Computational Intelligence

Master AI

2023-2024

Àngela Nebot

Introduction to Fuzzy Computation

Introduction to Fuzzy Computation

- Fuzzy logic; Fuzzy sets; Fuzzy rules
- Fuzzy logic concepts (some more…)
- Fuzzy rule-based systems (Mamdani and Sugeno)
- Neuro-Fuzzy System
- Genetic Fuzzy Systems

* Planning and Approximate Reasoning course (PAR-MAI)

Fuzzy Logic vs. Fuzzy Arithmetic

- *Fuzzy logic* is the generalization of ordinary logic (Boolean) and allows truth values other than absolute truth and utter falsity
- *Fuzzy arithmetic* is a formalism for making calculations with numerical quantities that are imprecisely known. These quantities are known as fuzzy numbers and are defined as fuzzy sets on the real numbers
- Fuzzy logic is comparable to fuzzy arithmetic in the same way that logic is comparable to arithmetic. They don't have much in common in terms of the kinds of problems they're used to address

Fuzzy Logic (origins)

- ***Heraclitus*** (535-475 BC): things can be simultaneously *True* and *False*
- ***Plato*** (428-348 BC): there is a third region beyond *True* and *False*
- ***Łukasiewicz*** (1878-1956): many valued logic. First described three-valued logic, it was later generalized to *n*-valued (for all finite *n*)
- ***Knuth*** (1938): three-valued logic similar to *Łukasiewicz*. His insight, apparently missed by *Łukasiewicz*, was to use the integral range [-1, 0, +1] rather than [0, 1, 2]

Fuzzy Logic (origins)

Lotfi A. Zadeh

- was a mathematician, computer scientist, electrical engineer, and professor emeritus at the University of California, Berkeley
- in his paper fuzzy sets (1965), proposed using a membership function operating on the domain of all possible values and new operations for the calculus of logic and showed that fuzzy logic was a generalization of Boolean logic
- he also proposed fuzzy numbers as a special case of fuzzy sets, as well as the corresponding rules for consistent mathematical operations (fuzzy arithmetic) and introduces the extension principle concept

Fuzzy Logic Characteristics

- Represent vagueness and imprecision of statements in natural language
- Knowledge is interpreted as a collection of elastic/fuzzy constraints on a set of variables
- Inference is viewed as a process of propagation of elastic constraints
- Fuzzy logic allows decision making with estimated values under incomplete or uncertain information

Fuzzy Set Definition (Type-1)

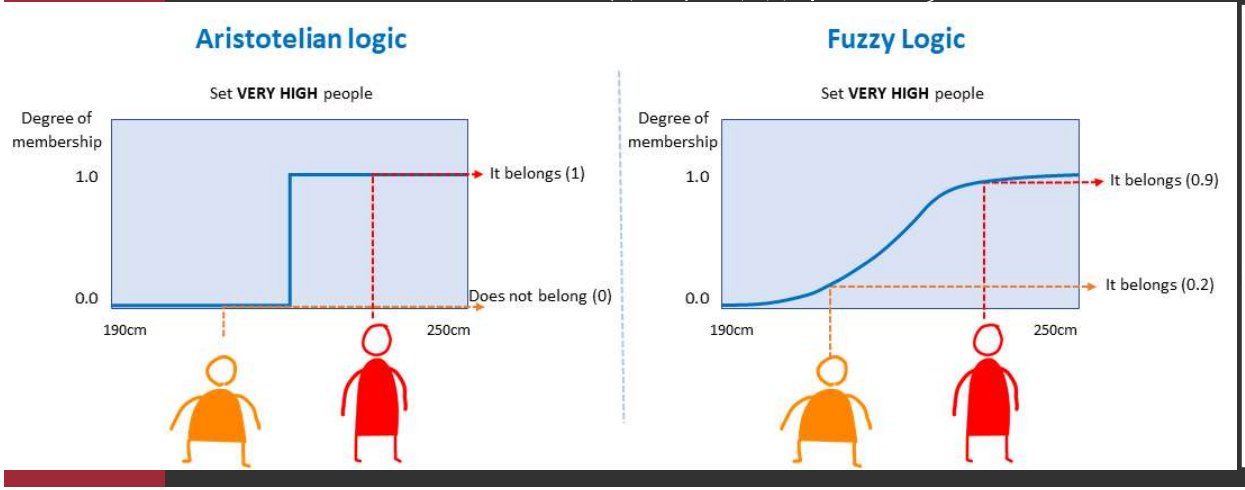
Let X be a nonempty set. A fuzzy set A in X is characterized by its membership function

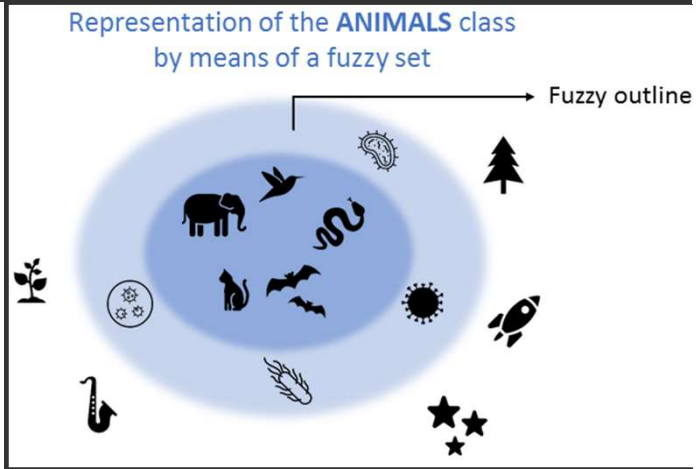
$$\mu_A : X \rightarrow [0, 1]$$

and $\mu_A(x)$ is interpreted as the degree of membership of element x in the fuzzy set A for each $x \in X$.

It is clear that *A* is completely determined by the set of ordered pairs:

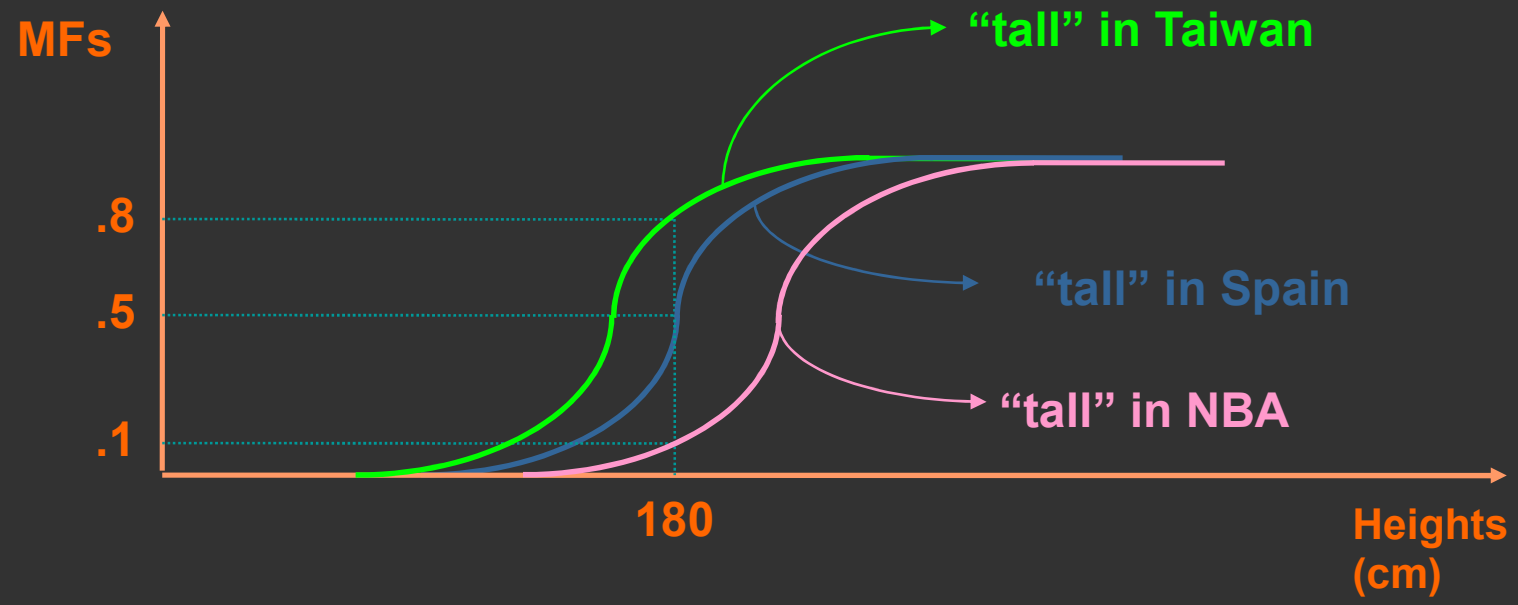
$$A = \{(x, \mu_A(x)) | x \in X\}$$



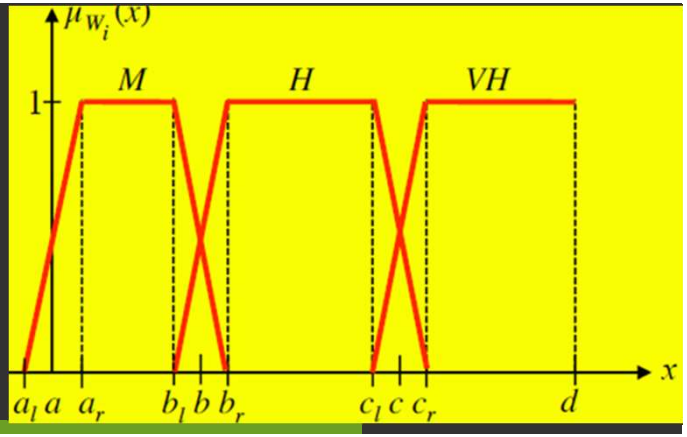


Membership Functions

- About MFs
  - Subjective measures (MFs represent distributions of possibility rather than probability)
  - Not probability functions



Fuzzy Sets: Types



Thus far we introduced only one type of fuzzy set! First-order FS or Ordinary FS or Type-1 FS

Given a relevant universal set *X*, any arbitrary fuzzy set of this type (set *A*) is defined by a function of the form:

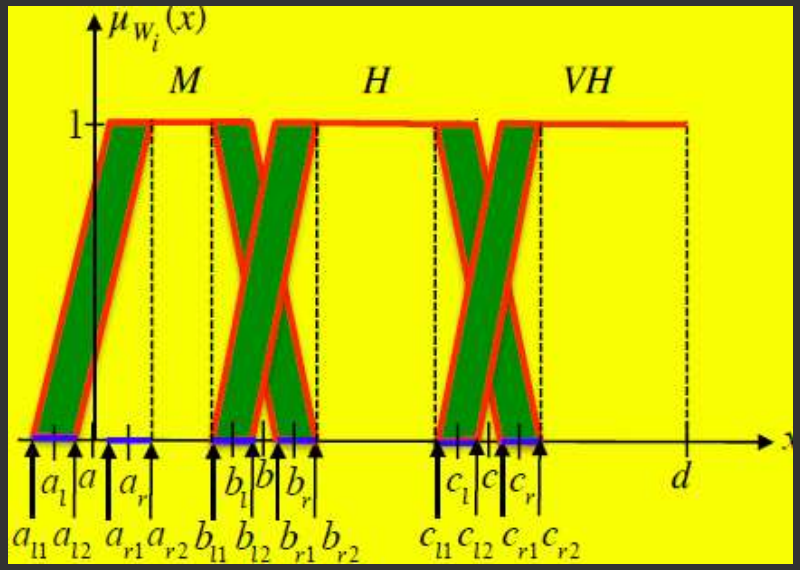
$$A : X \rightarrow [0, 1]$$

Ordinary Fuzzy Sets

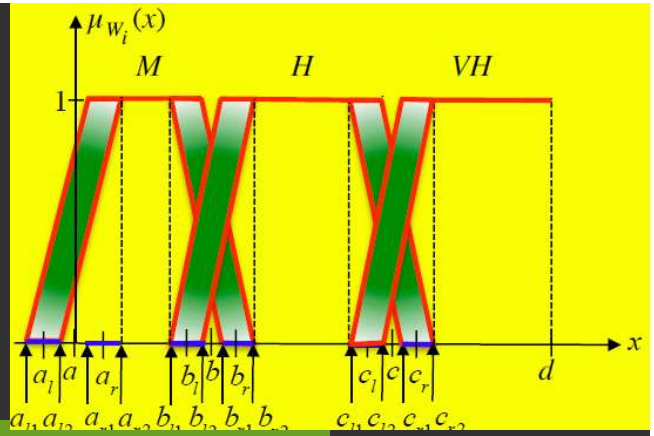
- By far the most common in the literature and applications
- Their membership functions are often overly precise, i.e. they require that each element of the universal set be assigned a particular real number (degree of belonging to the set)

Fuzzy Sets: Types (Type-2)

- A second-order uncertainty partition of the real-variable, x, partitions it into overlapping intervals, where one is unsure about where the overlap begins and ends, so that the degree of membership in each region of overlap is an interval of real numbers that is a subset of x. **Interval-valued fuzzy sets**



Fuzzy Sets: Types



Interval-valued fuzzy sets can further be generalized by allowing their intervals to be fuzzy.

Fuzzy sets of **type 2**

Fuzzy power set: Set of all ordinary fuzzy sets that can be defined within the universal set [0, 1]

$$A : X \rightarrow \mathcal{F}([0,1])$$

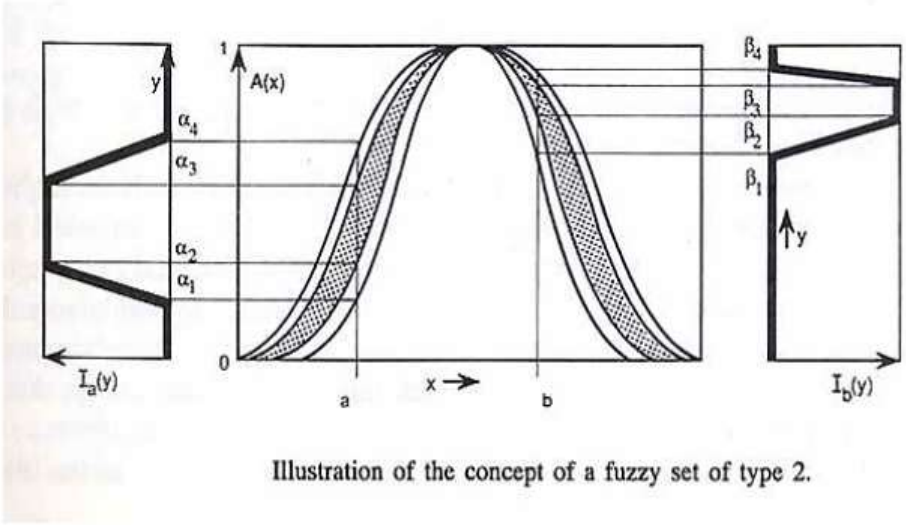
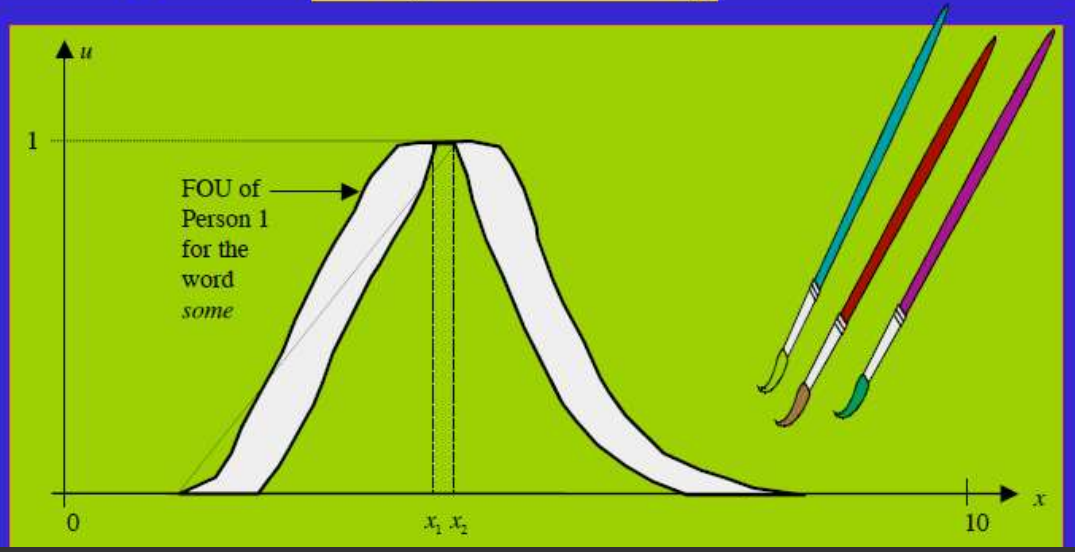


Illustration of the concept of a fuzzy set of type 2.

