [0, 1]$. Classical sets only allow values 1 (entirely contained) or 0 (not contained), whereas fuzzy set theory also deals with values in between 0 and 1. This idea was introduced in 1965 by Lotfi A. Zadeh [435].

### 8.2.1 Linguistic Variables and Fuzzy Sets

Covering the domain of a variable with several such fuzzy sets together with a corresponding semantic results in *linguistic variables*, allowing the computation with words. For our example this could mean that we define two more membership functions for middle-aged and old people, covering the entire domain of the variable *age*. This type of representation is especially appropriate for many real-world applications, where certain concepts are inherently vague in nature, either due to imprecise measurements or subjectivity.

The above example for a linguistic variable is shown in Figure 8.1. People are distinguished using their age (a function defined for all people) in groups of *young*, *middle-aged* and *old* people. Using fuzzy sets allows us to incorporate the fact that no sharp boundary between these groups exists. Figure 8.1 illustrates how the corresponding fuzzy sets overlap in these areas, forming non-crisp or fuzzy boundaries. Elements at the border between two sets belong to both. For example some person $p$ with an age of $\text{age}(p) = 24$ years belongs to both groups *young* and *middle-aged* to a degree of 0.6 and 0.4 resp.; that is, $\mu_{\text{young}}(p) = 0.6$ and $\mu_{\text{middle-aged}}(p) = 0.4$. With an increase in age, the degree of membership to the group of young people will decrease whereas $\mu_{\text{middle-aged}}$ increases. The linguistic variable *age* is therefore described through three *linguistic values*, namely *young*, *middle-aged*, and *old*. The overlap between the membership functions reflects the imprecise nature of the underlying concept. We should keep in mind, however, that most concepts depend on the respective context. An old student can easily be a young professor.

This way of defining fuzzy sets over the domain of a variable is often referred to as *granulation*, in contrast to the division into crisp sets (quantization) which is used by classical sets. Granulation results in a grouping of objects into imprecise clusters or *fuzzy granules*, with the objects forming a granule drawn together

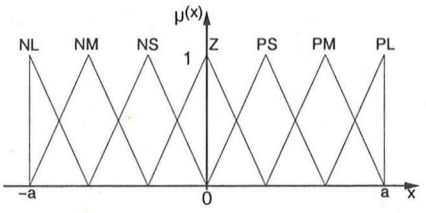


**Fig. 8.2.** The standard granulation using an odd number (here seven) of membership functions.

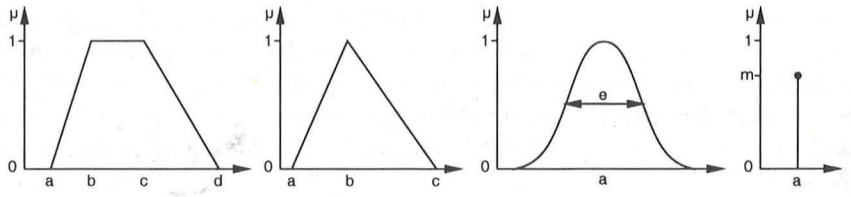


**Fig. 8.3.** Most commonly used shapes for membership functions (trapezoidal, triangular, Gaussian, singleton).

by similarity. Thus fuzzy quantization or granulation could also be seen as a form of fuzzy data compression. Often the granulation for some or all variables is obtained manually through expert interviews. If such expert knowledge is not available or the usage of a predefined granulation seems harmful, it is also possible to find a suitable granulation automatically. In the context of data analysis both approaches are used, depending on the focus of analysis. Later in this chapter we will discuss ways to use predefined granulations as well as algorithms that automatically build fuzzy clusters from data.

If no real semantic about the variable is known, a commonly used approach to label fuzzy sets is illustrated in Figure 8.2. For a symmetrical domain $[-a, a]$ of the variable often an odd number of membership functions is used, usually five or seven. The membership functions are then labeled `NL` (for "negative large"), `NM` ("negative medium"), `NS` ("negative small"), `Z` ("zero"), and the corresponding labels `PS`, `PM`, and `PL` for the positive side.

In real applications the shape of membership functions is usually restricted to a certain class of functions that can be specified with only few parameters. Figure 8.3 shows the most commonly used shapes for membership functions. On the left a trapezoidal function is depicted which can be specified through the four corner-points $< a, b, c, d >$. The triangular membership function can be seen as a special case of this trapezoidal function. Often used is also a Gaussian membership function which can be simply specified through two parameters $a$ and $e$ and offers nice mathematical properties such as continuity and differentiability. This is an often required property when membership functions are to be fine-tuned automatically during a training stage, as will be demonstrated later.

Finally the singleton $< a|m >$ on the right can be used to define a fuzzy set containing only one element to a certain degree $m \leq 1$. The choice of membership function is mostly driven by the application. The Gaussian membership functions are usually used when the resulting system has to be adapted through gradient-descent methods. Knowledge retrieved from expert interviews will usually be modeled through triangular or trapezoidal membership functions, since the three resp. four parameters used to define these functions are intuitively easier to understand. An expert will prefer to define his notion of a fuzzy set by specifying the area where the degree of membership should be 1 ($[b, c]$ for the trapezoid) and where it should be zero (outside of $(a, d)$), rather than giving mean $a$ and standard deviation $e$ of a Gaussian. The resulting fuzzy system will not be affected drastically. Changing from one form of the membership function to another will affect the system only within the boundaries of its granulation.

The following parameters can be defined and are often used to characterize any fuzzy membership function:

- *support:* $s_A := \{x : \mu_A(x) > 0\}$, the area where the membership function is greater than 0.
- *core:* $c_A := \{x : \mu_A(x) = 1\}$, the area for which elements have maximum degree of membership to the fuzzy set $A$. Note that the core can be empty when the membership function stays below 1 over the entire domain.
- *$\alpha$-cut:* $A_\alpha := \{x : \mu_A(x) \geq \alpha\}$, the cut through the membership function of $A$ at height $\alpha$.
- *height:* $h_A := \max_x\{\mu_A(x)\}$, the maximum value of the membership function of $A$.

Before discussing operators on fuzzy sets, the following section will discuss how fuzzy numbers can be treated.

### 8.2.2 Fuzzy Numbers

The motivation for using fuzzy numbers again stems from real-world applications. Real-world measurements, besides counting, are always imprecise in nature and a crisp number can not describe this fact adequately. Usually such measurements are modeled through a crisp number $x$ for the most typical value together with an interval describing the amount of imprecision. In a linguistic sense this could be expressed as *about x*. Using fuzzy sets we can incorporate this additional information directly. This results in *fuzzy numbers* which are simply a special type of fuzzy sets restricting the possible types of membership functions:

- the membership function must be *normalized* (i.e. the core is non-empty, $c_A \neq \emptyset$) and *singular*. This results in precisely one point which lies inside the core modeling the typical value of our fuzzy number. This point is often also called the *modal value* of the corresponding fuzzy number.
- in addition $\mu_A$ has to be monotonically increasing left of the core and monotonically decreasing on the right. This makes sure that only one peak and therefore only one typical value exists. The spread of the support (i.e. the non-zero area of the fuzzy set) describes the degree of imprecision.

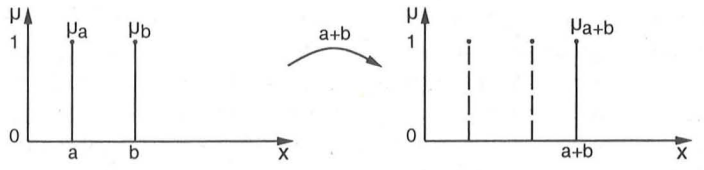


**Fig. 8.4.** An example for the membership functions of two crisp numbers and the result of adding the two.

Typically a triangular membership-function is chosen, making it possible to specify a fuzzy number through only three parameters $< a, b, c >$ (Fig. 8.3).

Of course, we now want to use these fuzzy numbers for normal calculations, for example we want to add or multiply two fuzzy numbers or apply a certain function to one or more fuzzy arguments.

For clarification let us first consider the classical, crisp version. Here one would, for example, add two numbers $a$ and $b$, resulting in $c = a + b$. This can also be illustrated as shown in Figure 8.4 using membership functions to represent the two numbers. The two singletons for $a$ and $b$ are then combined, resulting in a new singleton for $c$. Let us spend a moment to formulate this using membership functions for $a$ and $b$. Obviously these functions are simple singletons, that is, $\mu_a(x) = 1$ for $x = a$ and $\mu_a(x) = 0$ everywhere else (similarly for $\mu_b(x)$). We can now construct the membership function for $\mu_{a+b}$ from here:

$$\mu_{a+b}(x) = \begin{cases} 1 & \text{if } \exists y, z \in \mathbb{R} : y + z = x \wedge \mu_a(y) = 1 \wedge \mu_b(z) = 1 \\ 0 & \text{else} \end{cases}$$

In effect we define a function that assigns a degree of membership of 1 only to such points $x \in \mathbb{R}$ for which we can find points $y, z \in \mathbb{R}$ which also have a degree of membership of 1 using the membership functions for $a$ and $b$ and, in addition, who satisfy $y + z = x$. As expected for the crisp case, this results in another singleton, this time at the point $a + b$.