Types 3, 4, etc...

Examples Type-2 Fuzzy Set

- ***Intra-uncertainty*** **about a word can be modeled using a type-2** ***person fuzzy set*** $\tilde{A}(p_j), j = 1, \ldots, n_{\tilde{A}}$

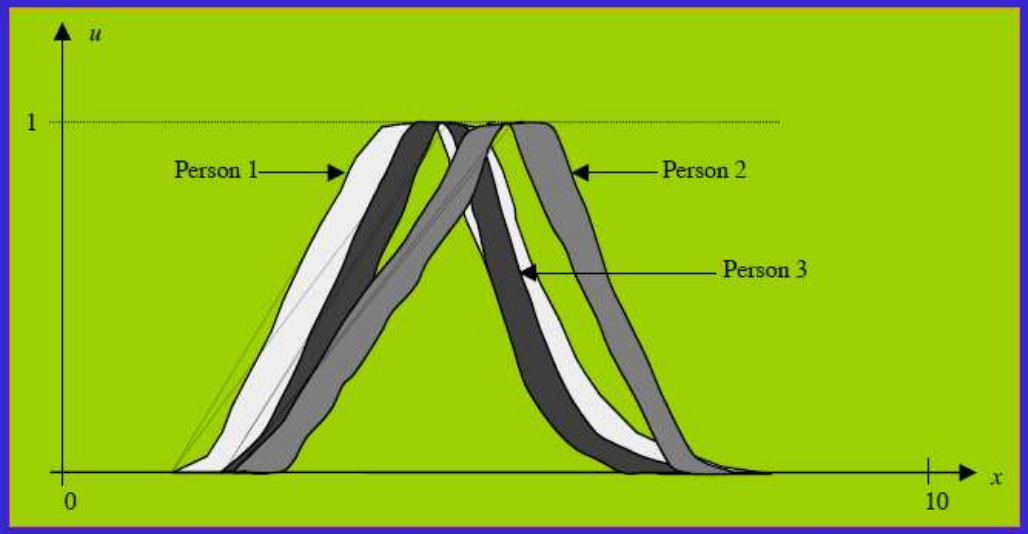


FOU: footprint of uncertainty

From J.M. Mendel
IEEE-FUZZ

Examples Type-2 Fuzzy Set

- **Collect person MFs from a group of people**



From J.M. Mendel
IEEE-FUZZ

A specific application may only involve teenage girls, beer-drinking men, naturalists, bikers, etc. It is important then to collect person MFs from members of the representative group.

Extension Principle

Suppose that f is a function from X to Y, and A is a fuzzy set on X defined as:

$$A = \{(x, \mu_A(x))\} = \left\{\left(x_1, \mu_A(x_1)\right), \left(x_2, \mu_A(x_2)\right), \cdots, \left(x_n, \mu_A(x_n)\right)\right\}$$

Then the extension principle states that the image of the fuzzy set A under the mapping of f can be expressed as a fuzzy set $B \subseteq Y$.

$$B = f(A) = \{(y, \mu_B(y)), \text{ where } \mu_B(y) = sup\{\mu_A(x) | x \in X, y = f(x)\}$$

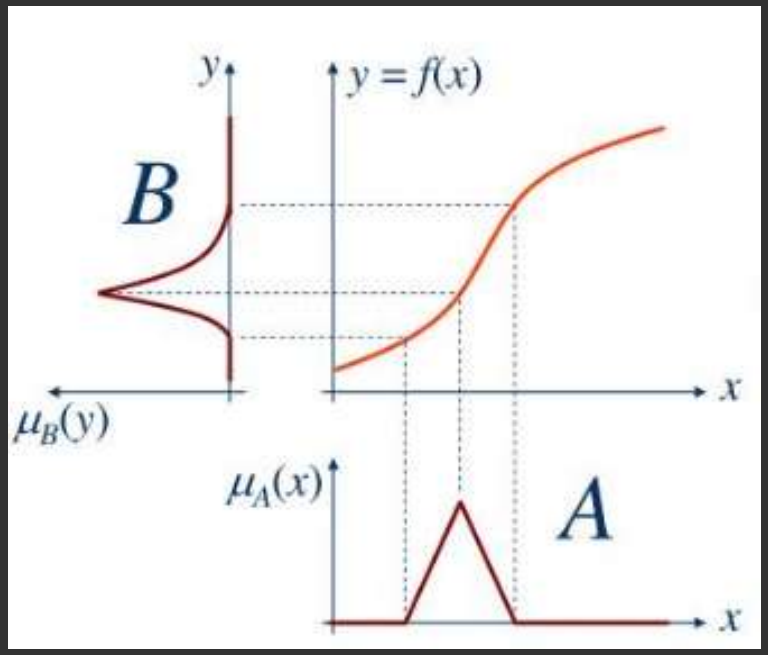
The supremum, sup, function applies if there are two or more values of X that have the same value in f.

The *extension principle* describes how to extend a classical function to a fuzzy relation

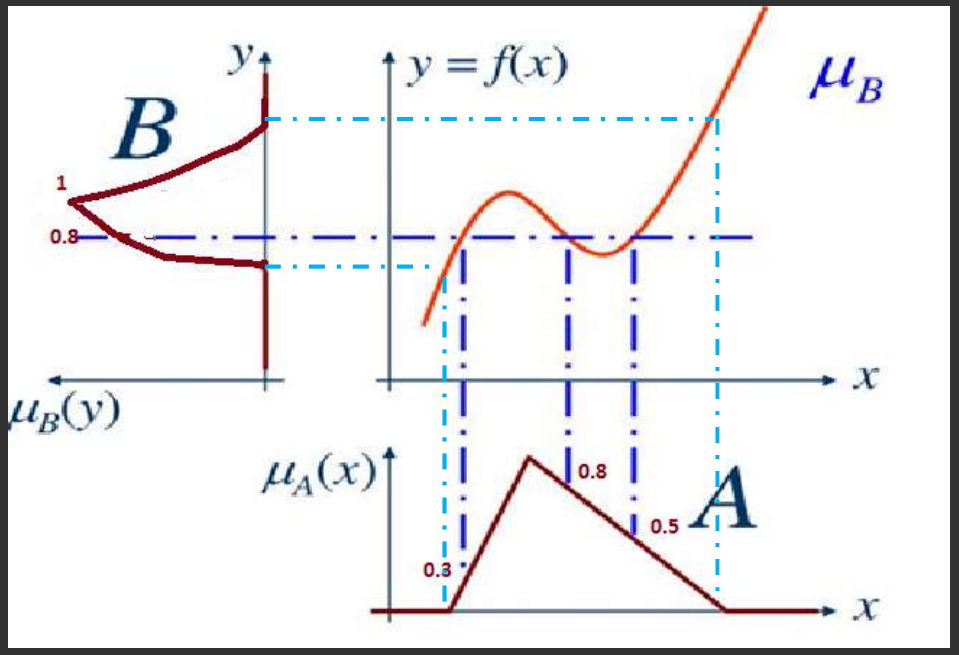
Extension Principle

- The extension principle generalizes an ordinary mapping of a function to a mapping between fuzzy sets.

monotonic function



non-monotonic function



Some Misconceptions on Fuzzy Logic

Fuzzy logic is the same as imprecise logic

Fuzzy logic is not any less precise than any other form of logic: it is an organized and mathematical method of handling *inherently* imprecise concepts.

Some Misconceptions on Fuzzy Logic

Fuzzy logic is a new way of expressing probability

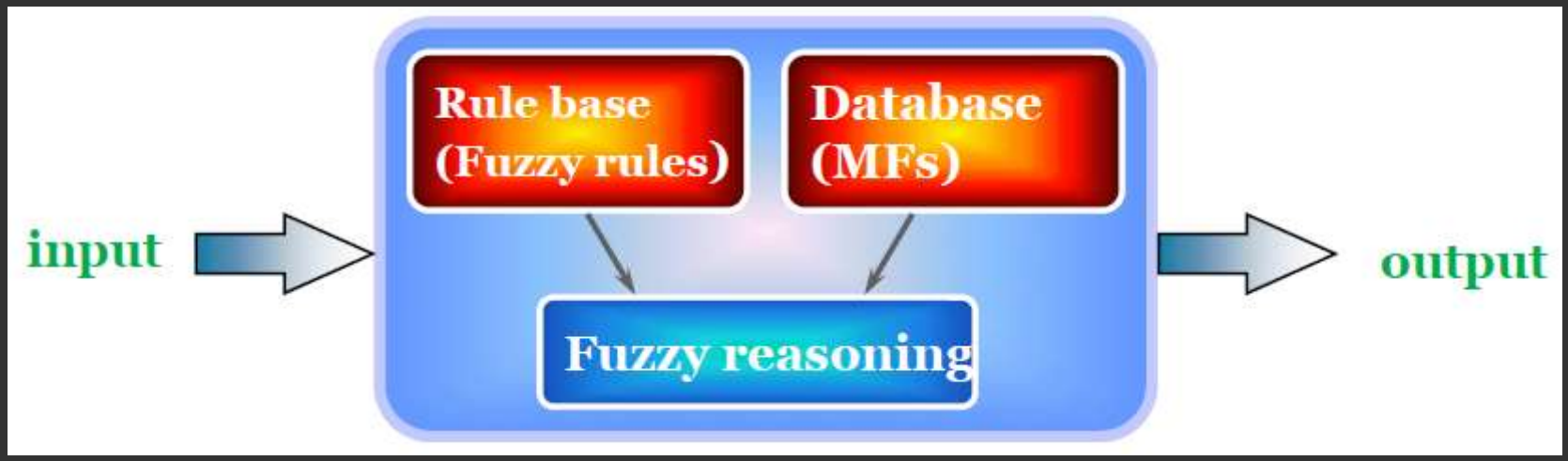
Fuzzy logic and probability refer to different kinds of uncertainty. Fuzzy logic is specifically designed to deal with **imprecision** of facts (fuzzy logic statements), while probability deals with **chances** of that happening *(but still considering the result to be precise)*

Example:

- P(A) = 0.5 means that A may be true or may be false
- A membership value of 0.5 means both true and false at the same time

⇨ However, still is a point of controversy

Fuzzy Rule-Base Systems



Fuzzy Rule-Based Systems

Principal FRBS for engineering problems (i.e., dealing with real-valued inputs and outputs):

- *Mamdani*
- *Takagi-Sugeno-Kang*

Fuzzy Rules

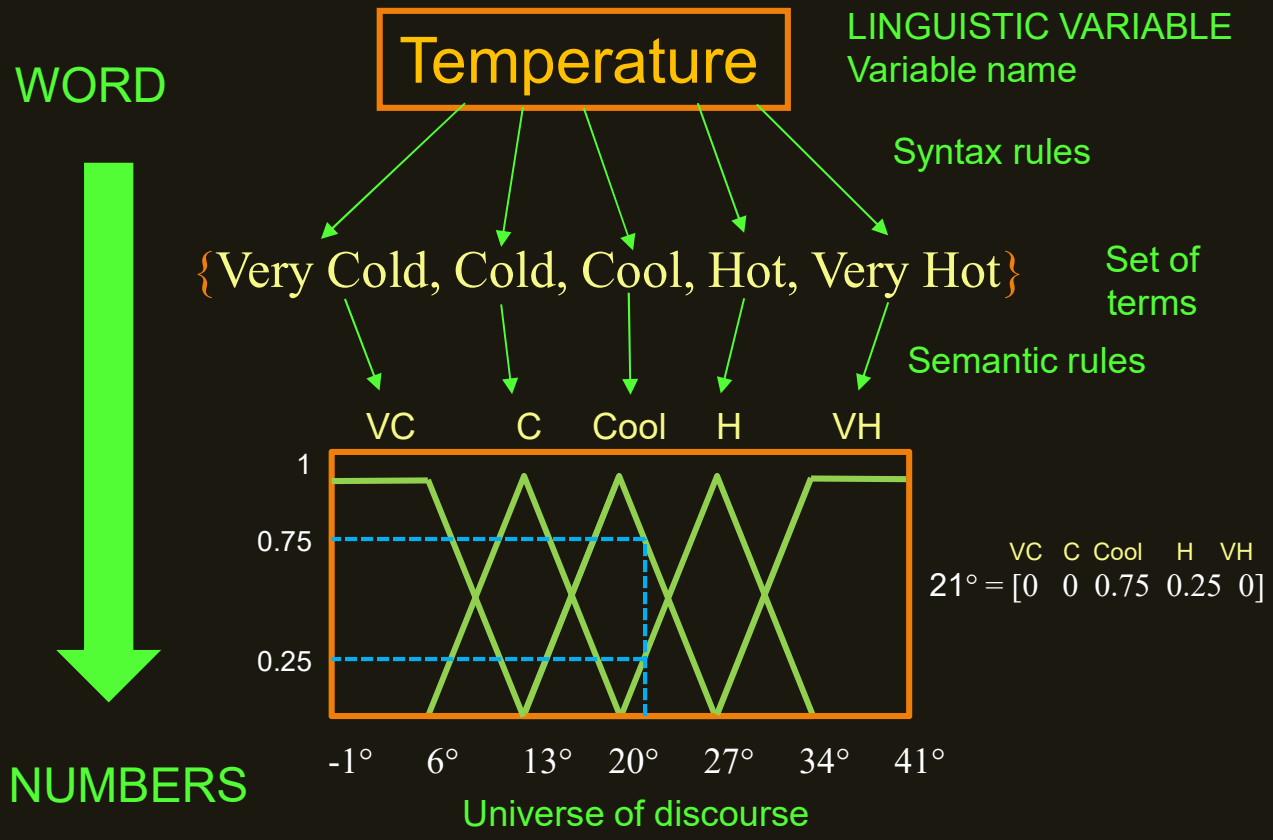
- A *fuzzy rule* is a conditional statement in the familiar form:

  IF x is A

  THEN y is B

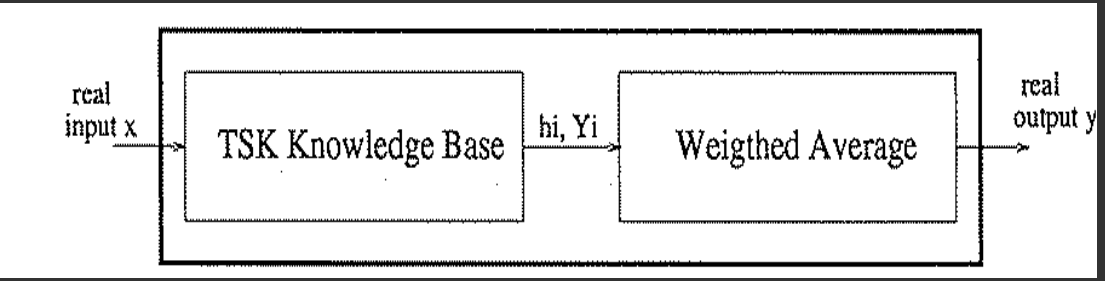
  - x and y are linguistic variables
  - A and B are linguistic values determined by fuzzy sets on the universe of discourses X and Y, respectively.