Extending classical operators to their fuzzy counterparts can now be achieved similarly to the above, using the so-called extension principle [437]. We only have to extend the above method to also handle intermediate degrees of membership. For an arbitrary binary operator $\star$, this would be interpreted as follows:

$$\mu_{A \star B}(x) = \max_{y,z \in \mathbb{R}} \{\min\{\mu_A(y), \mu_B(z)\} \mid y \star z = x\}$$

that is, for a value $x \in \mathbb{R}$ a degree of membership is derived which is the maximum of $\min\{\mu_A(y), \mu_B(z)\}$ over all possible pairs of $y, z \in \mathbb{R}$ for which $y \star z = x$ holds.

In other words, the new membership function assigns the maximum degree of membership which can be achieved by finding the best combination of parameters on the domain $\mathbb{R}$ of the involved variables. In effect, such operators convert fuzzy sets into a new fuzzy set, describing the result of the operation.

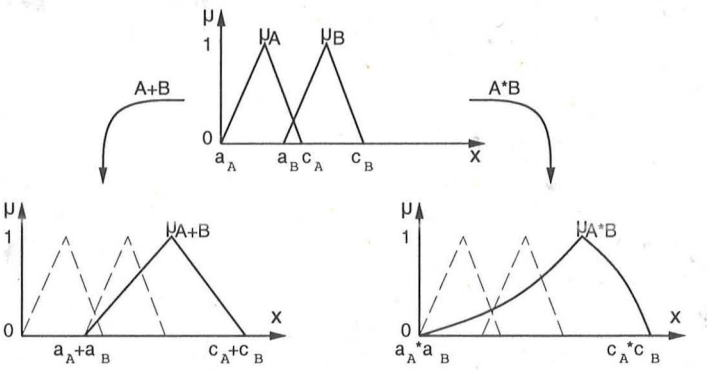


**Fig. 8.5.** An example for adding (left) and multiplying (right) two fuzzy numbers.

Summation of two fuzzy numbers *about a* and *about b* would then result in a somewhat "fuzzier" version *(very) about a+b*. (It is interesting to note, however, that the result of 2·*about a* - *about a* and *about a* might not be the same when using the extension principle sequentially.) This extension principle can easily be extended to binary and also higher dimensional operators. Figure 8.5 demonstrates the result for addition and multiplication of two triangular fuzzy numbers. It is obvious how the resulting fuzzy numbers maintain their triangular shape after addition and become more complex after multiplication. They still maintain all properties required for fuzzy numbers but loose their triangular shape. Subtraction and division can be handled similarly, but for the latter special care has to be taken for cases where $\mu_B(0) \neq 0$, that is, where 0 is part of the quotient because then the resulting fuzzy set does not exist.

Whenever a non-linear function is applied to such fuzzy numbers, the triangular shape is not maintained. Similar to above, the extension principle is used again, resulting in:

$$\mu_{f(A)}(y) = \max\{\mu_A(x) \mid \forall x : f(x) = y\}$$

Figure 8.6 demonstrates this effect (In section 8.2.6 also the concept of fuzzy functions in contrast to such crisp functions will be discussed).

The fact that triangular fuzzy numbers are not closed under all operators, that is, their original (i.e. triangular) representation is not maintained through all operations, results in difficulties implementing calculations on fuzzy numbers, since the error of such a triangular approximation can possibly grow extremely fast [156]. A possible ad-hoc solution through an approximate representation uses a certain number of $\alpha$-cuts for computation. On each of these $\alpha$-cuts an exact computation can be done, but the linear approximation between two neighboring $\alpha$-cuts again leads to a possibly large approximation error. The more different $\alpha$-cuts are used, the smaller is the loss in accuracy, but also the complexity grows. Another approach is based on using a polynomial representation with respect to $\alpha$. This representation is closed under addition and multiplication, that is,

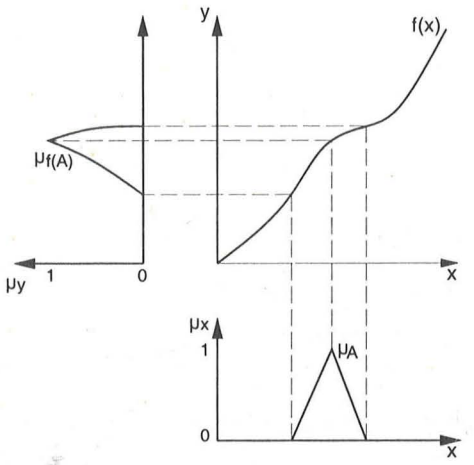


**Fig. 8.6.** An example for applying a function to a fuzzy number.

the result of adding or multiplying such fuzzy numbers will again be a polynom in $\alpha$. This makes it possible to find an internal (though not necessarily finite) representation for such fuzzy numbers. Several other parametric representations of fuzzy numbers have been proposed, a good overview can be found in [156]. A more extensive discussion of fuzzy numbers can be found in [283].

### 8.2.3 Fuzzy Sets and Fuzzy Logic

Similar to the arithmetic operators defined on fuzzy numbers, classical operators from boolean logic like conjunction, disjunction, and complement can also be extended to fuzzy sets. Consider an example where we want to find the set of people that are both, young and tall. Using classical logic we would construct the new set using boolean conjunction:

$$m_{\text{tall and young}}(x) = \begin{cases} 1 & : \quad m_{\text{tall}}(x) = 1 \quad \text{and} \quad m_{\text{young}}(x) = 1 \\ 0 & : \quad \text{else} \end{cases}$$

Using fuzzy sets defining such a strict condition is undesirable. Assume for example a person that belongs to the (fuzzy) set *young* to a degree of 0.5 and to the set *tall* to a degree of 0.8. What degree of membership should he or she have for the set of *young* **and** *tall* people? Should we use the minimum of the two degrees of membership? Another possibility would be the product, or one could even imagine more complicated arithmetic functions.

In effect, here the question arises how continuous truth-values can be processed. In contrast to the case of fuzzy numbers we are not interested in combining the entire fuzzy sets, but want to retrieve a new degree of membership resulting from an operation on one or more existing degrees of membership. Obviously an entire family of operators can be defined to derive the resulting

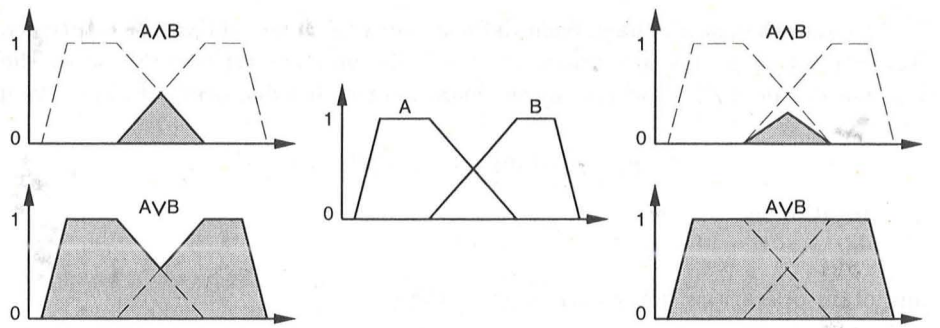


**Fig. 8.7.** Two variants for interpretation of fuzzy union and fuzzy intersection. Zadeh's min/max-interpretation (left) and the product/bounded sum-variant (right).

membership functions. For the case of fuzzy logic, the most prominent examples were introduced by Lotfi Zadeh in [435], and are defined as follows:

- conjunction: $\mu_{A\wedge B}(x) := \min\{\mu_A(x), \mu_B(x)\}$
- disjunction: $\mu_{A\vee B}(x) := \max\{\mu_A(x), \mu_B(x)\}$
- complement: $\mu_{\neg A}(x) := 1 - \mu_A(x)$

For our example this would mean that the person belongs to the group of *young* and *tall* people to a degree of 0.5 ($= \min\{0.8, 0.5\}$).

Subsequently several other possible functions to interpret fuzzy set operators were proposed. Most of them can be formalized using the concept of T-norms (for the conjunction) and T-conorms (disjunction), the chapter by Gottwald in [324] and also [420] discuss these norms in more detail. The min/max-version above represents the most optimistic resp. most pessimistic version. Other common choices include the product for conjunction and the bounded sum for disjunction:

- conjunction II: $\mu_{A\wedge B}(x) := \mu_A(x) \cdot \mu_B(x)$
- disjunction II: $\mu_{A\vee B}(x) := \min\{\mu_A(x) + \mu_B(x), 1\}$

Figure 8.7 illustrates these two norms. As was pointed out in [257], however, using these definitions not all tautologies known from classical logic are true for the fuzzy case. For example consider the following (use $\mu_A(x) = 0.3$)[1]:

- $A \wedge \neg A \neq \emptyset$
- $A \vee \neg A \neq I$

Hence we can not simply use laws from classical logic to derive other operators. Most notably we have to define implication, rather than deriving it from disjunction and negation. For the min/max-interpretation the following definition for fuzzy implication is most often used (motivated by $A \rightarrow B = \neg A \vee (A \wedge B)$):

$$\mu_{A\rightarrow B}(x) := \max\{1 - \mu_A(x), \min\{\mu_A(x), \mu_B(x)\}\}$$

[1] $I$ and $\emptyset$ are defined according to classical logic, that is, $\mu_I \equiv 1$ and $\mu_\emptyset \equiv 0$.

Several other norms have been defined, most of them motivated intuitively. Another set of norms are axiomatizable [257], and are all isomorphic to the Lukasiewicz-norm [278] which can be defined through a definition of implication as:

$$\mu_{A \to B}(x) := \min\{1, 1 - \mu_A(x) + \mu_B(x)\}$$

and negation as:

$$\mu_{\neg A} = 1 - \mu_A$$

and other operations being derived from there:

- disjunction III or strong-or, derived from $A \vee B = \neg A \to B$:

$$\mu_{A \vee B} = \min\{1, \mu_A + \mu_B\}$$

- conjunction III or strong-and, derived from $A \wedge B = \neg(\neg A \vee \neg B)$:

$$\mu_{A \wedge B} = 1 - \min\{1, 1 - \mu_A + 1 - \mu_B\} = \max\{0, \mu_A + \mu_B - 1\}$$

It should be noted, however, that also these definitions are not valid under all tautologies known from boolean logic. If the disjunction operator is, for example, derived through DeMorgan's law and $(A \to B) \to B$, the result will be the well-known maximum-operator.

In the next section it will be shown how these different definitions for fuzzy-implication affect the interpretation of inference in a fuzzy context.

### 8.2.4 Generalized Modus Ponens

In classical logic, conclusions can be drawn from known facts and implications based on these facts. In fuzzy logic this process of inference can be extended also to partial true facts, resulting in a generalized version of the classical Modus Ponens:

$$\frac{\begin{array}{l} x \texttt{ IS } A' \\ \texttt{IF } x \texttt{ IS } A \texttt{ THEN } y \texttt{ IS } B \end{array}}{y \texttt{ IS } B'}$$

where $A'$ is the membership function for $x$, which does not have to be the same as in the antecedent of the implication (therefore the name *generalized* Modus Ponens). In terms of fuzzy membership functions this can be expressed as follows:

$$\frac{\begin{array}{l} \mu_{A'}(x) \\ A \to B \end{array}}{\mu_{B'}(y)}$$

According to the previous different definitions of implication, different interpretations of such fuzzy inference can arise:

- Joint Constraint: using min-max norms, the implication can be seen as forming a constraint on a joint variable, that is, $(x, y)$ is $A \times B$, where $\times$ denotes the Cartesian product $\mu_{A\times B}(x, y) = \min\{\mu_A(x), \mu_B(y)\}$. $B'$ can now be obtained through $B' = A' \wedge (A \times B)$, or

$$\mu_{B'}(v) = \sup_u \{\min\{\mu_A(u), \mu_{A\times B}(u, v)\}\}.$$

  Note that if $A'$ is similar to $A$, also $B'$ will be similar to $B$. On the other hand a very dissimilar $A'$ will result in an empty set $B'$, i.e. $\mu_{B'} \equiv 0$.
- Conditional Constraint: using Lukasiewicz's definition a possibility dependency arises: $\text{Poss}(x = u|y = v) = \min\{1, 1 - \mu_A(u) + \mu_B(v)\}$. Also here $B'$ can be easily obtained through:

$$\mu_{B'}(v) = \sup_u \{\min\{\mu_A(u), 1 - \mu_A(u) + \mu_B(v)\}\}.$$

  If $A'$ is very dissimilar to $A$ the resulting $B'$ will be generic, i.e. $\mu_{B'} \equiv 1$.

Both definitions lead to a very different interpretation of fuzzy implication. In most rule-based applications the interpretation of fuzzy implication as a constraint on joint variables is used, because this variant is easier to implement and computationally less expensive. The remainder of this chapter will focus on this interpretation, but it should always be kept in mind that other interpretations exist (for a more detailed discussion see [325]). This lack of a generally accepted interpretation of fuzzy implication is often cited as a disadvantage of fuzzy logic, but if the user is aware of the potential of both interpretations they can provide interesting alternatives for different foci of analysis.

### 8.2.5 Fuzzy Rules

Fuzzy rules can be used to characterize imprecise dependencies between different variables. Consider for example a classical rule:

$$\texttt{IF age}(x) \leq 25 \texttt{ THEN risk}(x) > 60\%$$

describing the risk-factor for a car-insurance company. Obviously using linguistic variables can make such a rule much more readable:

$$\texttt{IF age}(x) \texttt{ IS young THEN risk}(x) \texttt{ IS high}$$

Fuzzy rules are therefore of interest whenever a dependency is either imprecise or a high level of precision is not desired in order to maintain a high level of interpretability. A basic type of a categorical rule[2] which is widely used in control and other applications has the following form:

$$\texttt{IF } x_1 \texttt{ IS } A_1 \texttt{ AND } \ldots \texttt{ AND } x_n \texttt{ IS } A_n \texttt{ THEN } y \texttt{ IS } B$$

[2] Qualified rules that assign additional constraints to rules (such as "very likely" or "not very true"), will not be dealt with in this chapter since they are not widely used.

where the $A_i$ in the antecedent and the $B$ in the consequent are linguistic values of the input vector $\boldsymbol{x}$ and the output variable $y$, respectively. These types of rules are called *Mamdani rules* [282]. Most often, the consequent only consists of one variable and multi-dimensional cases are modeled through separate rules for each output.

In many modeling applications, also rules that assign simple (crisp) equations to the output variable are used. Most commonly these are either linear or quadratic dependencies on one or more input variables (first order or second order *Takagi-Sugeno models* [400]). Note that in this case the input $\boldsymbol{x}$ is required to consist of scalar values:

$$\texttt{IF } x_1 \texttt{ IS } A_1 \texttt{ AND } \ldots \texttt{ AND } x_n \texttt{ IS } A_n \texttt{ THEN } y = f(\boldsymbol{x})$$

Note that by using this expression, the output is no fuzzy set, but rather a singleton $< y, w >$. The degree of membership $w$ of this singleton is equal to the degree of fulfillment of the antecedent:

$$w = \min\{\mu_{A_1}(x_1), \cdots, \mu_{A_n}(x_n)\}.$$

If several such rules exist, the corresponding output-singletons are combined using a fuzzy aggregation operator. Most often, the product is used to compute $\wedge$ and a weighted sum is used for aggregation (assuming $r$ rules):

$$y_{\text{final}} = \frac{\sum_{i=1}^{r} w_i \cdot y_i}{\sum_{i=1}^{r} w_i}.$$

The resulting system is then similar to Radial Basis Function Networks, as discussed in chapter 7. Rules of this type are commonly used in control applications. In the context of data analysis applications, however, the above-mentioned Mamdani rules with fuzzy consequent "$\cdots$ THEN $y$ IS $B$" are usually given preference because of their better interpretability.

Using these types of rules, our car-insurance agency might now describe a particular group of young drivers with high-output engines in their cars as follows:

IF *age* IS *young* AND *car power* IS *high* THEN *risk* IS *high*

As mentioned in section 8.2.4, such an implication is usually interpreted as a constraint on a joint variable. If a min-norm is used, this results in a "clipping" of the fuzzy set of the consequent. Sometimes also the product-norm is used, resulting in a scaling of the consequent's membership function.

A fuzzy rule based system is a set of such fuzzy rules. The process of combining the individual results is often done through composition. An example for a fuzzy rule set in action using min/max-norms is shown in Figure 8.8. The rule-set contains two fuzzy `if-then`-rules for the example car-insurance risk-predictor:

IF *age* IS *young* AND *car power* IS *high* THEN *risk* IS *high*

IF *age* IS *middle aged* AND *car power* IS *medium* THEN *risk* IS *medium*

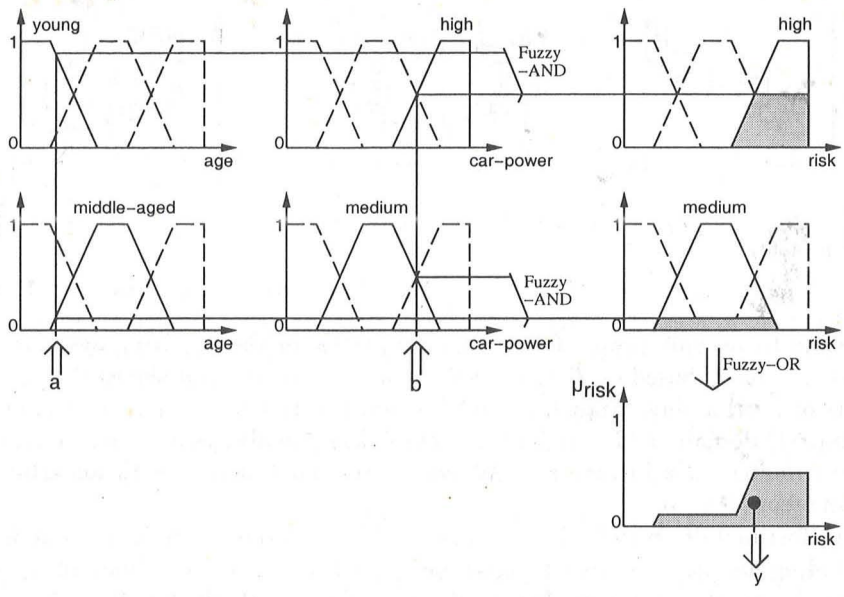


**Fig. 8.8.** An example for a fuzzy rule base in action. Receiving the crisp inputs age=a and car-power=b the defuzzified output will be risk=y.