Linguistic Variable = <X, T(X), U, G, M>



Fuzzy Inference: TSK

- Takagi-Sugeno-Kang (1985), the TSK method for fuzzy inference is:



$$IF\ X_1\ is\ A_1\ and\ \ldots\ and\ X_n\ is\ A_n$$
$$THEN\ Y = p_1 \cdot X_1 + \cdots + p_n \cdot X_n + p_0,$$

Polynomial function

Output TSK (m rules): $\frac{\sum_{i=1}^{m} h_i \cdot Y_i}{\sum_{i=1}^{m} h_i}$

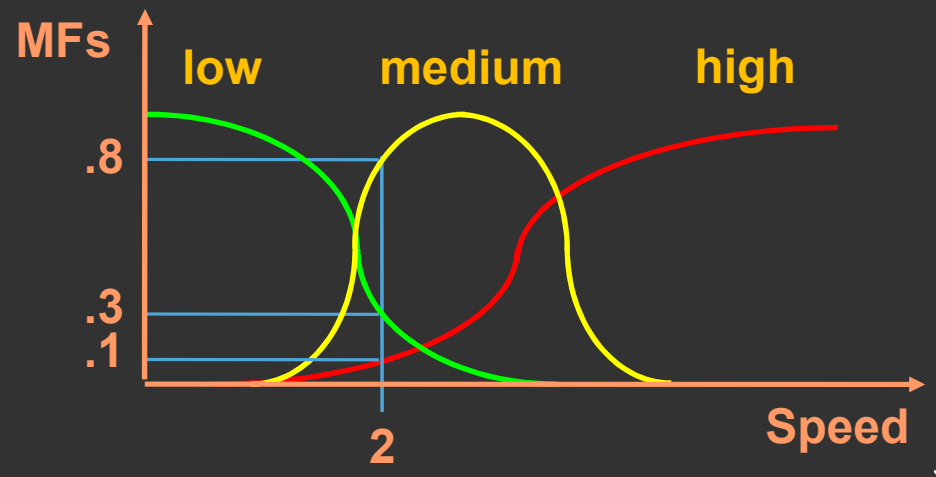
$$h_i = T(\mu_{A_{i1}}(x_1), \ldots, \mu_{A_{in}}(x_n))$$

Matching degree between the antecedent part of the i-th rule and the current inputs to the system. T-norm = conjunctive operators (minimum, algebraic product)

Zero-Order TSK FRBS

Takagi-Sugeno-Kang

If speed is low then resistance = 2
If speed is medium then resistance = 4
If speed is high then resistance = 8



Rule 1: h1 = .3; Y1 = 2
Rule 2: h2 = .8; Y2 = 4
Rule 3: h3 = .1; Y3 = 8

Resistance = Σ(hi*Yi) / Σhi
= (0.3*2+0.8*4+0.1*8)/(0.3+0.8+0.1)
= 3.83

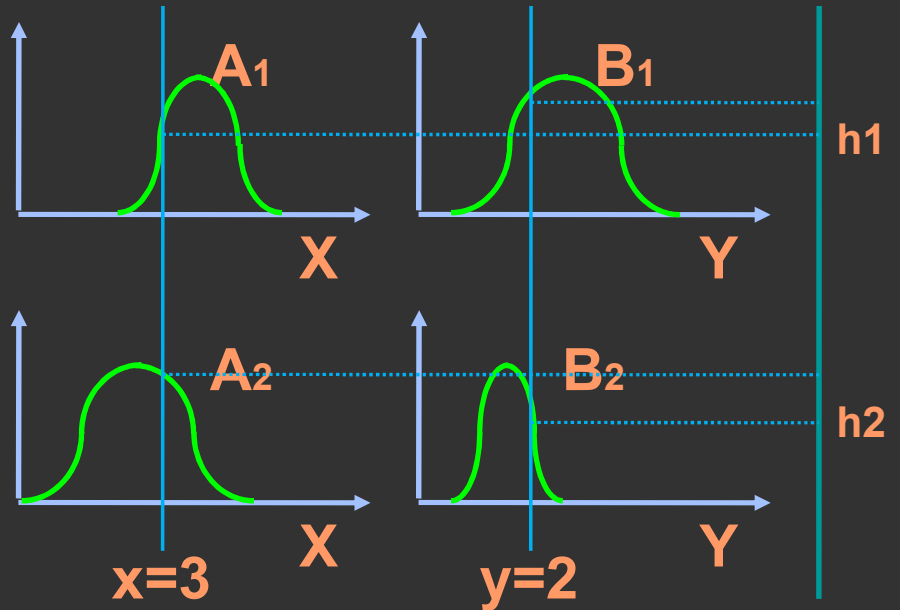
First-Order TSK FRBS

Takagi-Sugeno-Kang

Rule base

If X is A1 and Y is B1 then Z = p1*x + q1*y + r1
If X is A2 and Y is B2 then Z = p2*x + q2*y + r2

Fuzzy reasoning



$Z_1 = p_1*3+q_1*2+r_1$

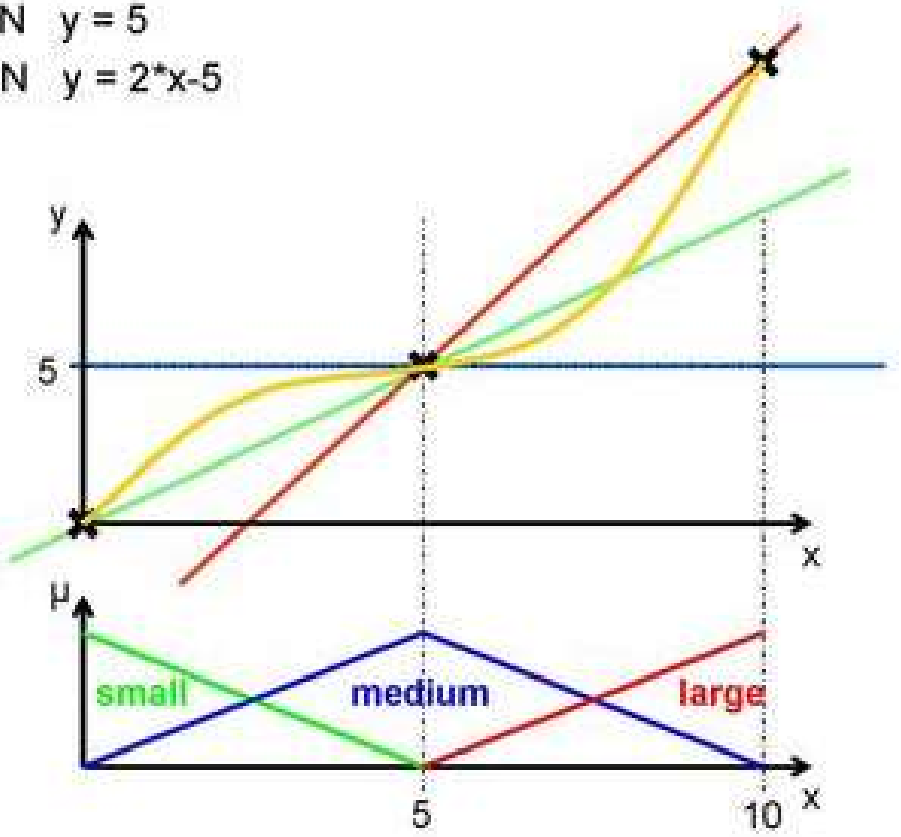
$Z_2 = p_2*3+q_2*2+r_2$

$$Z = \frac{h_1*Z_1+h_2*Z_2}{h_1+h_2}$$

TSK: Visual example

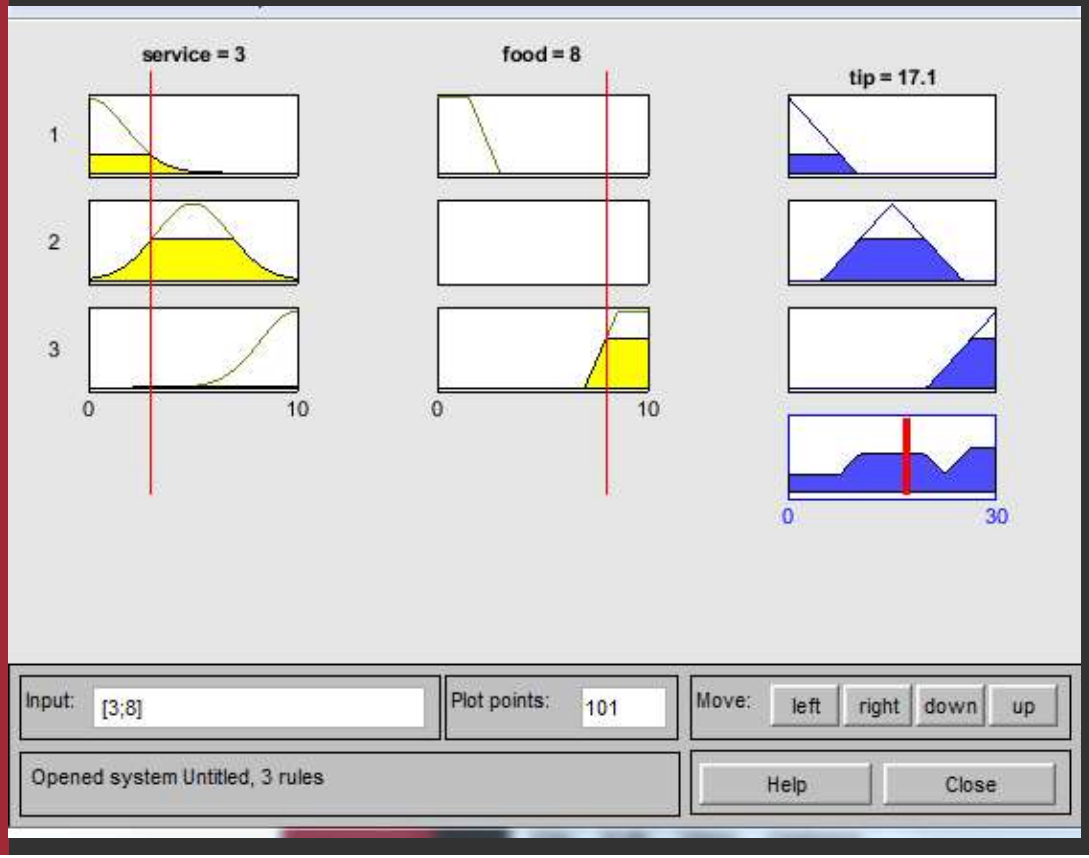
R1: IF x IS small THEN y = x
R2: IF x IS medium THEN y = 5
R3: IF x IS large THEN y = 2*x-5

$$y = \frac{\sum_{i=1}^{r} \mu_{R_i} \cdot y_i(\vec{x})}{\sum_{i=1}^{r} \mu_{R_i}}$$



Example: Dinner for two (Mamdani)

Rule Viewer



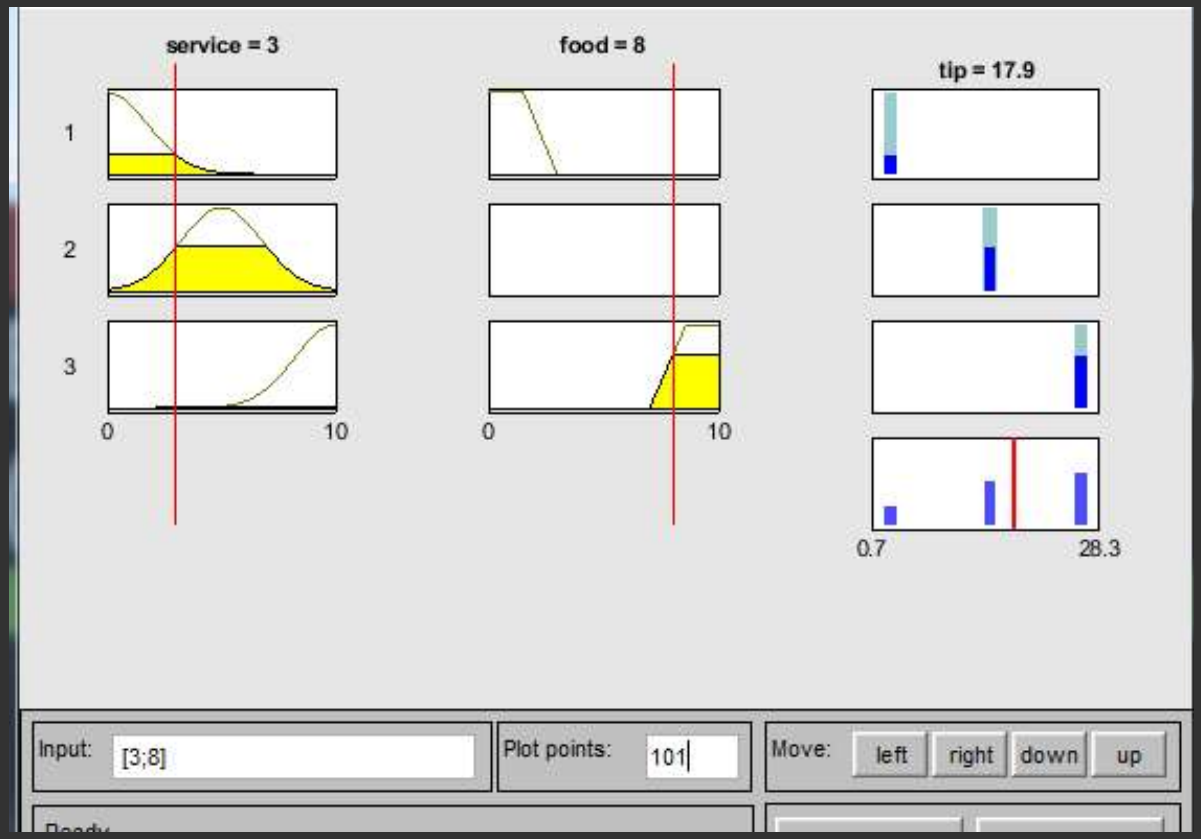
IF *service* is *poor*
OR *food* is *bad*
THEN *tip* is *cheap*