The first rule describes the fact that young drivers with high-power cars pose high risks for a car-insurance. Similarly the second rule describes middle-aged drivers with normal (i.e. medium power) cars. Figure 8.8 now shows how – using min-max-inference – a crisp input of *age=a* and *car-power=b* is being processed. For both rules the degrees of membership for the corresponding fuzzy terms are computed using Fuzzy-AND. This results in a clipped membership function of the rule-conclusion and together they produce the resulting membership function for the linguistic variable *risk* using Fuzzy-OR. A number of methods to determine a crisp output value are available. The most common one is shown in Figure 8.8. Here the crisp value $y$ is determined from the generated membership function $\mu_{\text{risk}}$ through:

$$y = \int y \cdot \mu_{\text{risk}}(y) \mathrm{d}y$$

In practical applications it is often ignored that the membership functions of the linguistic values can overlap and the crisp output is then easily computable through:

$$y = \frac{\sum_{j=1}^{r} \mu_j \cdot s_j}{\sum_{j=1}^{r} \mu_j}$$

where $r$ indicates the number of rules, $\mu_j$ the corresponding degree of membership for each linguistic value and $s_j$ denotes the center of gravity of the membership function. This method of *defuzzification* is known as COG- or *Center of Gravity*-method. It is worthwhile to note that using such rule sets it is

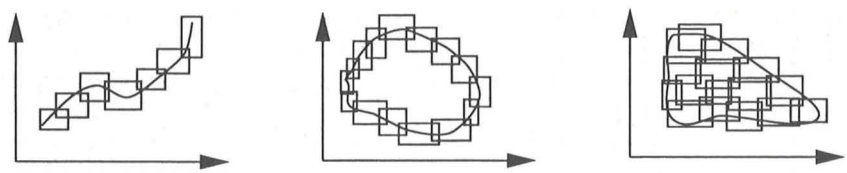

**Fig. 8.9.** Approximate representations of functions, contours, and relations (after Zadeh [439]).

possible to provide proofs for certain properties of the resulting system. The most notable is based on the Stone-Weierstrass theorem and shows that on the basis of overlapping, triangular membership functions, which cover the entire (compact) domain of all attributes, a set of rules can always be constructed that approximates an arbitrary real-valued, continuous function with an arbitrary small error [419, 70].

Unfortunately, such global granulation of the input space will result in an exploding number of rules for high-dimensional spaces. Hence, different ways to cover the input space using fuzzy rules were developed which will be discussed in the next section.

### 8.2.6 Fuzzy Graphs

To avoid a global granulation of the feature space the concept of fuzzy graphs can be used [439]. In contrast to a global grid, specified through linguistic variables as described above, fuzzy graphs employ individual membership functions for each fuzzy rule; that is, the membership function of the constraint on the joint variable is defined individually for each rule. Based on the interpretation of implication as a constraint on a joint variable, a set of individual fuzzy rules can also be seen as forming a fuzzy graph. Through the concept of such fuzzy graphs approximate representations of functions, contours, and sets can be derived (Figure 8.9).

In the following we will concentrate on fuzzy graphs for the representation of functions which approximate only one output variable[3] $y$ depending on $n$ input variables $x_i$ $(1 \leq i \leq n)$.

The typical fuzzy graphs build upon Mamdani rules similar to the ones mentioned in the previous section:

$$R: \texttt{IF } x_1 \texttt{ IS } A_1 \texttt{ AND } \ldots \texttt{ AND } x_n \texttt{ IS } A_n \texttt{ THEN } y \texttt{ IS } B$$

Some of the input membership functions $A_i$ can be constant ($\mu_{A_i} \equiv 1$). Thus different fuzzy rules may only depend on a subset of input variables. This is extremely beneficial in applications in high-dimensional spaces, where such rules will be much easier to interpret. Simplifying the above equation, using $\boldsymbol{A} =$

[3] Multidimensional output variables can easily be modeled through a set of fuzzy graphs.

$A_1 \times \cdots \times A_n$ (again $\times$ denotes the Cartesian product), leads to:

$$\texttt{IF } x \texttt{ IS } A \texttt{ THEN } y \texttt{ IS } B.$$

And, as mentioned in section 8.2.4, such a rule can also be seen as a fuzzy constraint on a joint variable $(x, y)$, that is,

$$(x, y) \texttt{ IS } A \times B$$

In this representation $A \times B$ is often called a *fuzzy point*. Using Zadeh's min/max-based interpretation of fuzzy operators, the membership function of $A \times B$ can be computed as discussed above:

$$\mu_{A \times B}(x, y) = \min \{\mu_A(x), \mu_B(y)\}$$

or, more precisely

$$\mu_{A \times B}(x, y) = \min \{\mu_{A_1}(x_1), \cdots, \mu_{A_n}(x_n), \mu_B(y)\}$$

A collection of such rules can now be regarded as forming a superposition of $r$ fuzzy points:

$$(x, y) \texttt{ IS } (A_1 \times B_1 + \cdots + A_r \times B_r)$$

where $+$ denotes the disjunction operator (here defined as maximum). Zadeh calls this characterization of a dependency *fuzzy graph*, because the collection of rules can be seen as a coarse representation of a functional dependency $f^*$ of $y$ on $x$. This fuzzy graph $f^*$ can thus be defined as:

$$f^* = \sum_{j=1}^{r} (A_j \times B_j)$$

The task of interpolation, that is, deriving a linguistic value $B$ for $y$ given an arbitrary linguistic value $A$ for $x$ and a fuzzy graph $f^*$:

$$\begin{array}{l} x \texttt{ IS } A \\ f^* \texttt{ IS } \sum_{j=1}^{r} A_j \times B_j \\ \hline y \texttt{ IS } B \end{array}$$

results in an intersection of the fuzzy graph $f^*$ with a cylindrical extension of the input fuzzy set $A$:

$$\text{proj}_Y\{(A \times I) \cap f^*\}$$

with $A \times I$ denoting the cylindrical extension along $y$ and $\text{proj}_Y$ the projection on $Y$. Figure 8.10 shows an example. This functional dependency can be computed through:

$$\begin{aligned} \mu_B(y) &= \mu_{f^*(A)}(y) = \sup_x \{\min\{\mu_{f^*}(x, y), \mu_A(x)\}\} \\ &= \sup_x \{\min\{\max\{\mu_{A_1 \times B_1}(x, y), \cdots, \mu_{A_r \times B_r}(x, y)\}, \mu_A(x)\}\} \end{aligned}$$

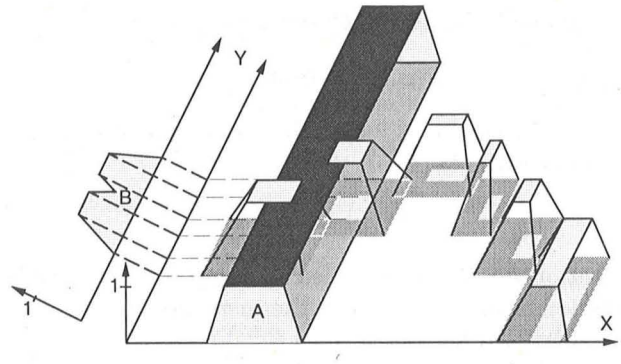


**Fig. 8.10.** Interpolation as the intersection of a fuzzy graph with a cylindrical extension of $A$.

## 8.3. Extracting Fuzzy Models from Data

In the context of intelligent data analysis it is of great interest how such fuzzy models can automatically be derived from example data. Since, besides prediction, understandability is of prime concern, the resulting fuzzy model should offer insights into the underlying system. To achieve this, different approaches exist that construct grid-based rule sets defining a global granulation of the input space, as well as fuzzy graph based structures. Approaches that produce such models will be presented in the next two sections. Afterwards we will also briefly discuss a fuzzy clustering algorithm, similar to the one described in chapter 7.

### 8.3.1 Extracting Grid-Based Fuzzy Models from Data

Grid-based rule sets model each input variable through a usually small set of linguistic values. The resulting rule base uses all or a subset of all possible combinations of these linguistic values for each variable, resulting in a global granulation of the feature space into "tiles":

$$\begin{array}{lll} R_{1,\cdots,1}: & \texttt{IF} \;\; x_1 \;\texttt{IS}\; A_{1,1} & \texttt{AND...AND} \;\; x_n \;\texttt{IS}\; A_{1,n} \;\; \texttt{THEN...} \\ & \vdots \qquad\qquad & \vdots \\ R_{l_1,\cdots,l_n}: & \texttt{IF} \;\; x_1 \;\texttt{IS}\; A_{l_1,1} & \texttt{AND...AND} \;\; x_n \;\texttt{IS}\; A_{l_n,n} \;\; \texttt{THEN...} \end{array}$$

where $l_i$ $(1 \leq i \leq n)$ indicates the numbers of linguistic values for variable $i$ in the $n$-dimensional feature space. Figure 8.11 illustrates this approach in two dimensions with $l_1 = 3$ and $l_2 = 2$. Extracting grid-based fuzzy models from data is straightforward when the input granulation is fixed, that is, the antecedents of all rules are predefined. Then only a matching consequent for each rule needs to be found.

Wang&Mendel [418] presented a straightforward approach to learning fixed-grid Mamdani models. After predefinition of the granulation of all input variables

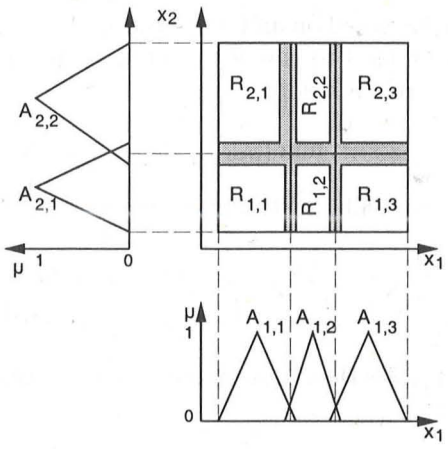


**Fig. 8.11.** A global granulation of the input space using three membership functions for $x_1$ and two for $x_2$.

and also the output variable, one sweep through the entire dataset determines the closest example to the geometrical center of each rule, assigning the closest output fuzzy value to the corresponding rule:

1. Granulate the input and output space. Divide each variable $x_i$ into $l_i$ equidistant triangular membership functions. Similarly the granulation into $l_y$ membership functions for the output variable $y$ is determined, resulting in the typical overlapping distribution of triangular membership functions. Figure 8.11 illustrates this approach in two dimensions with 3 resp. 2 membership functions, resulting in six tiles.
2. Generate fuzzy rules from given data. For the example in Figure 8.11, this means that we have to determine the best consequence for each rule. For each example pattern $(\boldsymbol{x}, y)$ the degree of membership to each of the possible tiles is determined:

$$\min\left\{\mu_{\mathrm{msx}_{j_1,1}}(x_1), \cdots, \mu_{\mathrm{msx}_{j_n,n}}(x_n), \mu_{\mathrm{msy}_{j_y}}(y)\right\}$$

   with $1 \leq j_i \leq l_i$ and $1 \leq j_y \leq l_y$. Then $\mathrm{msx}_{j_i,i}$ indicates the membership function of the $j_i$-th linguistic value of input variable $i$ and similar for msy for the output variable $y$. Next the tile resulting in the maximum degree of membership is used to generate one rule:

$$R_{(j_1,\cdots,j_n)} : \texttt{IF } x_1 \texttt{ IS } \mathrm{msx}_{j_1,1} \cdots \texttt{ AND } x_n \texttt{ IS } \mathrm{msx}_{j_n,n}$$
$$\texttt{THEN } y \texttt{ IS } \mathrm{msy}_{j_y}$$

   assuming that tile $(j_1, \cdots, j_n, j_y)$ resulted in the highest degree of membership for the corresponding training pattern.
3. Assign a rule weight to each rule. The degree of membership will in addition be assigned to each rule as rule-weight $\beta_{(j_1,\cdots,j_n)}$.

4. Determine an output based on an input-vector. Given an input $\boldsymbol{x}$ the resulting rule-base can be used to compute a crisp output $\hat{y}$. First the degree of fulfillment for each rule is computed:

$$\mu_{(j_1,\cdots,j_n)}(\boldsymbol{x}) = \min\left\{\mu_{\text{msx}_{j_1,1}}(x_1),\cdots,\mu_{\text{msx}_{j_n,n}}(x_n)\right\}$$

then the output $\hat{y}$ is combined through a centroid defuzzification formula:

$$\hat{y} = \sum_{j_1=1,\cdots,j_n=1}^{l_1,\cdots,l_n} \frac{\beta_{(j_1,\cdots,j_n)} \cdot \mu_{(j_1,\cdots,j_n)}(\boldsymbol{x}) \cdot \overline{y}_{(j_1,\cdots,j_n)}}{\beta_{(j_1,\cdots,j_n)} \cdot \mu_{(j_1,\cdots,j_n)}(\boldsymbol{x})}$$

where $\overline{y}_{(j_1,\cdots,j_n)}$ denotes the center of the output region of the corresponding rule with index $(j_1,\cdots,j_n)$.

Figure 8.12 demonstrates this algorithm using four membership functions for the input variable $x$ and output $y$. The graph in the top left corner shows the example points used to generate the fuzzy model. On the right the used granulation is shown, and the data points which lie closest to the centers of the respective rules are marked (thick circles). At the bottom left the generated set of rules is shown, represented by their $\alpha = 0.5$-cuts:

$$\begin{array}{lll}
R_1: & \texttt{IF } x \texttt{ IS } zero_x & \texttt{THEN } y \texttt{ IS } medium_y \\
R_2: & \texttt{IF } x \texttt{ IS } small_x & \texttt{THEN } y \texttt{ IS } medium_y \\
R_3: & \texttt{IF } x \texttt{ IS } medium_x & \texttt{THEN } y \texttt{ IS } large_y \\
R_4: & \texttt{IF } x \texttt{ IS } large_x & \texttt{THEN } y \texttt{ IS } medium_y
\end{array}$$

and on the right the resulting crisp approximation is shown (thick line). Note how the generated model misses extrema that lie far from existing rule-centers. Intuitively rule $R_2$ should probably be used to describe the minimum of the function:

$$R_2': \texttt{IF } x \texttt{ IS } small_x \texttt{ THEN } y \texttt{ IS } small_y$$

This behavior is due to the fact that only one pattern per rule is used to determine the outcome of this rule. Even a combined approach would very much depend on the predefined granulation. If the function to be modeled has a high variance inside one rule, the resulting rule model will fail to model this behavior. One interesting aspect of this approach is the availability of a proof that arbitrary real continuous functions over a compact set can be approximated to arbitrary accuracy. This proof is based on the Stone-Weierstrass theorem and can be found in [418].

For practical applications it is obvious, however, that using such a predefined, fixed grid results in a fuzzy model that will either not fit the underlying functions very well or consist of a large number of rules. This is why approaches are of more interest that fine-tune or even automatically determine the granulations of both input and output variables.

An approach that builds upon Wang&Mendel's fixed-grid algorithm was proposed by Higgins&Goodman in [204]. Initially only one membership function is

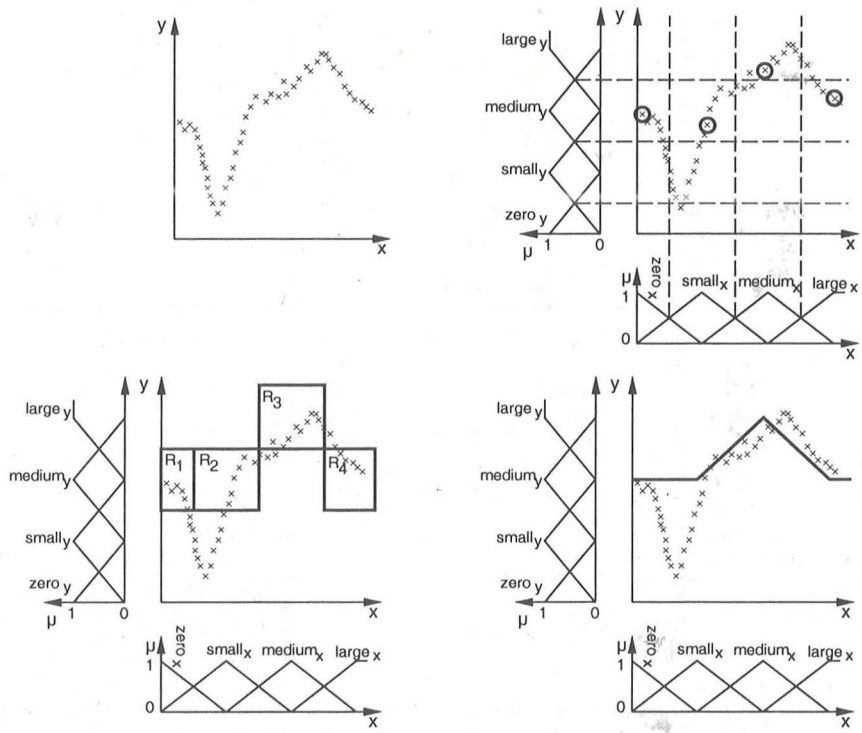


**Fig. 8.12.** An example for a fixed-grid fuzzy rule set produced by the Wang&Mendel-algorithm [418]. On the top the data points (left) and the used granulation (right) are shown. The figures at the bottom show the generated rules (left) and the resulting crisp approximation (right).