IF *service* is *good*
THEN *tip* is *average*

IF *service* is *excellent*
OR *food* is *delicious*
THEN *tip* is *generous*

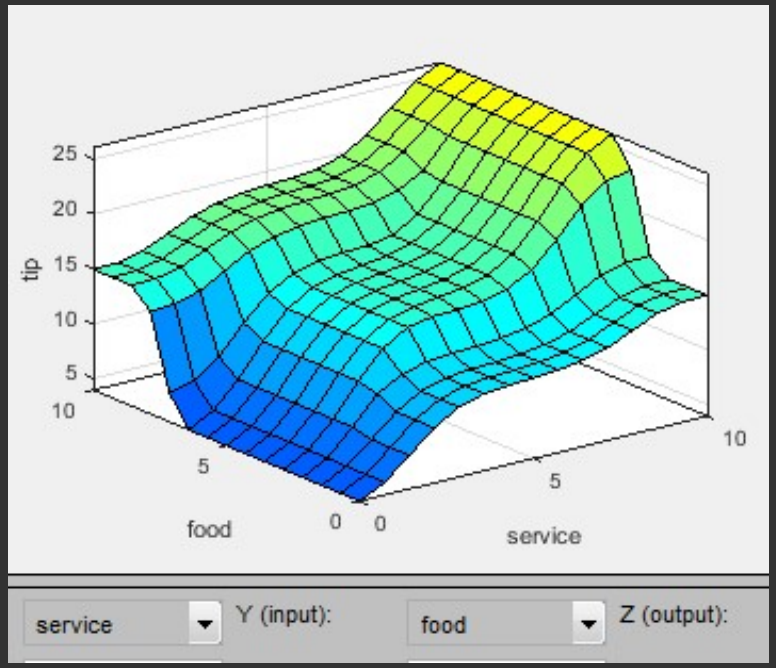
Example: Dinner for two (Sugeno)

Rule Viewer

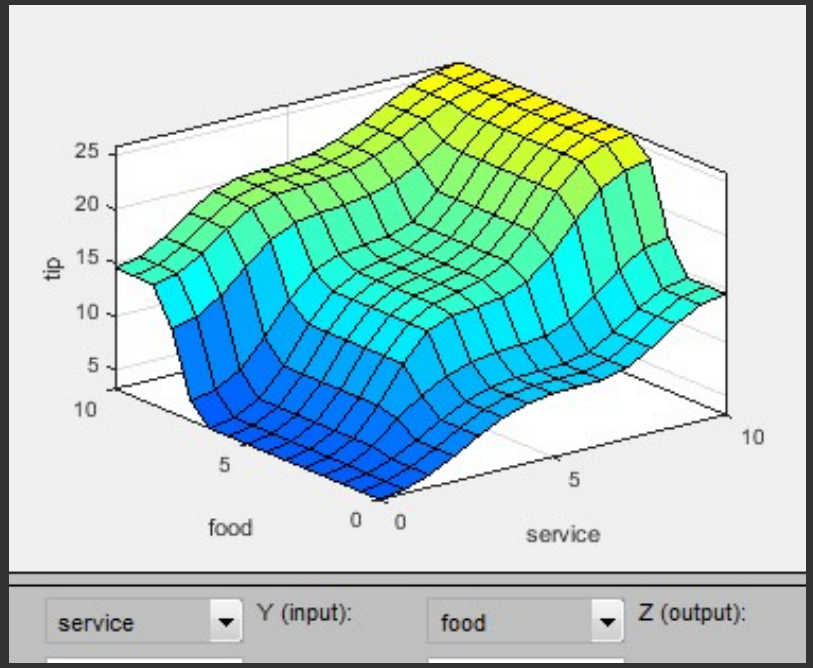


Example: Dinner for two

Mamdani



Sugeno



Demo TSK

Fuzzy Juggling ball system

Matlab: juggler

Mamdani vs. TSK

Advantages of the TSK Method

- It is computationally efficient
- It works well with linear techniques (e.g., PID control)
- It works well with optimization and adaptive techniques
- It has guaranteed continuity of the output surface
- It is well suited to mathematical analysis

Advantages of the Mamdani Method

- It is intuitive
- It has widespread acceptance
- It is well suited to human input

Applications of Fuzzy Logic

- Why use *fuzzy systems*?
  - Apply fuzziness (*and therefore accuracy*) to linguistically defined terms and rules
  - Lack of crisp or concrete mathematical models exist
- When do you avoid *fuzzy systems*?
  - Traditional approaches produce acceptable results
  - Crisp or concrete mathematical models exist and are easily implemented

Applications of Fuzzy Logic

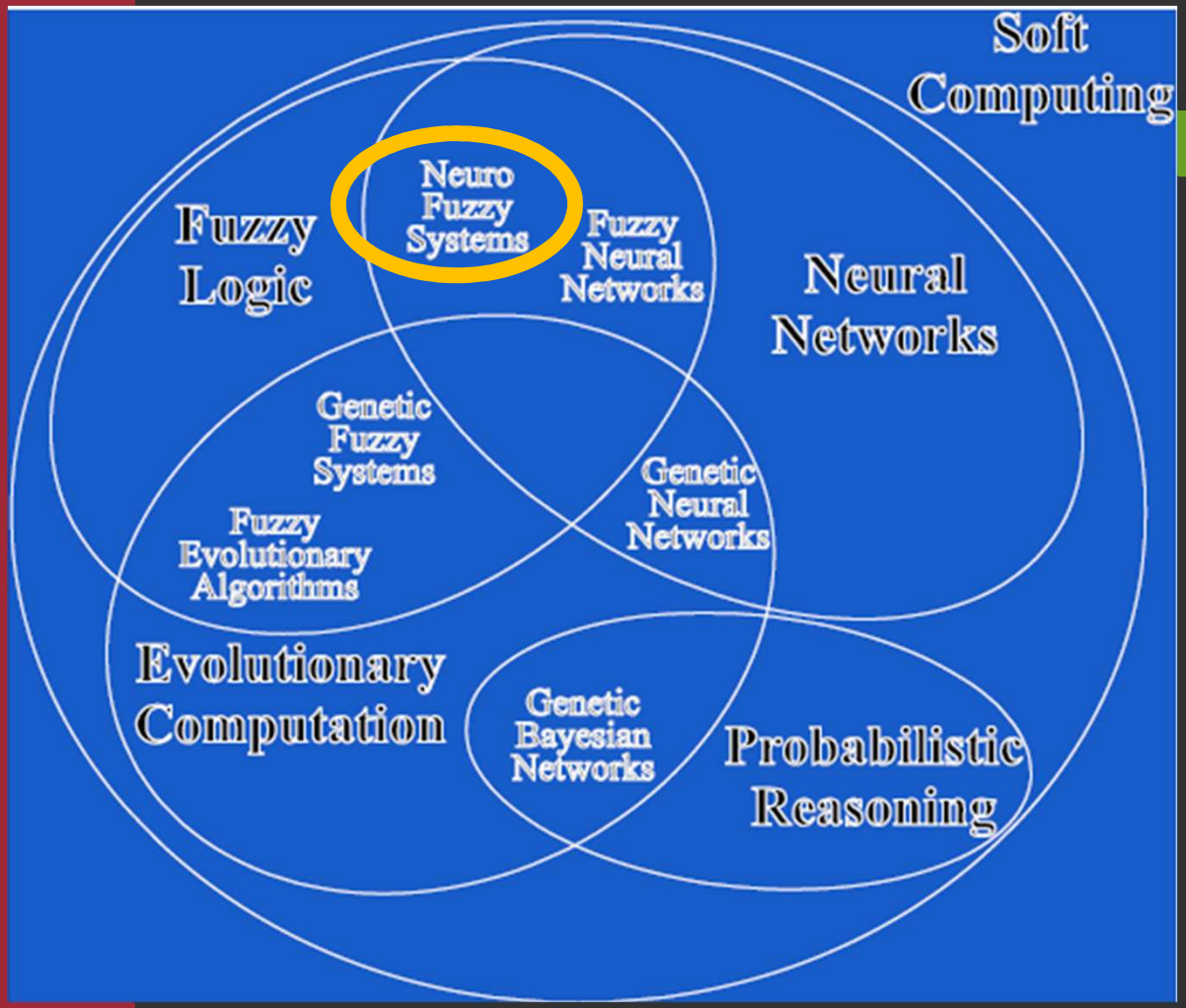
Application Areas of Fuzzy Logic

The Blow given table shows how famous companies using fuzzy logic in their products.

Product	Company	Fuzzy Logic
Anti-lock brakes	Nissan	Use fuzzy logic to controls brakes in hazardous cases depend on car speed, acceleration, wheel speed, and acceleration
Auto transmission	NOK/Nissan	Fuzzy logic is used to control the fuel injection and ignition based on throttle setting, cooling water temperature, RPM, etc.
Auto engine	Honda, Nissan	Use to select geat based on engine load, driving style, and road conditions.
Copy machine	Canon	Using for adjusting drum voltage based on picture density, humidity, and temperature.
Cruise control	Nissan, Isuzu, Mitsubishi	Use it to adjusts throttle setting to set car speed and acceleration
Dishwasher	Matsushita	Use for adjusting the cleaning cycle, rinse and wash strategies based depend upon the number of dishes and the amount of food served on the dishes.
Elevator control	Fujitec, Mitsubishi Electric, Toshiba	Use it to reduce waiting for time-based on passenger traffic
Golf diagnostic system	Maruman Golf	Selects golf club based on golfer's swing and physique.
Fitness management	Omron	Fuzzy rules implied by them to check the fitness of their employees.
Kiln control	Nippon Steel	Mixes cement
Microwave oven	Mitsubishi Chemical	Sets lunes power and cooking strategy
Palmtop computer	Hitachi, Sharp, Sanyo, Toshiba	Recognizes handwritten Kanji characters
Plasma etching	Mitsubishi Electric	Sets etch time and strategy

Sendai subway (1987)
Control speed:
smoothness and
10% energy efficient

Hybridization: Neuro Fuzzy Systems

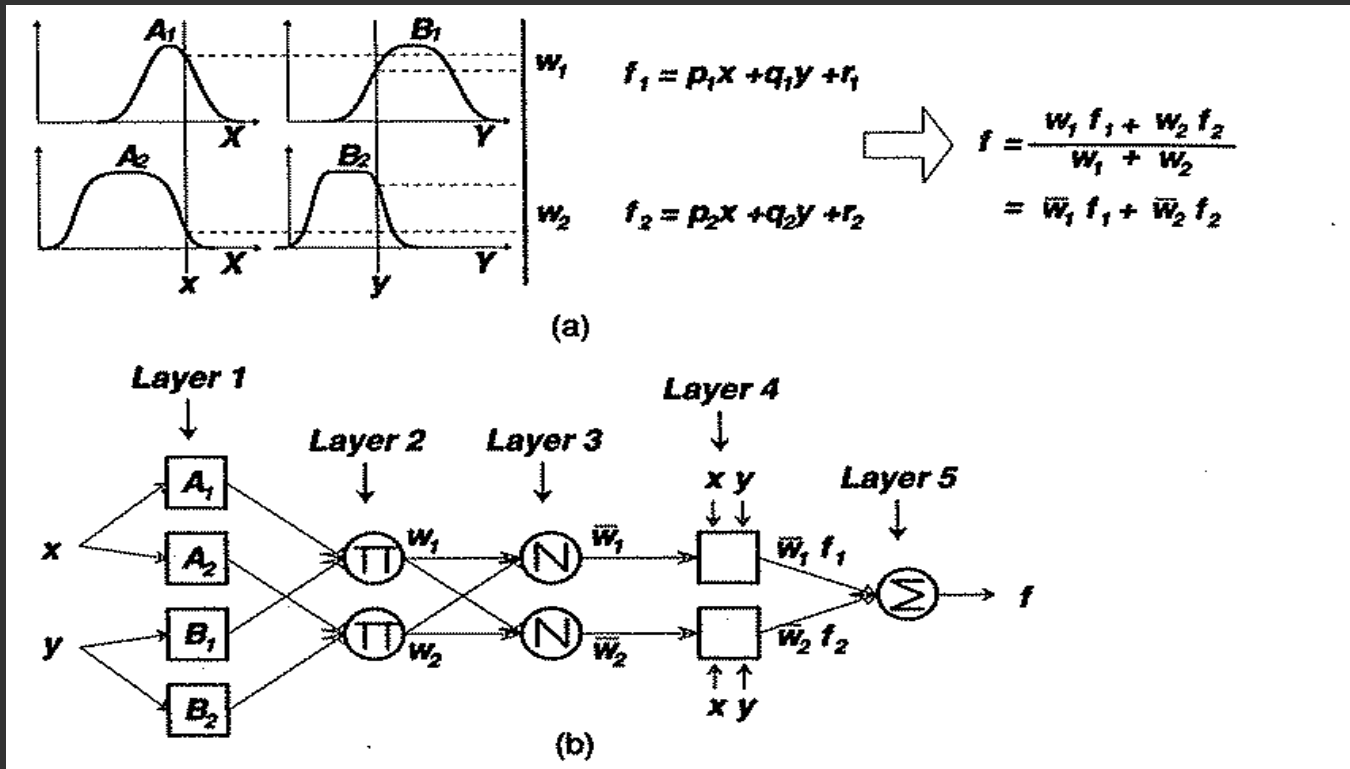


- FSs are neither capable of learning, adaptation or parallel computation, whereas these characteristics are clearly attributed to NNs.
- NNs lack flexibility, human interaction or knowledge representation, which lies at the core of FL.

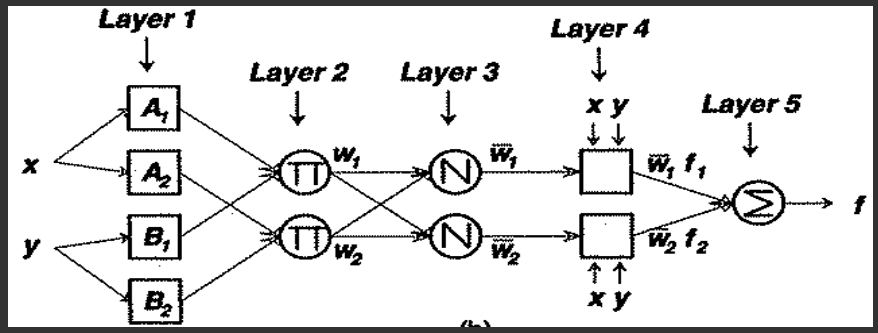
ANFIS: Adaptive Neuro-Fuzzy Inference System

Sugeno

Rule 1: If x is A_1 and y is B_1, then $f_1 = p_1x + q_1y + r_1$,
Rule 2: If x is A_2 and y is B_2, then $f_2 = p_2x + q_2y + r_2$.