used to model each of the input variables as well as the output variable, resulting in one large rule covering the entire feature space. Subsequently new membership functions are introduced at points of maximum error. This is done until a maximum number of divisions is reached or the approximation error remains below a certain threshold. Figure 8.13 demonstrates this algorithm for the same set of data points as used before to demonstrate the Wang&Mendel algorithm in Figure 8.12. Here training was stopped after a maximum of four membership functions was generated for each variable and the graphs show all four steps. Each graph shows the generated membership functions and the corresponding approximation for the corresponding instance. The circles indicate the point of maximum error at each step. In the top left corner one large rule covers the entire input space:

$$\texttt{IF } x \texttt{ IS } A_x \texttt{ THEN } y \texttt{ IS } A_y$$

The next step generates a second rule, describing the maximum of the example data points:

$$\texttt{IF } x \texttt{ IS } B_x \texttt{ THEN } y \texttt{ IS } B_y$$

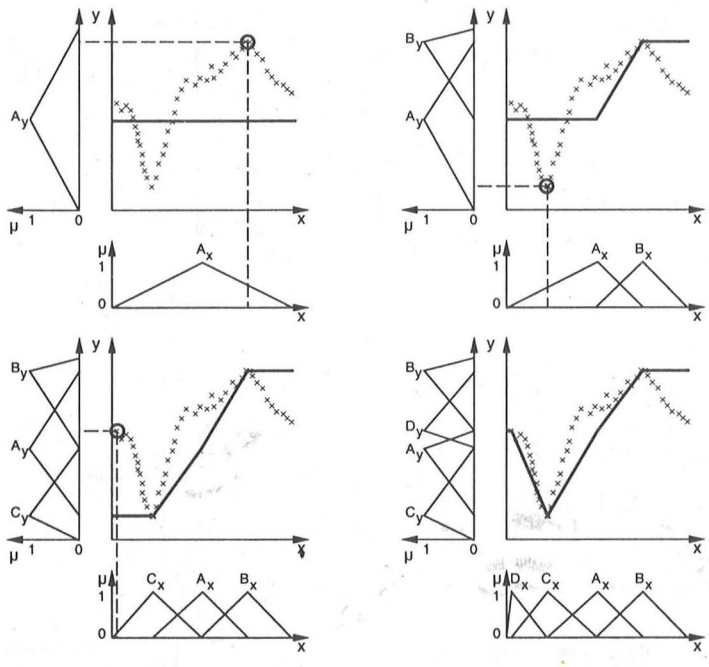


**Fig. 8.13.** An example how a grid-based fuzzy rule set will be produced by the Higgins&Goodman-algorithm [204] in four steps. The thick line shows the approximation and the circle indicates the point resulting in maximum error at this instance.

During the third step another rule is generated:

$$\texttt{IF } x \texttt{ IS } C_x \quad \texttt{THEN } y \texttt{ IS } C_y$$

The final step generates yet another rule, but note how in the final set of four rules the consequent of the second rule is changed from $B_y$ to $C_y$:

$$\begin{array}{l} \texttt{IF } x \texttt{ IS } A_x \quad \texttt{THEN } y \texttt{ IS } A_y \\ \texttt{IF } x \texttt{ IS } B_x \quad \texttt{THEN } y \texttt{ IS } C_y \\ \texttt{IF } x \texttt{ IS } C_x \quad \texttt{THEN } y \texttt{ IS } C_y \\ \texttt{IF } x \texttt{ IS } D_x \quad \texttt{THEN } y \texttt{ IS } D_y \end{array}$$

This is due to the underlying Wang&Mendel algorithm, which is used to generate the intermediate set of rules. At each step, new membership functions for all variables are introduced and therefore different choices can be made for the rules' output, effectively moving the output closer to the output value of the particular training pattern also in other regions then the one that caused the creation of new membership functions. Obviously this approach is able to model extrema much better than the Wang&Mendel algorithm alone, but has a definite preference to favor extrema and therefore a strong tendency to concentrate on outliers.

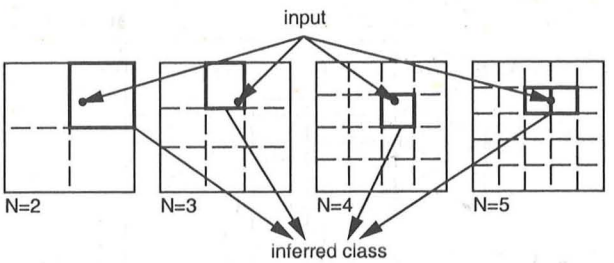


**Fig. 8.14.** An example for the multi-rule approach presented in [326]. Several rule sets are initialized, using different levels of granulation.

A hierarchical algorithm was presented in [326]. Here an ensemble of rule sets with different granulation is built at the beginning. Starting with a coarse granulation (usually having only two membership functions for each attribute), the remaining rule sets have increasingly finer granulation. Figure 8.14 illustrates this method. In the resulting multi-rule table the grade of certainty for each rule is then used for pruning; that is, rules with a low grade of certainty will be removed from the rule set.

Several other approaches still require a predefined grid but fine-tune the initial position of the grid-lines to better match the data. Similar to the Error Backpropagation algorithm for Neural Networks described in chapter 7, a heuristic version of gradient descent is used in [321]. The position and the shape of the membership functions is altered iteratively to reduce the overall approximation error.

Other approaches convert the fuzzy model into a neural network and use conventional training algorithms. In [4] an approach based on Radial Basis Function Networks with elliptical regions is described. The used training algorithm, however, needs to be restrained to maintain a possibility to extract a meaningful fuzzy model after training has finished. Also Genetic Algorithms (see chapter 9) can be used to fine tune an existing rule set. In [218] such an approach was presented. An algorithm which generates positive and negative rules using evolutionary concepts can be found in [253].

### 8.3.2 Extracting Fuzzy Graphs from Data

We mentioned before that especially in high-dimensional feature spaces a global granulation results in a large number of rules. For these tasks a fuzzy graph based approach is more suitable. Here the individual fuzzy rules are defined through independent membership functions in the feature space:

$$\begin{array}{l} R_1: \texttt{IF } x_1 \texttt{ IS } A_{1,1} \texttt{ AND...AND } x_n \texttt{ IS } A_{1,n} \texttt{ THEN...} \\ \qquad\qquad \vdots \qquad\qquad \vdots \\ R_r: \texttt{IF } x_1 \texttt{ IS } A_{r,1} \texttt{ AND...AND } x_n \texttt{ IS } A_{r,n} \texttt{ THEN...} \end{array}$$

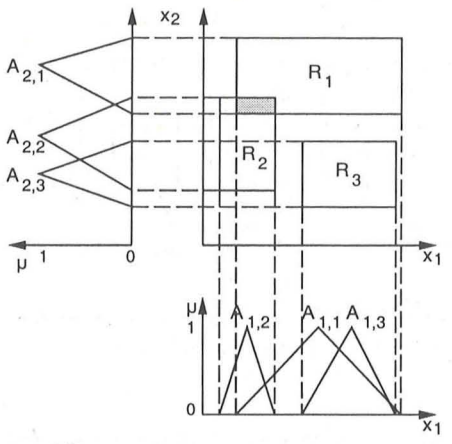


**Fig. 8.15.** A granulation of the input space using individual membership functions for each rule.

Figure 8.15 shows an example in two dimensions using three rules. For comparison, Figure 8.11 in the previous section shows a global, grid-based granulation. A possible disadvantage of the individual membership functions is the potential loss of interpretation. Projecting all membership functions onto one variable will usually not lead to meaningful linguistic values. In many data analysis applications, however, such a meaningful granulation of all attributes is either not available or hard to determine automatically.

An algorithm that constructs such fuzzy graphs based on example data was presented in [40]. The only parameter specified by the user is the granulation of the output variable $y$; that is, the number and shape of the membership functions of $y$ have to be determined manually[4]. In most applications this is no disadvantage, because it enables the user to define foci of attention or areas of interest where a finer granulation is desired. Thus $c$ fuzzy sets are defined through membership functions $\mu_y^k$ ($y$ indicating that this membership function is defined over the domain of the output variable $y$) for each output region $k$, with $1 \leq k \leq c$. The algorithm then iterates over the example data, and subsequently fine-tunes the evolving model.

The resulting fuzzy graph consists of a set of fuzzy points or fuzzy rules $R_j^k$, $1 \leq j \leq r_k$, where $r_k$ indicates the number of rules for the output region $k$. A rule's activity $\mu_j^k(\boldsymbol{x})$ indicates the degree of membership of the pattern $\boldsymbol{x}$ to the corresponding rule $R_j^k$ (note the difference between $\mu_j^k$ defined on input $x_j$ and $\mu_y^k$, defined on the output $y$). Using the normal notation of fuzzy rules, each rule can be decomposed into $n$ individual, one-dimensional membership functions:

$$\mu_j^k(\boldsymbol{x}) = \min_{i=1,\ldots,n} \left\{\mu_{j,i}^k(x_i)\right\}$$

[4] In case of multi-dimensional output variables, several fuzzy graphs can be built independently.

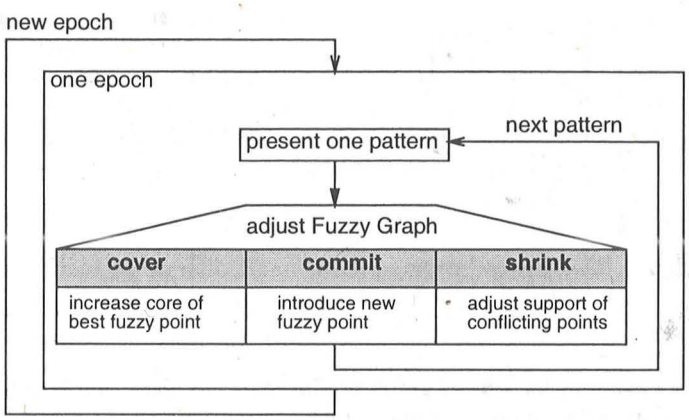


**Fig. 8.16.** The algorithm to construct a fuzzy graph based on example data [40].

where $\mu_{j,i}^k$ indicates the projection of $\mu_j^k$ onto the $i$-th attribute. The degree of membership for output-region $k$ is then computed through:

$$\mu^k(\boldsymbol{x}) = \max_{1 \leq j \leq r_k} \{\mu_j^k(\boldsymbol{x})\}$$

The algorithm relies on trapezoidal membership functions, so each rule can be described through four parameters for each dimension:

$$\texttt{IF } x_1 \texttt{ IS } < a_1, b_1, c_1, d_1 > \texttt{ AND } \cdots x_n \texttt{ IS } < a_n, b_n, c_n, d_n > \texttt{ THEN } y \texttt{ IS } \mu_y^k$$

It is important to note, however, that some trapezoids can cover the entire domain of the attribute, making the rule's degree of fulfillment independent of this dimension.