ANFIS: Architecture



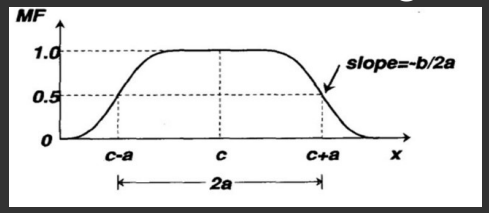
Layer 1: Every node *i* is an adaptive node with a node function:

$$O_{1,i} = \mu_{A_i}(x), \quad \text{for } i = 1, 2, \text{ or}$$
$$O_{1,i} = \mu_{B_{i-2}}(y), \quad \text{for } i = 3, 4,$$

where *x* (or *y*) is the input to node *i* and A_i (or B_{i-2}) is a linguistic label ("small", "large",…) associated with this node

The membership function for *A* or B (=A_1, A_2, B_1 or B_2) can be any appropriate parameterized membership function such as the generalized bell function:

$$\mu_A(x) = \frac{1}{1 + \left|\frac{x - c_i}{a_i}\right|^{2b}},$$

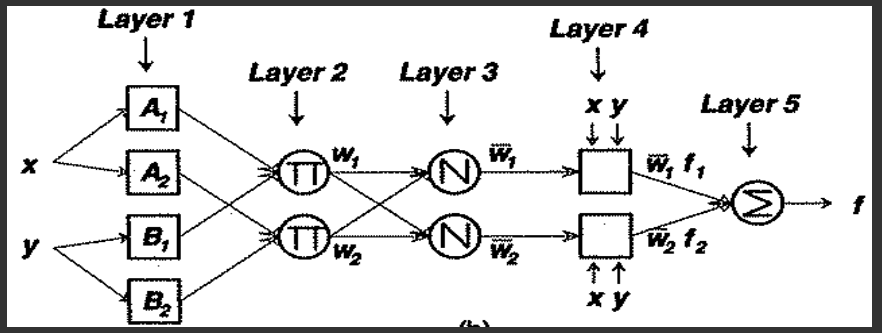


where $\{a_i, b_i, c_i\}$ is the parameter set (**premise parameters**)

Layer 2: Every node in this layer is a fixed node labeled Π, whose output is the result of applying any other T-norm operator that performs fuzzy AND

$$O_{2,i} = w_i = \mu_{A_i}(x)\mu_{B_i}(y), \quad i = 1, 2.$$

ANFIS: Architecture



Layer 3: Every node in this layer is a fixed node labeled N. The i^{th} node calculates the ratio of the i^{th} rule's firing strength to the sum of all rules' firing strengths

$$O_{3,i} = \overline{w}_i = \frac{w_i}{w_1 + w_2}, \quad i = 1, 2.$$

Outputs of this layer are called **normalized firing strengths**

Layer 4: Every node in this layer is an adaptive node with a node function

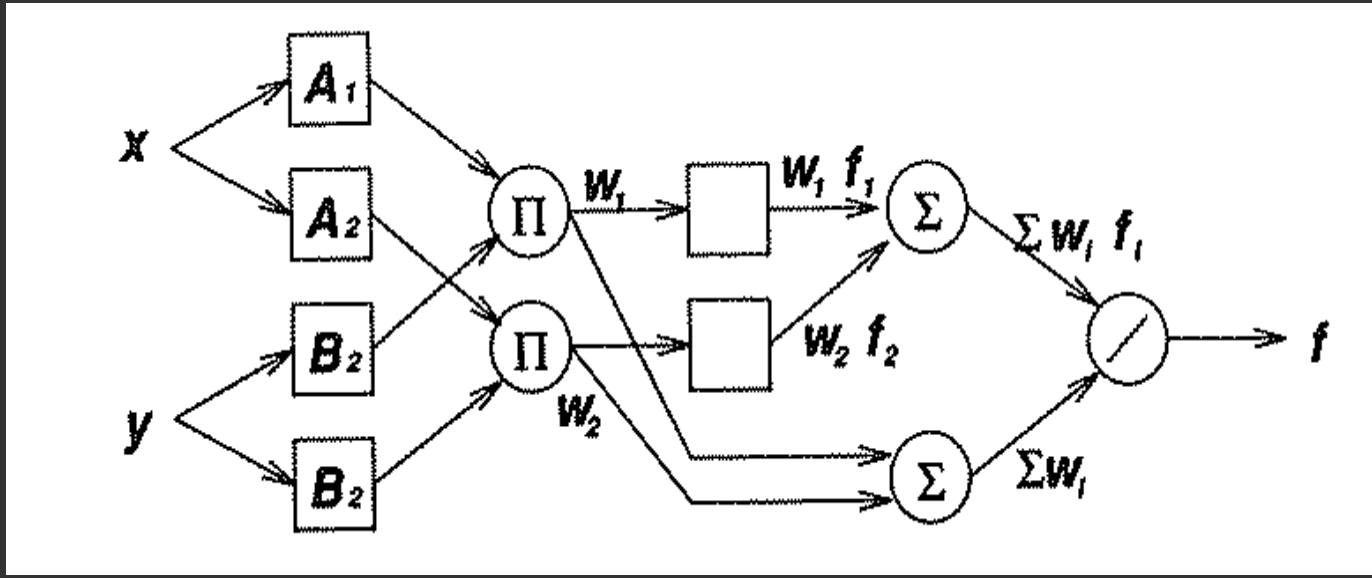
$$O_{4,i} = \overline{w}_i f_i = \overline{w}_i (p_i x + q_i y + r_i),$$

where $\overline{w}_i$ is a normalized firing strength from layer 3 and $\{p_i, q_i, r_i\}$ is the parameter set of this node (**consequent parameters**)

Layer 5: The single node in this layer is a fixed node labeled Σ, which computes the overall output as the summation of all incoming signals:

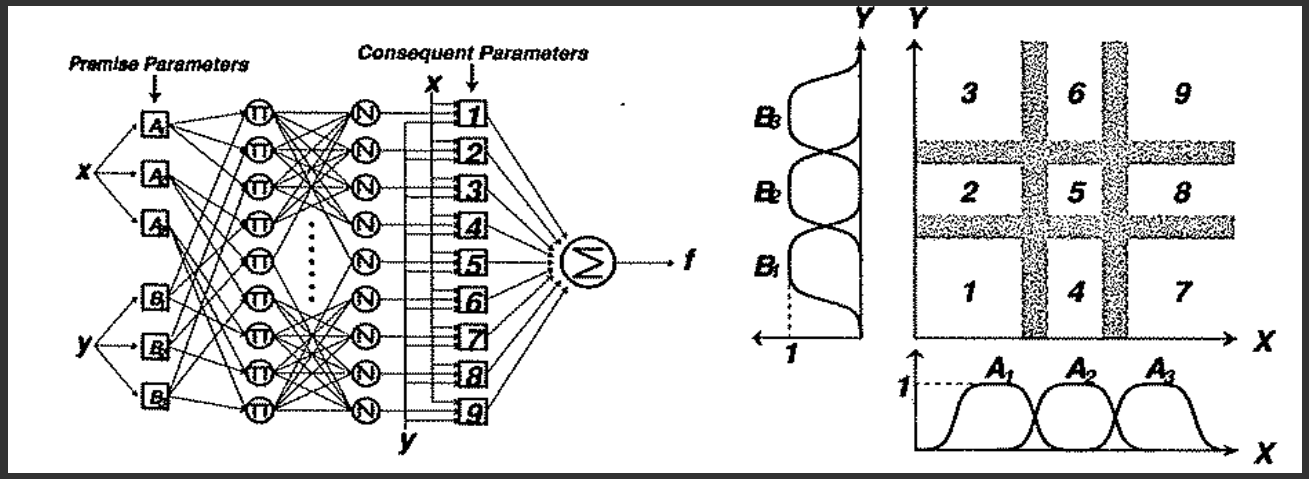
$$O_{5,1} = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

ANFIS: Another Architecture



Weight normalization is performed at the very last layer

ANFIS: Hybrid Learning



The premise part of a rule defines a fuzzy region, while the consequent part specifies the output within the region

	Forward pass	Backward pass
Premise parameters	Fixed	Gradient descent
Consequent parameters	Least-squares estimator	Fixed

S1= Set of premise (nonlinear) parameters
S2= Set of consequent (linear) parameters

Example 1: Two-Input Sinc Function

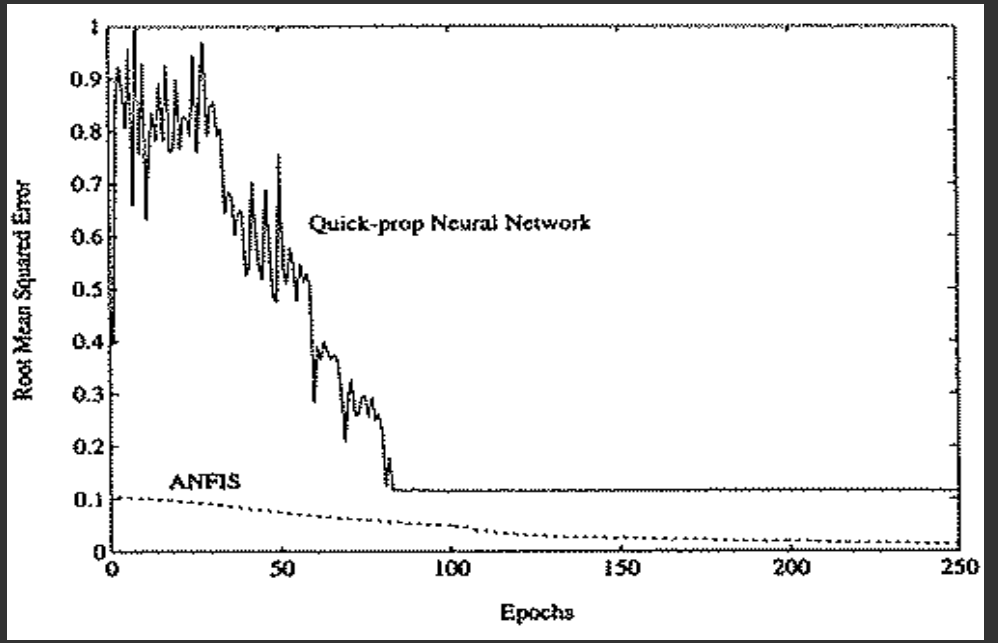
$$z = \mathrm{sinc}(x, y) = \frac{\sin(x)\sin(y)}{xy}.$$

ANFIS

- 121 training data pairs
- 16 rules with 4 membership functions assigned to each input
- 72 total fitting parameters (24 premise and 48 consequent)

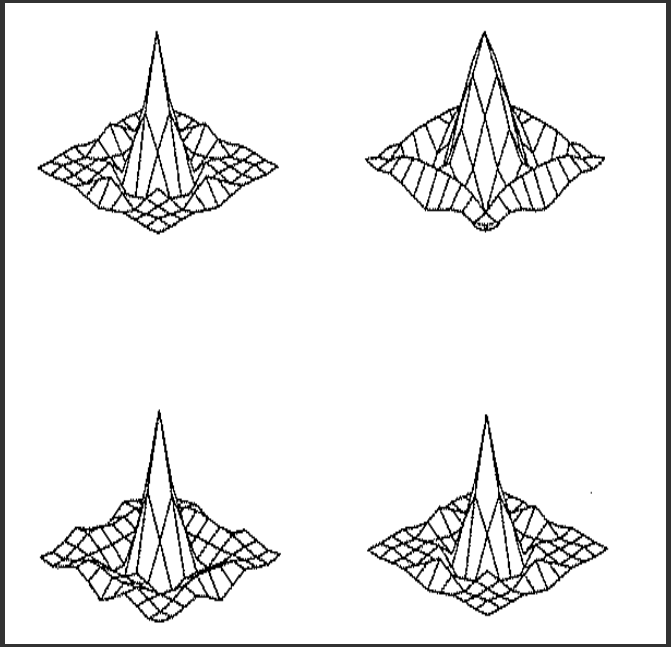
2-18-1 Backpropagation MLP

- 121 training data pairs
- Trained with quick propagation
- 73 total fitting parameters (connection weights and thresholds)

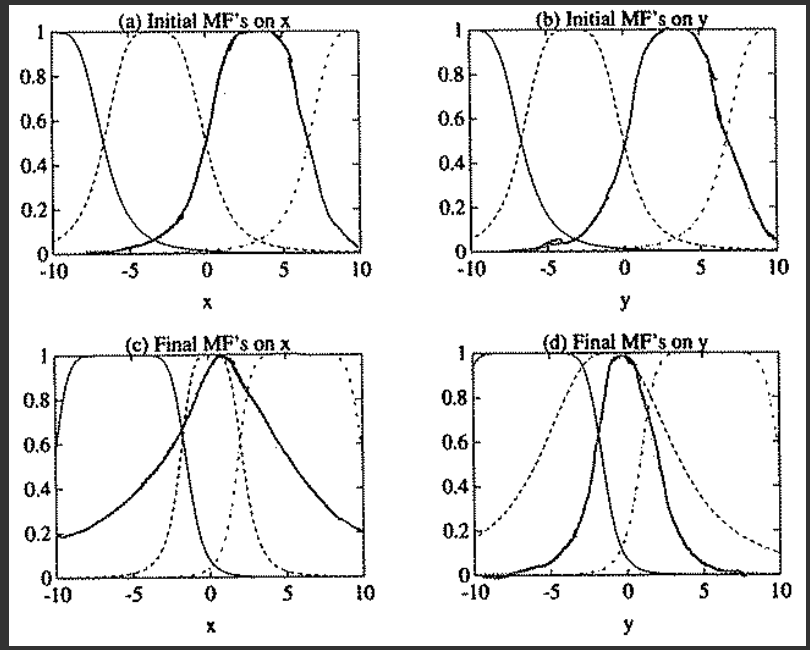


ANFIS approximates the highly nonlinear surface more effectively than the MLP

Example 1: Two-Input Sinc Function



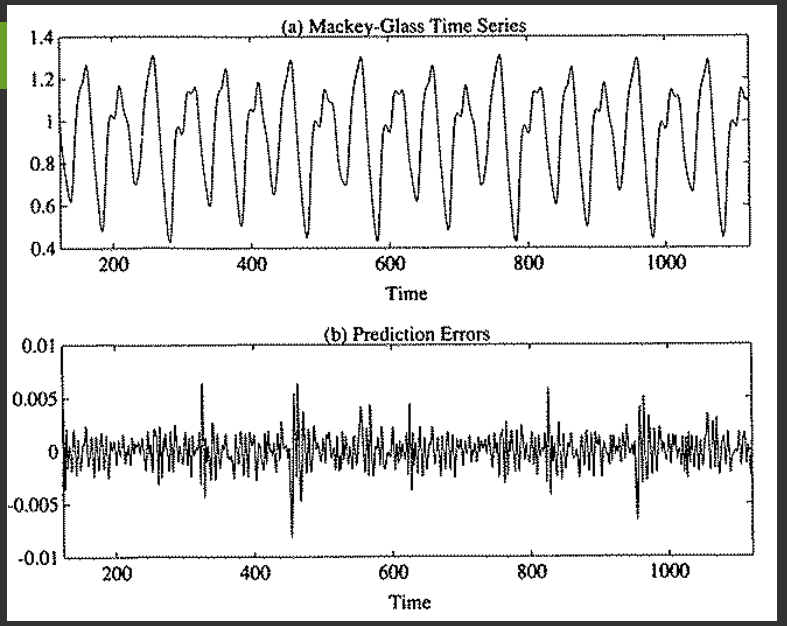
- Training data and reconstructed surfaces at different epochs: 0.5, 99.5 and 249.5



- Initial and final membership functions

Example 2: Predicting Mackey-Glass Chaotic Time Series

- Desired and predicted values for both training and test data are essentially the same



NDEI: root mean square error divided by the standard deviation of the target series; Prediction: 500 values test data set

Method	Training cases	Non-dimensional error index
ANFIS	500	0.007
AR model	500	0.19
Cascaded-correlation NN	500	0.06
Backpropagation MLP	500	0.02
6th-order polynomial	500	0.04
Linear predictive method	2000	0.55

PARAM (104)

(693)
(540)

Advantages of ANFIS

The remarkable generalization capability of ANFIS is due to:

- ANFIS can achieve a highly nonlinear mapping, far superior to MLP and common methods of similar complexity
- The initial parameters are intuitively reasonable and all the input space is covered properly (fast convergence)
- ANFIS requires fewer adjustable parameters than those required in other Neural Network structures and, specifically, backpropagation MLPs
- ANFIS consists of fuzzy rules that are local mappings instead of global ones, then facilitate the **minimal disturbance principle** (the adaptation should not only reduce the output error for the current training pattern but also minimize disturbance to response already learned)

Disadvantages of ANFIS

- ANFIS only deals with **parameter identification**, we still need methods for **structure identification** to determine an initial ANFIS architecture
- High number of rules when the number of inputs is high, reducing the interpretability
- Curse of dimensionality and high computational cost (but allows parallelization)

Subtractive Clustering

- **Clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense) to each other than to those in other groups (clusters)

- Subtractive clustering is a fast **one-pass algorithm** for estimating the number of clusters and the cluster centers in a data set

Subtractive Clustering

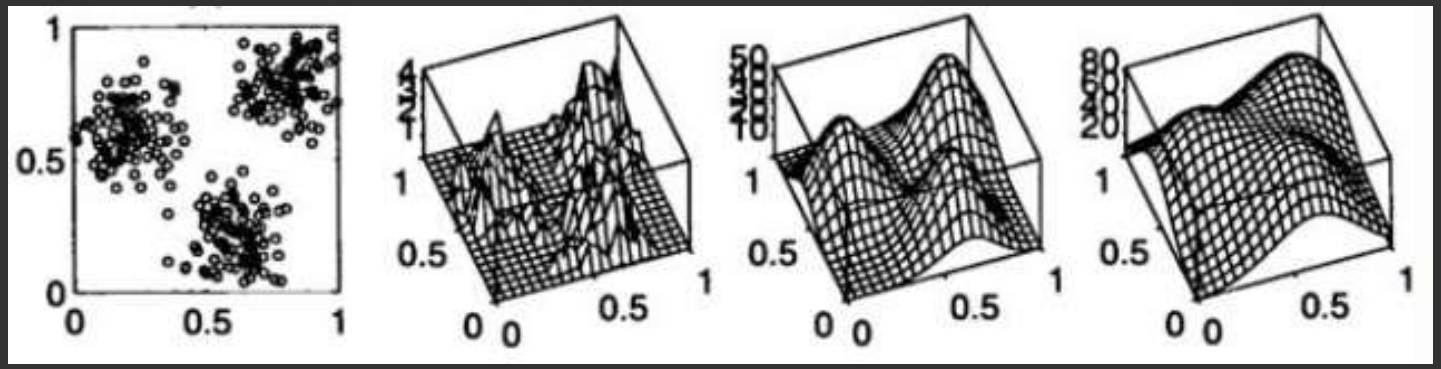
- Consider a collection of n data points in an M-dimensional space.
- Each data point is considered as a candidate for a cluster center.
- The data points are normalized within a hypercube [0,1].
- A *density measure* at data point x_i is defined as:

$$D_i = \sum_{j=1}^{n} e^{-\frac{||x_i - x_j||^2}{(r_a/2)^2}}$$

r_a (radius) positive number, defines a neighborhood; ||.|| denotes the Euclidean distance

- A data point will have a high density value if it has many neighboring data points.
- Once density measure is calculated at each data point, the one with the highest value is selected as the first cluster center.

Subtractive Clustering



r_a = 0.05 0.3 0.5

r_a may affect the smoothness of mountain functions

Subtractive Clustering

- To find the next cluster center, the density measure for each data point is revised subtracting the influence of the first cluster (data point: x_{c1}; its density measure: D_{c1})

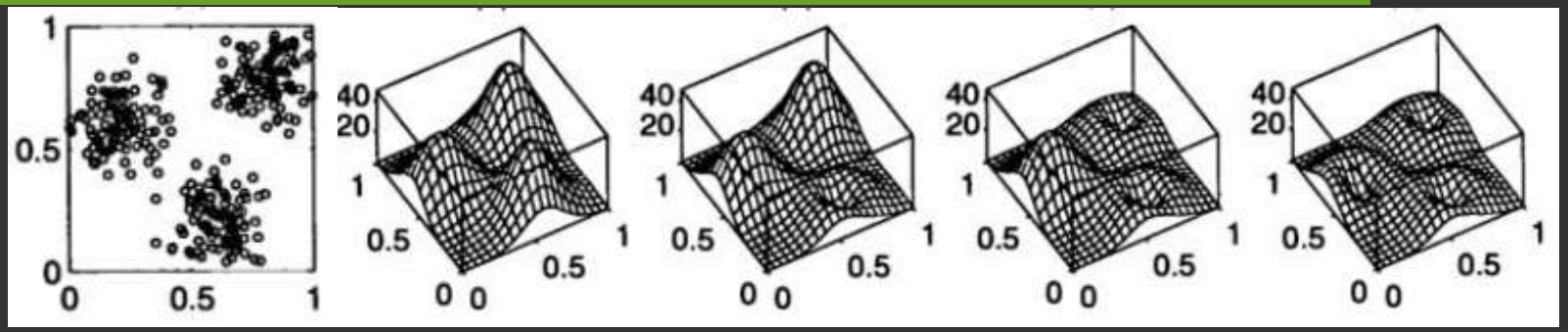
$$D_i = D_i - D_{c1} e^{-\frac{||x_i - x_{c1}||^2}{(r_b/2)^2}}$$

r_b (radius) positive number, defines a neighborhood with reductions in density measure;

$r_b = \eta\, r_a$ (squash factor > 1)

- The data points near the first cluster center x_{c1}, will have significantly reduced density measure, making the points unlikely to be selected as the next cluster
- Once the density measure for each data point is revised, the next cluster center is selected.

Subtractive Clustering



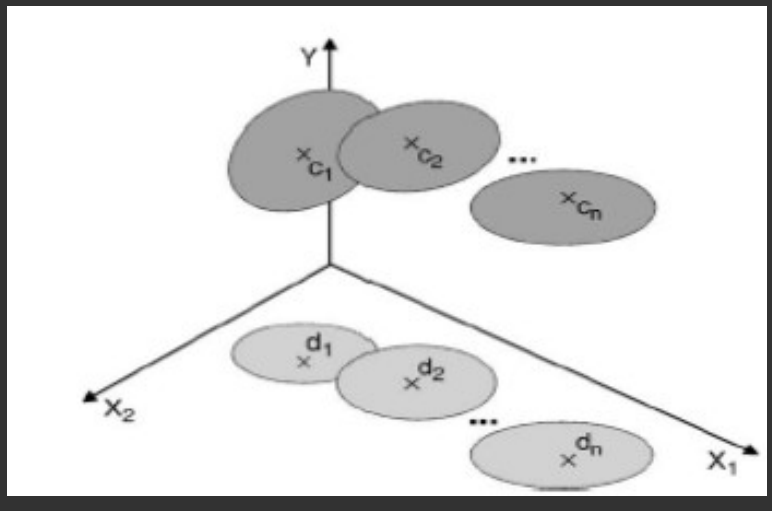
Cluster centers are selected, and mountains are destructed sequentially

- For ending the clustering process there are criteria for accepting and rejecting cluster centers that help avoid marginal cluster centers
  - If *Ratio* > *Accept ratio* then accept new cluster
  - If *Reject ratio* < *Ratio* <= *Accept ratio* and *high distance* to other clusters then accept new cluster
  - If *Ratio* <= *Reject ratio* then stop

where $Ratio = D_{ck}/D_{c1}$; *Accept ratio* and *Reject ratio* are positive numbers [0,1]

Subtractive Clustering

- The clusters are projected onto the input space in order to find the antecedent parts of the fuzzy rules.
- The centers of the membership functions are obtained by projecting the center of each cluster in the corresponding axis. The widths are obtained on the basis of the radius.
- In this way, one cluster corresponds to one rule of the TSK model.
- The consequent parts of the rules can then be simple functions.



Demo ANFIS

Automobile Data

Matlab:

>> load auto.mat (multiple input - single output)

>> anfisedit

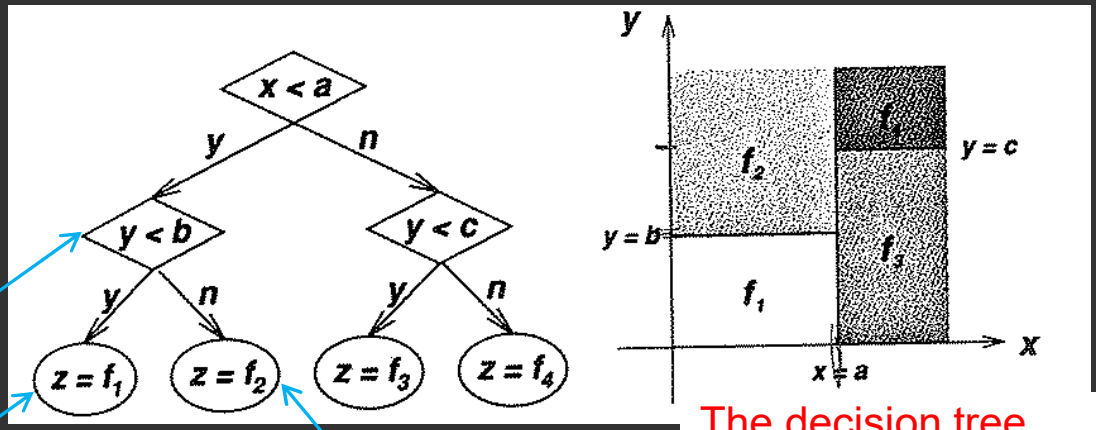
This example uses demographic and trip data from 100 traffic analysis zones in Delaware. Contains five demographic factors as **input variables**: population, number of dwelling units, vehicle ownership, median household income, and total employment and one **output variable:** number of automobile trips.

CART: Classification and Regression Trees

- CART generates a tree partitioning of the input space, which relieves the "curse of dimensionality" problem (num. of rules increasing exponentially with num. of inputs)

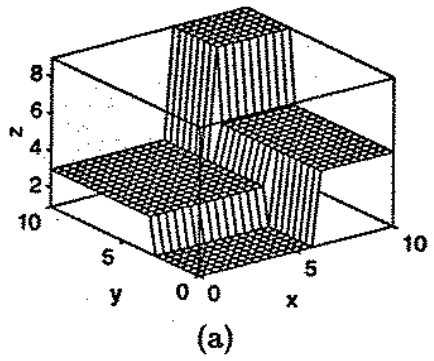
CART: Classification and Regression Trees

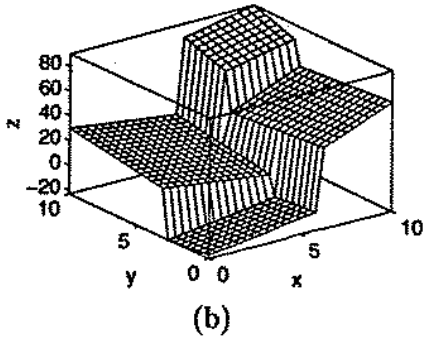
Binary tree with
two inputs and one output



The decision tree partitions the input space into 4 regions

Constants: a=6, b=3, c=7
f1=1, f2=3, f3=5, f4=9





Linear functions:
f1= 2x-y-20,
f2= -2x+2y+10,
f3=6x-y+5,
f4=3x+4y+20

CART: Classification and Regression Trees

CART first **grows** the tree extensively based on a training data set, and then **prunes** the tree back based on a minimum cost-complexity principle

Tree Growing

- CART grows a decision tree by determining a succession of splits (decision boundaries) that partition the training data into disjoint subsets
- Starting from the root node (contains all the training data), an **exhaustive search** is performed to find the split that best reduces an error measure (cost function)
- The data set is partitioned into two subsets and the same splitting method is applied to both child nodes
- This recursive procedure terminates either when the error measure associated with a node falls below a certain tolerance level, or when the error reduction does not exceed a certain threshold value

CART: Classification and Regression Trees

For a regression tree, the error measure that quantifies the performance of a node *t* in separating data into disjoint subsets is usually taken as the square error (or residual) of a local model employed to fit the data set of the node:

$$E(t) = \min_{\boldsymbol{\theta}} \sum_{i=1}^{N(t)} (y_i - d_t(\mathbf{x}_i, \boldsymbol{\theta}))^2,$$

where $\{x_i, y_i\}$ is a data point, $d_t(\mathbf{x}_i, \boldsymbol{\theta})$ is a local model for node *t* (with modifiable parameter θ) and E(*t*) is the mean-square error of fitting the local model to the data set in the node

Constant (d(x, θ) = θ): the minimizing of *E*(*t*) is the average value of the desired output

$$\theta^* = \frac{1}{N(t)} \sum_{i=1}^{N(t)} y_i.$$

CART: Classification and Regression Trees

Linear model (d(x, θ) = linear model with linear parameters): least-squares methods to identify the minimizing θ*

For any split *s* of node *t* into t_l and t_r, the change in the error measure is expressed as:

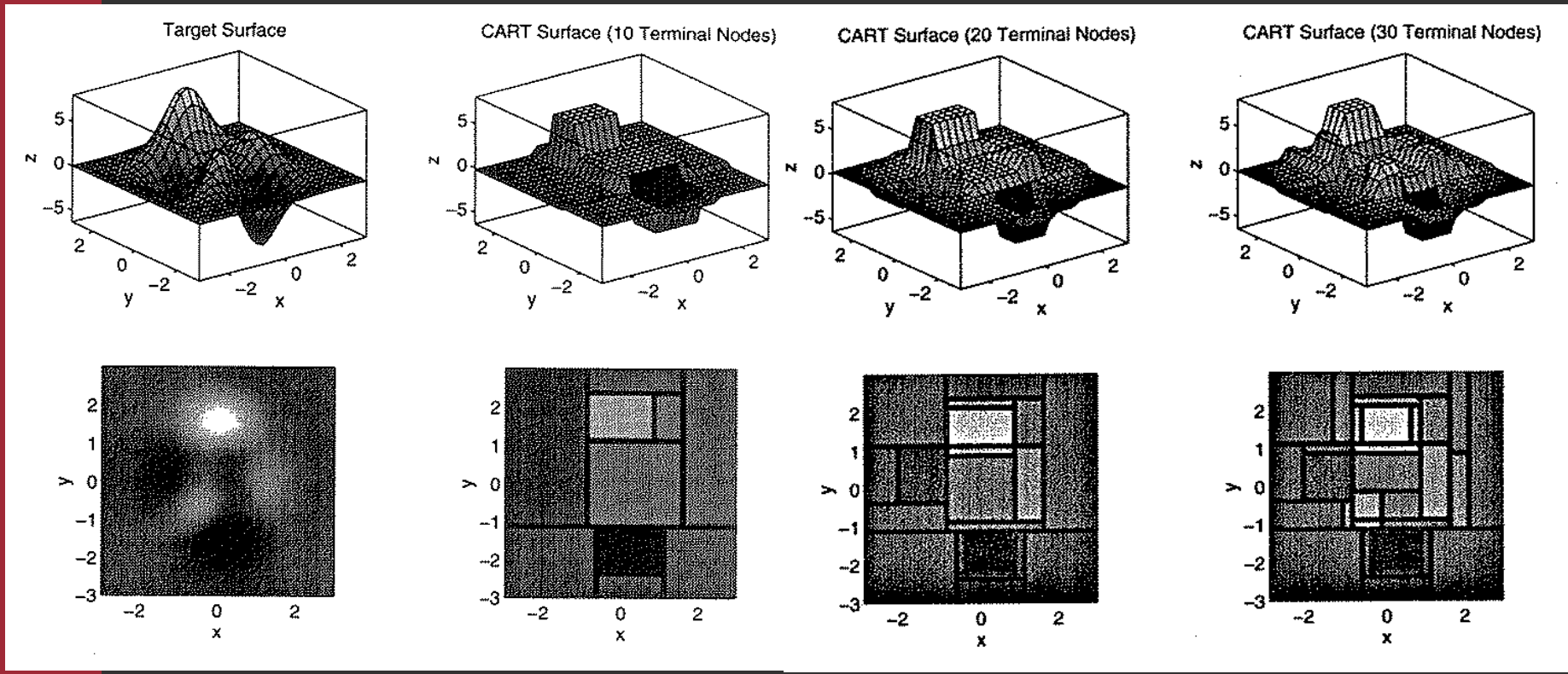
$$\Delta E(s,t) = E(t) - E(t_l) - E(t_r)$$

the best split *s** is the one that maximizes the decrease in the error measure:

$$\Delta E(s^*,t) = \max_{s \in S} \Delta E(s,t).$$

The strategy for growing a regression tree is to split nodes (or data set) iteratively and thus maximize the decrease in $E(T) = \sum_{t \in \tilde{T}} E(t)$, i.e. the overall error measure (or cost) of the tree.

CART: Classification and Regression Trees



CART: Classification and Regression Trees

Tree Pruning

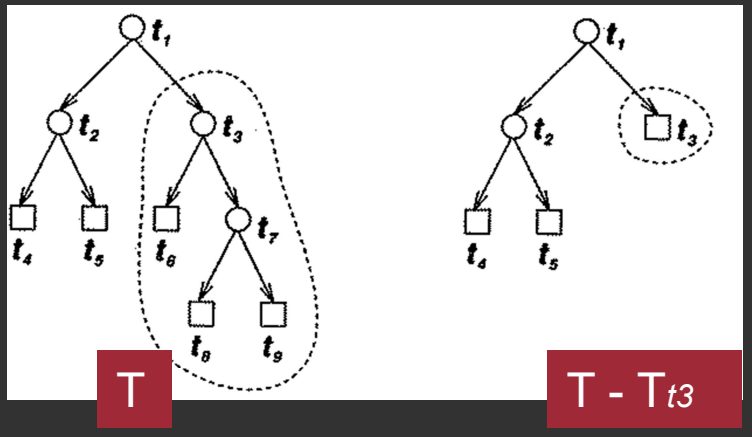
- The tree that the growing procedure yields is often too large and it is biased towards the training data set (overfitting)

Principle of minimum cost-complexity: prune the tree finding the weakest subtree in it, i.e. considering both the training error measure and the number of terminal nodes (measure of tree's complexity)

The internal node with the smallest α_t is chosen as the target node for shrinking

$$\alpha_t = \frac{E(\mathbf{T}) - E(T_t)}{|\tilde{T}_t| - 1}$$

number of terminal nodes in t



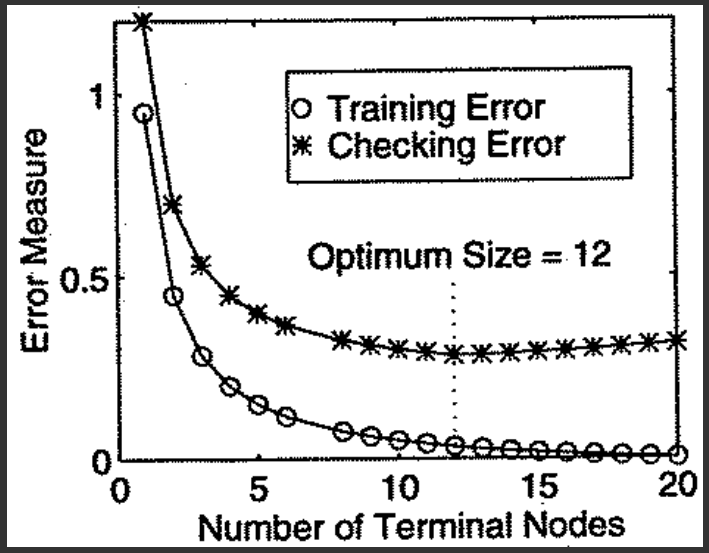
CART: Classification and Regression Trees

Therefore, a tree-pruning cycle consists of the following tasks:

1. Calculate α_t for each internal node t in Ti
2. Find the minimal α_t and choose T - T_t as the next minimizing tree

Next step: select one of these candidate trees as the optimum-size tree:

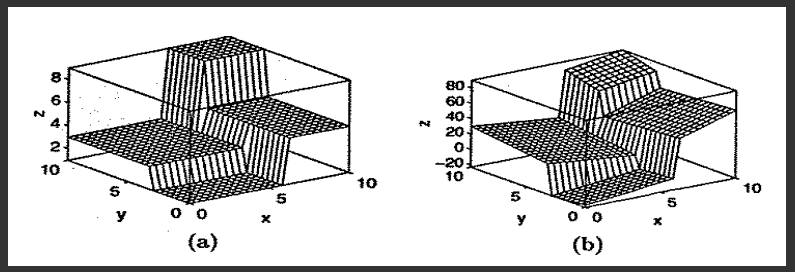
- use an independent test (checking)
- cross-validation



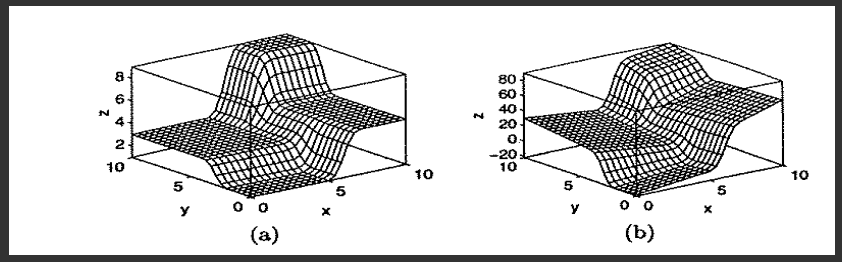
CART - ANFIS

CART is used to find the number of ANFIS rules and the initial locations of membership functions before training

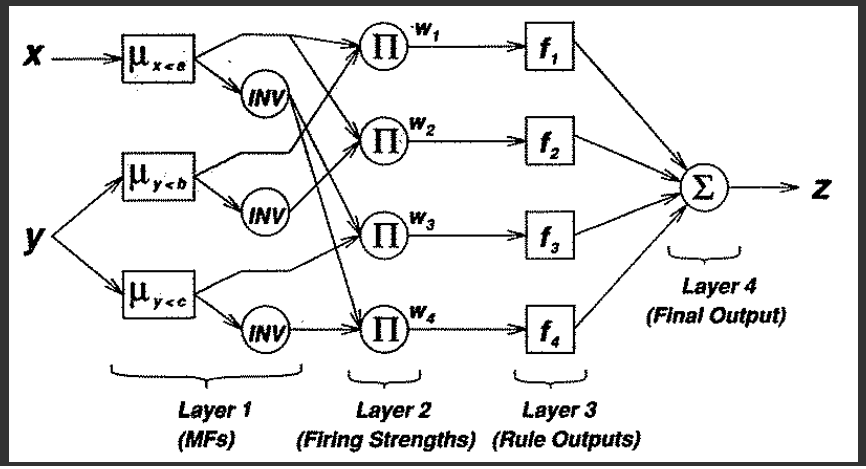
$$\begin{cases} \text{If } x < a \text{ and } y < b, \text{ then } z = f_1. \\ \text{If } x < a \text{ and } y \geq b, \text{ then } z = f_2. \\ \text{If } x \geq a \text{ and } y < c, \text{ then } z = f_3. \\ \text{If } x \geq a \text{ and } y \geq c, \text{ then } z = f_4. \end{cases}$$



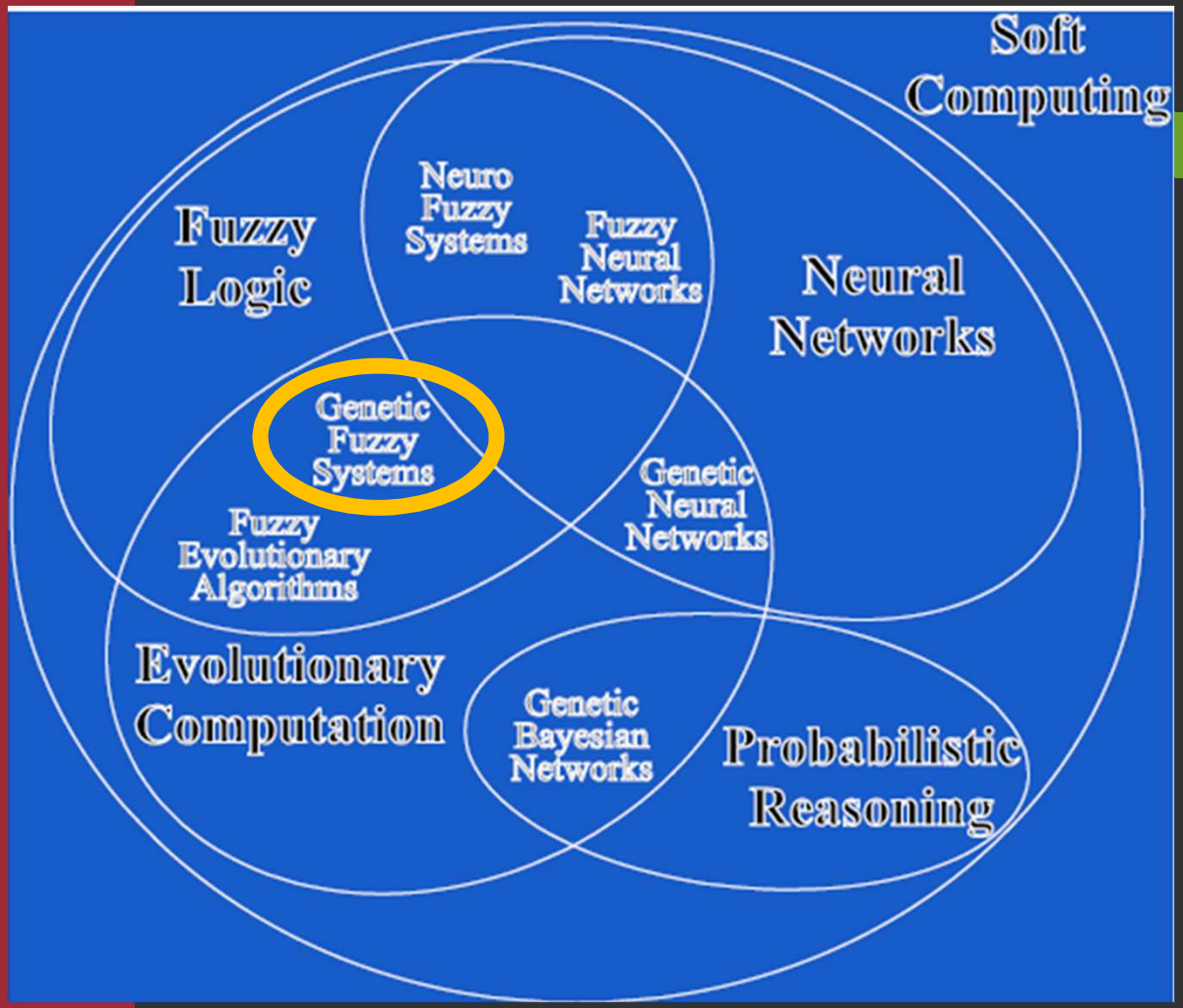
Crisp: discontinuous boundaries



Fuzzy: continuous boundaries



Hybridization: Genetic Fuzzy Systems



- Evolutionary Computation provide robust search capabilities (global and local) in complex spaces.
- FS present robust and flexible inference methods in domains subject to imprecision and uncertainty.