The flow of the underlying algorithm is illustrated in Figure 8.16. All existing example patterns are subsequently presented. During such an epoch, three different steps are executed. The algorithm ensures that each example pattern is covered by a fuzzy rule of the region to which it belongs and that rules of conflicting regions do not cover the pattern. This enables the fuzzy graph to tolerate moderately noisy patterns or small oscillations along boundaries (see [40] for an example). The three main steps of the algorithm introduce new fuzzy points when necessary and adjust the core- and support-regions of existing ones in case of conflict:

- **covered**: if the new training pattern lies inside the support-region of an already existing fuzzy point and it belongs to the correct output-region, the core-region of this fuzzy point is extended to cover the new pattern.
- **commit**: if a new pattern is not covered, a new fuzzy point belonging to the correct output-region will be introduced. The new example pattern is assigned to its core, whereas the support-region is initialized "infinite"; that is, the new fuzzy point covers the entire domain.

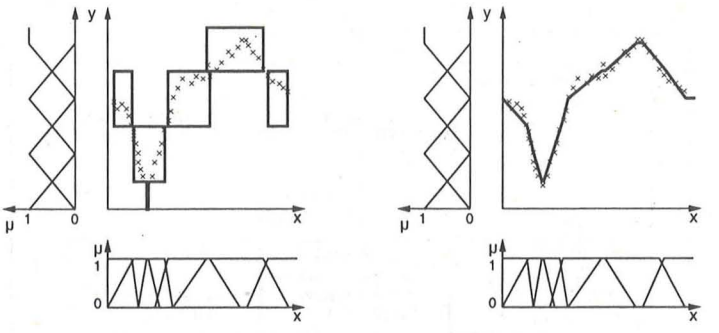


**Fig. 8.17.** An example for a fuzzy graph produced by the algorithm in [40] (left) along with the corresponding approximation (right).

- **shrink**: if a new pattern is incorrectly covered by an already existing fuzzy point of a conflicting region this fuzzy point's support-area will be reduced (e.g. shrunk) so that the conflict is solved. The underlying heuristic how this shrink is conducted aims to maximize the remaining volume. Details can be found in [40].

The algorithm clearly terminates after only few iterations over the set of example patterns. The final set of fuzzy points then forms a fuzzy graph, where each fuzzy point is associated with one output region and is used to compute a membership value for a certain input pattern. The maximum degree of membership of all fuzzy points for one region determines the overall degree of membership. Fuzzy inference then produces a soft value for the output and using the well-known center-of-gravity method a final crisp output value can be obtained, if so desired. Approximation of the above used example-function produces the result shown in Figure 8.17. Here the output variable was divided into four soft regions. The fuzzy graphs tend to ignore oscillations that lie inside output regions, an effect often desirable in real applications. The input variable is divided into many membership functions, some of them extremely thin. In scenarios with multi-dimensional inputs these membership functions may even overlap each other. Fuzzy graphs do not have a natural linguistic interpretation of the granulation of their input space. The main advantage is the low dimensionality of the individual rules. The algorithm only introduces restriction on few of the available input variables, thus making the extracted rules easier to interpret.

Other approaches that build fuzzy graphs or independent fuzzy rule sets based on example data were presented in [385, 386]. In [3] an algorithm was presented that also generates negative fuzzy rules, thus allowing to handle conflicting regions where rules of different classes overlap. Several of the rule learning concepts discussed in chapter 6 have also been extended to the fuzzy case. In section 8.4 an algorithm that builds a Fuzzy ID-3 decision tree will be discussed, along with the necessary extension of information theoretical concepts to the fuzzy case.

### 8.3.3 Fuzzy Clustering

In strong similarity to the clustering algorithm discussed in chapter 7, a fuzzy version can also be derived. This is especially interesting in cases where different clusters overlap or noisy patterns interfere with the cluster building process. In contrast to classical clustering techniques, a fuzzy cluster algorithm does not try to assign each pattern to exactly one cluster. Instead, a degree of membership to each cluster is derived as well. Hence each pattern has a degree of compatibility with each cluster, rather then belonging to only one cluster.

As usual the clustering algorithm starts with a set of $m$ patterns $\boldsymbol{x}_i$ ($1 \leq i \leq m$) and a predefined number of clusters $c$. The result of the clustering will be a set of prototypes, or cluster-centers, $\boldsymbol{u}_j$ ($1 \leq j \leq c$). A fuzzy clustering algorithm will then, in addition, produce a degree of membership $\mu_{i,j}$ to each of these clusters $j$ for all patterns $i$.

A first example of such an algorithm, called fuzzy c-means, was explained in [41]. Here we will concentrate on a slightly different version which was presented in [252] and aims to minimize the following objective function (as usual $n$ indicates the dimensionality of the feature space):

$$\sum_{j=1}^{c}\sum_{i=1}^{m}\mu_{i,j}^{n}\cdot \mathrm{d}(\boldsymbol{x}_i,\boldsymbol{u}_j)+\sum_{j=1}^{c}\eta_j\sum_{i=1}^{m}(1-\mu_{i,j})^n.$$

The first part of this objective function demands that the distances $\mathrm{d}(\boldsymbol{x}_i,\boldsymbol{u}_j)$ from the prototype vectors $\boldsymbol{u}_j$ to the patterns $\boldsymbol{x}_i$ is as low as possible, whereas the second term forces the degrees of membership to be as large as possible, thus avoiding the trivial solution with all $\mu_{i,j}$ being equal to zero. The $\eta_j$ are factors to normalize and control the influence of this second term.

The resulting partition of the feature space can be regarded as a possibilistic partition, and the degrees of membership can be seen as degrees of compatibility with respect to the cluster centers. We will not go into detail about the formalism to construct these clusters since this is very similar to the one presented in chapter 7. A detailed discussion of different approaches for the fuzzy case can be found for example in [252] and [102].

## 8.4. Fuzzy Decision Trees

In strong similarity to chapter 6, an extension to decision trees based on fuzzy logic can be derived. Different branches of the tree are then distinguished by fuzzy queries, an example for such a fuzzy decision tree is shown in Figure 8.18. Note how the decisions on each edge are fuzzy constraints rather than crisp conditions. In the following one possible extension to the concepts of information theory will be presented based on [225], followed by an example of a fuzzy ID3-algorithm to build decision trees.

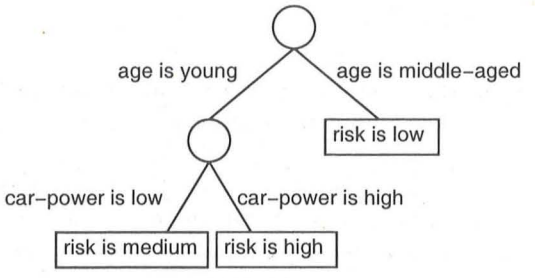


**Fig. 8.18.** An example fuzzy decision tree for a simple insurance risk predictor.

### 8.4.1 Fuzzy Information Theory

Similar to probability a measure of a-priori membership can be defined for each class $k$ ($1 \leq k \leq c$) based on a data sample $T = \{\boldsymbol{x}_j : 1 \leq j \leq m\}$ and a degree of membership $\mu^k(\boldsymbol{x})$ of $\boldsymbol{x}$ to class $k$:

$$\mathcal{M}^k(T) = \frac{1}{|T|} \sum_{\boldsymbol{x} \in T} \mu^k(\boldsymbol{x})$$

(of course initially $|T| = m$, but later we will also have to deal with subsets of $T$, making this notation more consistent). Because fuzzy sets are not required to sum to one over the entire domain (as is the case for probabilities), the a-priori measure of membership for all classes is also not always equal to 1:

$$\mathcal{M}(T) = \sum_{k=1}^{c} \mathcal{M}^k(T)$$

Based on these measures we can now define a standard information content:

$$\mathcal{I}(T) = -\sum_{k=1}^{c} \frac{\mathcal{M}^k(T)}{\mathcal{M}(T)} \cdot \log \frac{\mathcal{M}^k(T)}{\mathcal{M}(T)}.$$

As discussed previously in chapter 6, this measure will reflect the amount of information contained in this soft assignment to various classes. If only one class has a degree of membership $> 0$, then $\mathcal{I}(T)$ will be zero. On the other hand, if several classes have a non-zero degree of membership, $\mathcal{I}(T)$ will be closer to one.

For the construction of decision trees, it is now of interest how much information we can gain by asking specific questions. Let us, therefore, assume that we can impose a constraint on the data sample, that is, define a fuzzy condition $C$ that assigns a degree of fulfillment to each sample in $T$. In most cases this would represent a one-dimensional fuzzy condition, similar to a rule `IF` $x_i$ `IS` $C$ `THEN` $\cdots$. Now we can compute the conditional information:

$$\mathcal{M}^k(T|C) = \frac{1}{|T_C|} \sum_{\boldsymbol{x} \in T_C} \min\{\mu^k(\boldsymbol{x}), \mu_C(\boldsymbol{x})\}$$

$$\mathcal{M}(T|C) = \sum_{k=1}^{c} \mathcal{M}^k(T_C)$$

From there it is possible to derive a conditional information content:

$$\mathcal{I}(T|C) = -\sum_{k=1}^{c} \frac{\mathcal{M}^k(T|C)}{\mathcal{M}(T|C)} \cdot \log \frac{\mathcal{M}^k(T|C)}{\mathcal{M}(T|C)}$$

and we can easily derive how much information was obtained by applying the constraint $C$. The information gain computes to:

$$\mathcal{I}(T;C) = \mathcal{I}(T) - (\mathcal{I}(T|C) + \mathcal{I}(T|\neg C)).$$

Here we assume that we asked a question with only two possible outcomes, $C$ and $\neg C$. Of course we could also have more than two outcomes (for example if we have several linguistic values for an attribute), then the information gain computes to:

$$\mathcal{I}(T;C) = \mathcal{I}(T) - \sum_{\text{cond} \in C} \mathcal{I}(T|\,\text{cond}).$$

when we have a set $C$ of fuzzy constraints.

Using these measures we can now generate fuzzy decision trees, using an algorithm very similar to ID3 which was described in chapter 6.

### 8.4.2 Fuzzy ID3

The following procedure to build a fuzzy decision tree is described in more detail in [225] and was applied to the recognition of handwritten numericals in [82]. In the following we will concentrate on the main algorithm which generates the decision tree itself.

We assume that in addition to the set of data examples $T$, a granulation for the input and output variables is available, dividing each attribute into a finite set of (soft) regions[5]. For each input variable $x_i$ $(1 \leq i \leq n)$, this set of fuzzy terms is denoted by $D_i = \{v_p^i\}$. For each node $N$ of the decision tree

- $F^N$ denotes the set of fuzzy restrictions on the path from $F^{ROOT}$ leading to $N$ $(F^{ROOT} = \emptyset)$.
- $V^N$ is the set of attributes appearing on the path from $F^{ROOT}$ leading to $N$ $(V^{ROOT} = \emptyset)$. This assures that an attribute is only used once to come to a decision, analogous to the original ID3.
- $\chi$ assigns a membership to each training example at $N$ $(\forall \boldsymbol{x} \in T : \chi^{ROOT}(\boldsymbol{x}) = 1)$. If the training examples bear initial weights $\neq 1$ this can be taken into account as well. $\chi$ models the distribution of training patterns from $T$ at different nodes. In contrast to classical decision trees where $T$ would be split into disjunctive subsets, here only the degrees of memberships are adjusted, resulting in different values of $\chi$ for different nodes.

[5] In case of a classification task this granulation needs only to be done for the input variables since the output already consists of a finite number of classes.

- $N|v_p^j$ denotes the child of $N$ following the edge $v_p^j$ which was created by using the test "$V_j$ IS $v_p^j$" to split $N$.

The procedure to build the decision tree then operates iteratively.

1. At any node $N$ still to be expanded compute the degree of membership for each class
$$\mathcal{M}^k(N) = \frac{1}{\sum_{x \in T} \chi(x)} \sum_{x \in T} \min\left\{\mu^k(x), \chi(x)\right\}$$
and the entire node:
$$\mathcal{M}(N) = \sum_{k=1}^{c} \mathcal{M}^k(N)$$
2. Determine the information content of this node:
$$\mathcal{I}(N) = -\sum_{k=1}^{c} \frac{\mathcal{M}^k(N)}{\mathcal{M}(N)} \cdot \log \frac{\mathcal{M}^k(N)}{\mathcal{M}(N)}.$$
3. At each node we search the remaining attributes $V^{ROOT} - V^N$ to split this node:
   - compute the conditional information content, considering a test on $v_p^j$ for attribute $V_j \in V^{ROOT} - V^N$:
$$\mathcal{M}^k(N|V_j \text{ is } v_p^j) = \frac{1}{\sum_x \chi(x)} \sum_{x \in T} \min\left\{\mu^k(x), \mu_{v_p^j}(x), \chi(x)\right\}$$
$$\mathcal{M}(N|V_j \text{ IS } v_p^j) = \sum_{k=1}^{c} \mathcal{M}^k(N|V_j \text{ IS } v_p^j)$$
   and the conditional information content:
$$\mathcal{I}(N|V_j \text{ IS } v_p^j) = -\sum_{k=1}^{c} \frac{\mathcal{M}^k(N|V_j \text{ IS } v_p^j)}{\mathcal{M}(N|V_j \text{ IS } v_p^j)} \cdot \log \frac{\mathcal{M}^k(N|V_j \text{ IS } v_p^j)}{\mathcal{M}(N|V_j \text{ IS } v_p^j)}.$$
   - select attribute $V_j$ such that the information gain
$$\mathcal{I}(N) - \sum_{v_p^j \in V_j} \mathcal{I}(N|V_j \text{ IS } v_p^j)$$
   is maximized. In the example in Figure 8.19 this attribute would be $x_2$.
4. Split $N$ into $|V_j|$ subnodes $N|v_p^j$. The new memberships are computed using the fuzzy restriction leading to this node through
$$\chi^{N|v_p^j}(x) = \min\left\{\chi^N(x), \mu_{v_p^j}(x)\right\}.$$
In Figure 8.19 two new nodes are created and the tables show the new values for $\chi$. Note how two patterns have nonzero degrees of membership for both nodes. Here a crisp decision using $x_2$ can not be made, therefore these elements maintain two memberships.

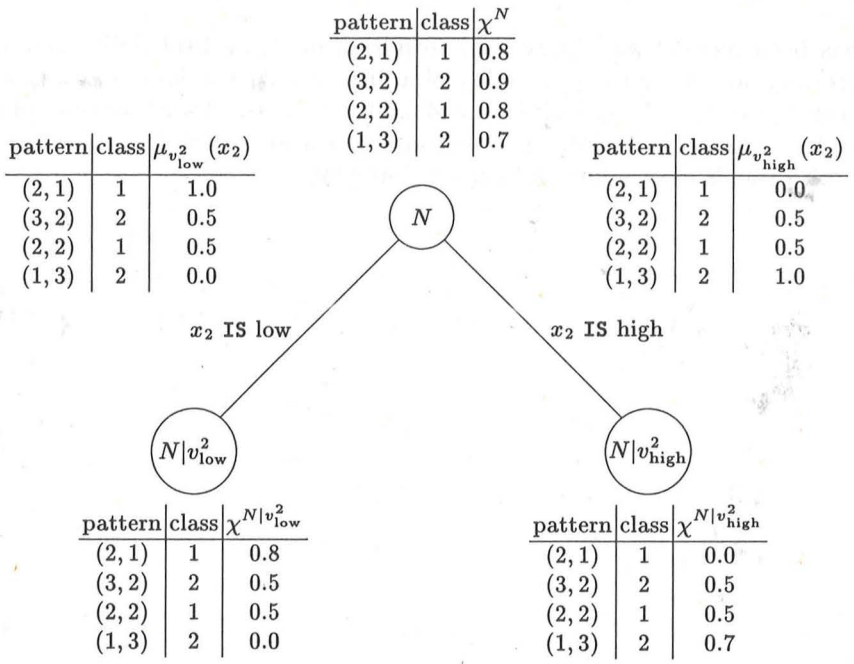


**Fig. 8.19.** An example for a split of one node of a Fuzzy ID3 decision tree into two children. The tables show the new degrees of membership $\chi^{N|v^2_*}$ computed from the original node $N$ and the degrees of membership $\mu_{v^2_*}$ for each pattern.

5. Continue until a certain depth of the tree is reached or node $N$ has an overall membership $\mathcal{M}(N)$ below a certain threshold.

The described training procedure is the same as for ID3. The major difference comes from the fact that training examples can be found in more than one node to a certain degree of membership $\chi(\boldsymbol{x})$.

## 8.5. Conclusion

This chapter has discussed how fuzzy systems can be used for the analysis of data sets. In the scope of this book, however, only basic ideas were presented. Obviously more sophisticated methodologies have been proposed, chapter 9, for example, touches on the subject of fine-tuning fuzzy rules by using genetic algorithms. We have also completely ignored how fuzzy rules can be used to inject expert knowledge into neural networks and how training algorithms from this area can then be used to adjust the membership functions of the corresponding rules. In [321] such Neuro-Fuzzy Systems are described in more detail.

In fact, most of the topics discussed in this book have in one way or another links with fuzzy logic, even probability – often seen as a rival to fuzzy logic.

It has been merged with fuzzy logic, resulting in Fuzzy Probability and such interesting notions as the probability of a fuzzy event. The interested reader is referred to books on fuzzy logic [321, 324] and also to the relevant journals in this area, for example [442, 444]. Of additional interest are certainly two collections of Lotfi Zadeh's most influential papers [240, 433].