Genetic Fuzzy Systems

- The use of genetic/evolutionary algorithms (GAs) to design fuzzy systems constitutes one of the branches of the Soft Computing paradigm: genetic fuzzy systems (GFSs)
- The most known approach is that of genetic fuzzy rule-based systems, where some components of a fuzzy rule-based system (FRBS) are derived (adapted or learnt) using a GA
- Some other approaches include genetic fuzzy neural networks and genetic fuzzy clustering, among others

Fuzzy Rule-based Systems

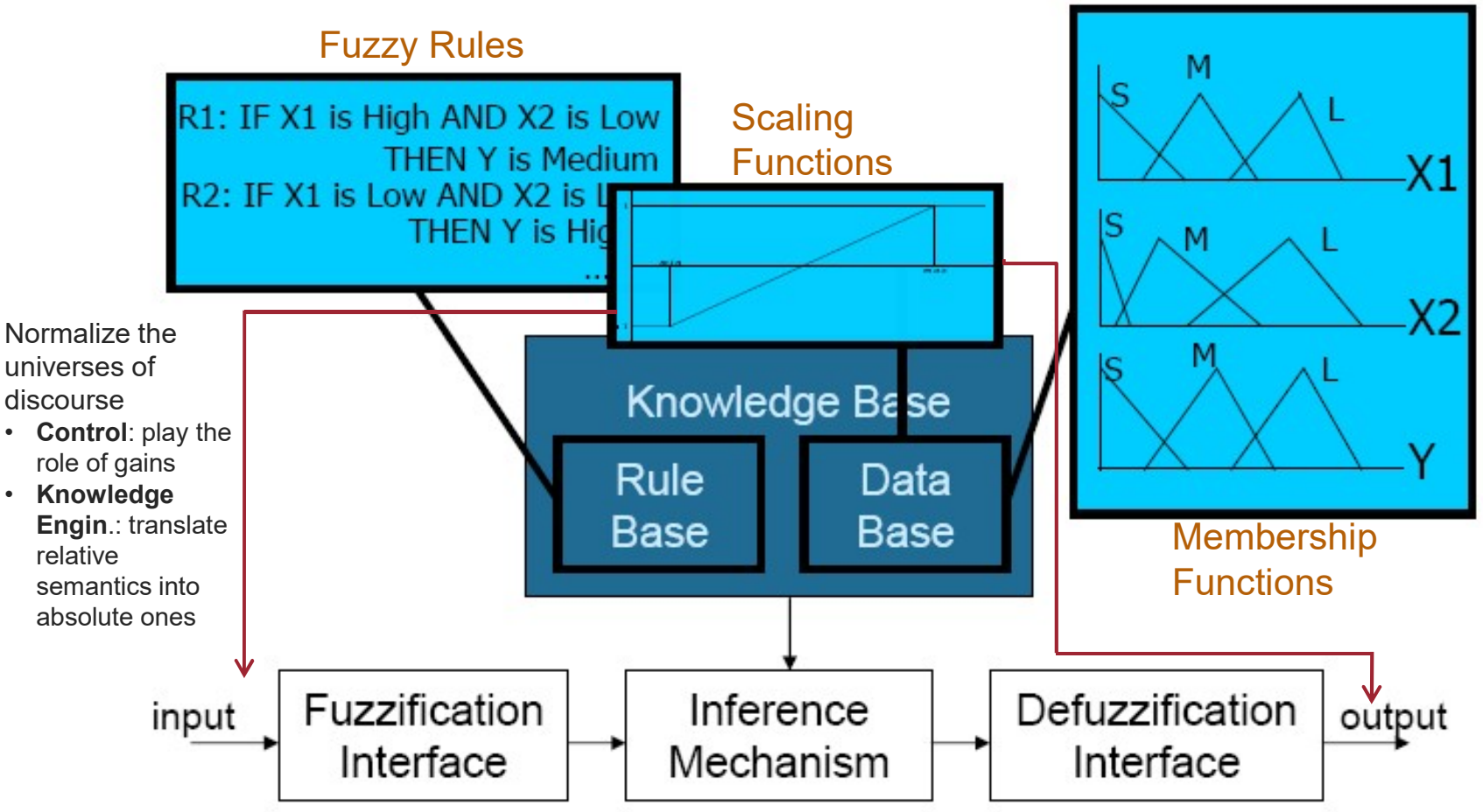
Design of fuzzy rule-based systems:

- An FRBS (regardless it is a fuzzy model, a fuzzy logic controller or a fuzzy classifier), is comprised by two main components:
  - The Knowledge Base (KB), storing the available problem knowledge in the form of fuzzy rules
  - The Inference System, applying a fuzzy reasoning method on the inputs and the KB rules to give a system output

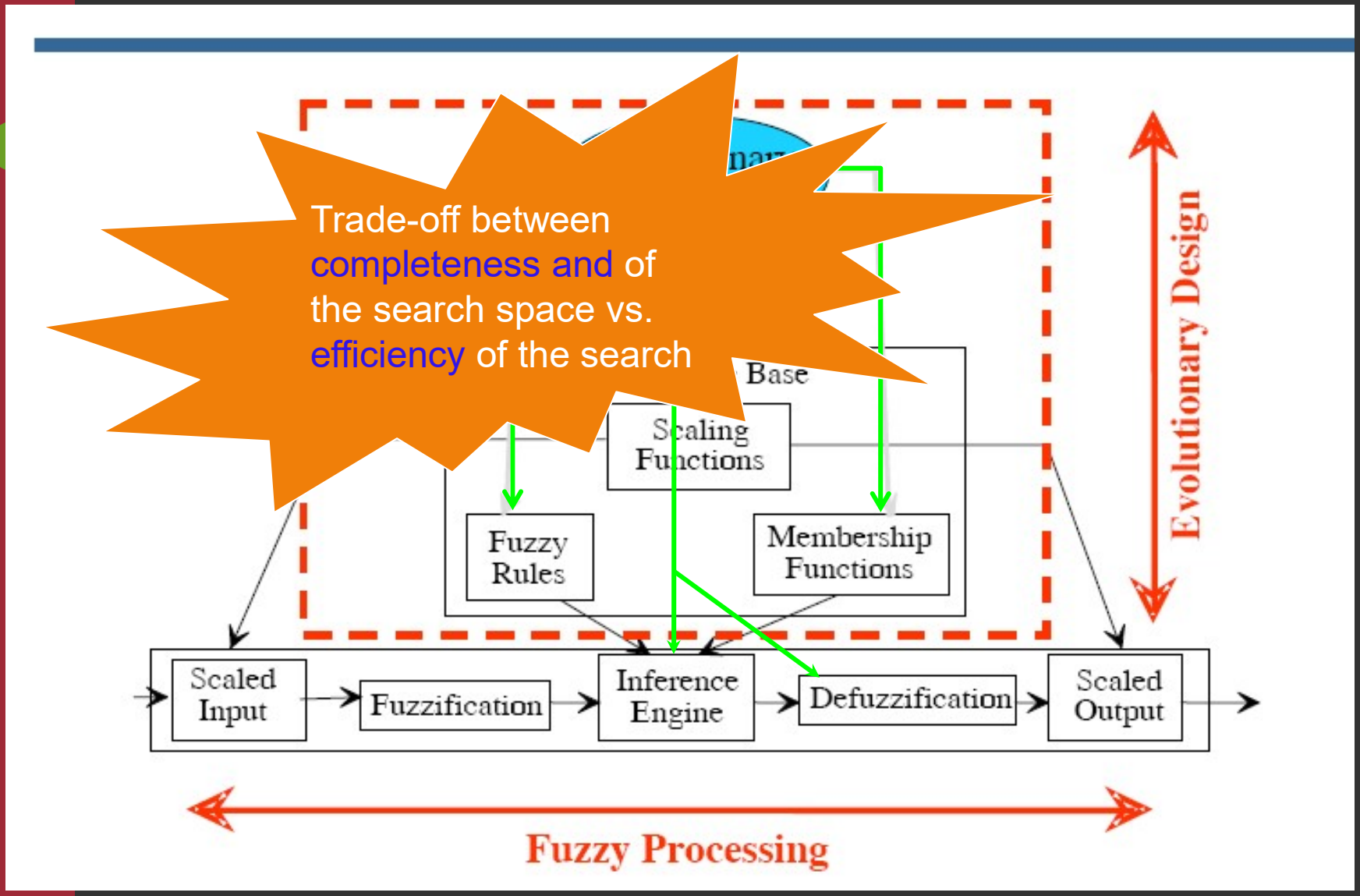
- Both must be designed to build an FRBS for a specific application:
  - The KB is obtained from expert knowledge or by machine learning methods
  - The Inference System is set up by choosing the fuzzy operator for each component (conjunction, implication, defuzzifier, etc.)

  Sometimes, the latter operators are also parametric and can be tuned using automatic methods

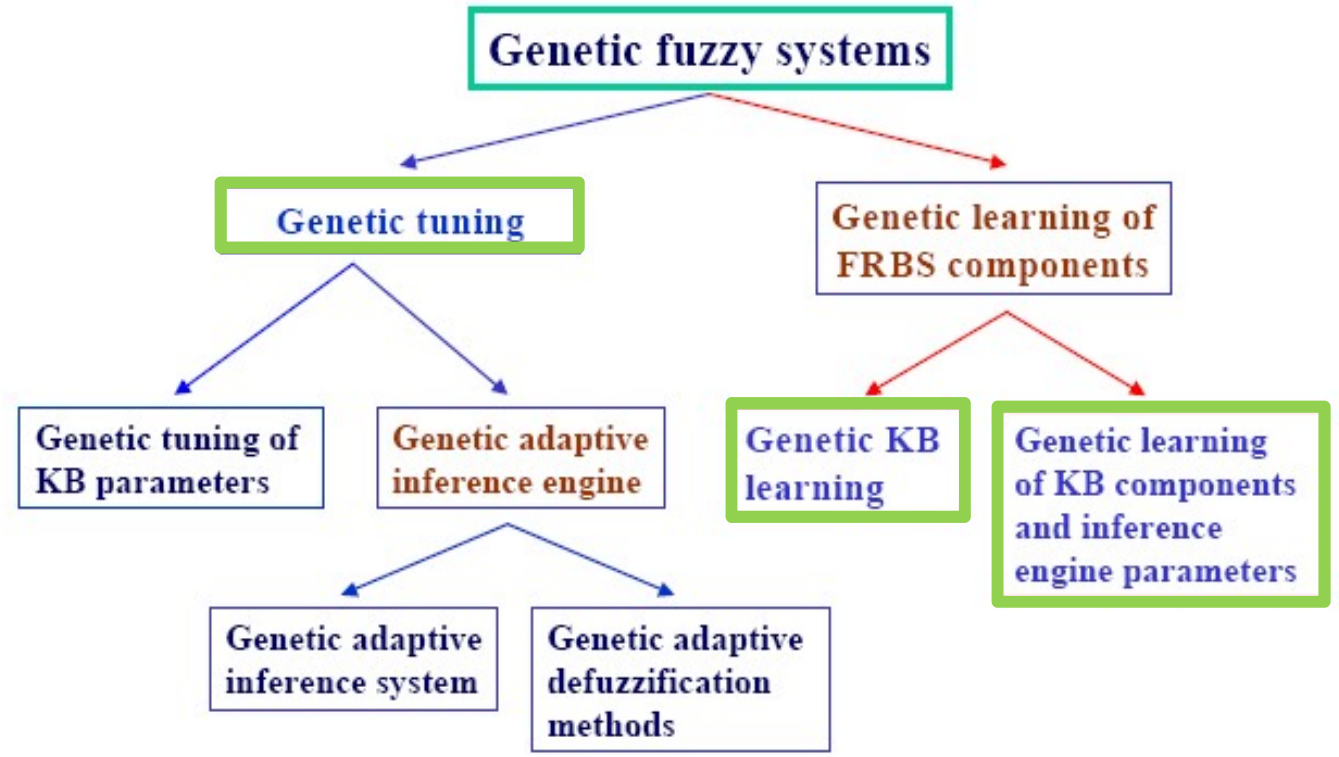
Fuzzy Rule-based Systems



Trade-off between completeness of the search space and efficiency of search



Taxonomy of GFS



Concerned with optimisation of an existing FRBS

Automated design method that does not depend on a predefined set of rules

Classical GFS learning approaches

- **Genetic learning of the FRBS Rule Base**
  - **Pittsburgh learning approach**
  - **Michigan learning approach**
  - **Iterative Rule learning approach (SLAVE)**

Cooperation versus competition

There is a major problem that appears when designing FRBSs by means of EAs which has to be solved adequately in order to obtain accurate FRBS:

• On the one hand, GFRBSs take an interesting feature of the FRBSs into account, the **interpolative reasoning they develop**, which is a consequence of *cooperation among the fuzzy rules composing the KB*.

• On the other hand, the main feature of an EA is considered, which is *the* ***competition induced among the population members*** *representing possible solutions to the problem being solved*, which allows us to obtain better ones.

Since a GFRBS combines both said features, it works by inducing competition to get the best possible cooperation. The problem is to find the best possible way to put this into effect.

Genetic learning of the FRBS Rule Base

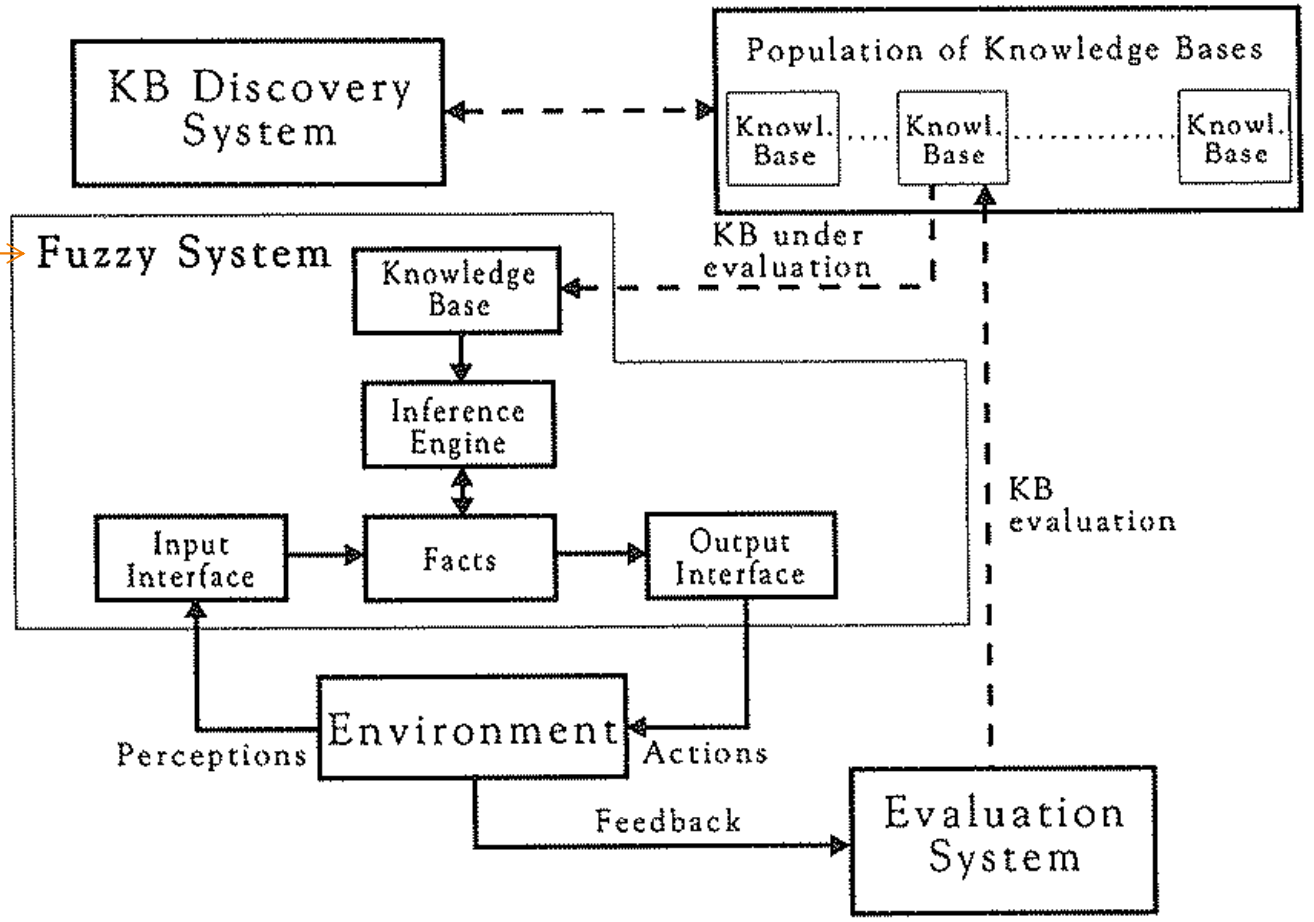
Pittsburgh Learning Approach:

- Each chromosome encodes a whole fuzzy rule set and the derived RB is the best individual of the last population
- The fitness function evaluates the performance at the complete RB level, so the CCP is easy to solve
- However, the search space is huge, thus making difficult the problem solving and requiring sophisticated GFS designs
- Mainly used in off-line learning (fuzzy modeling and classification applications)

Genetic learning of the FRBS Rule Base

Pittsburgh Learning Approach:

Mamdani / TSK



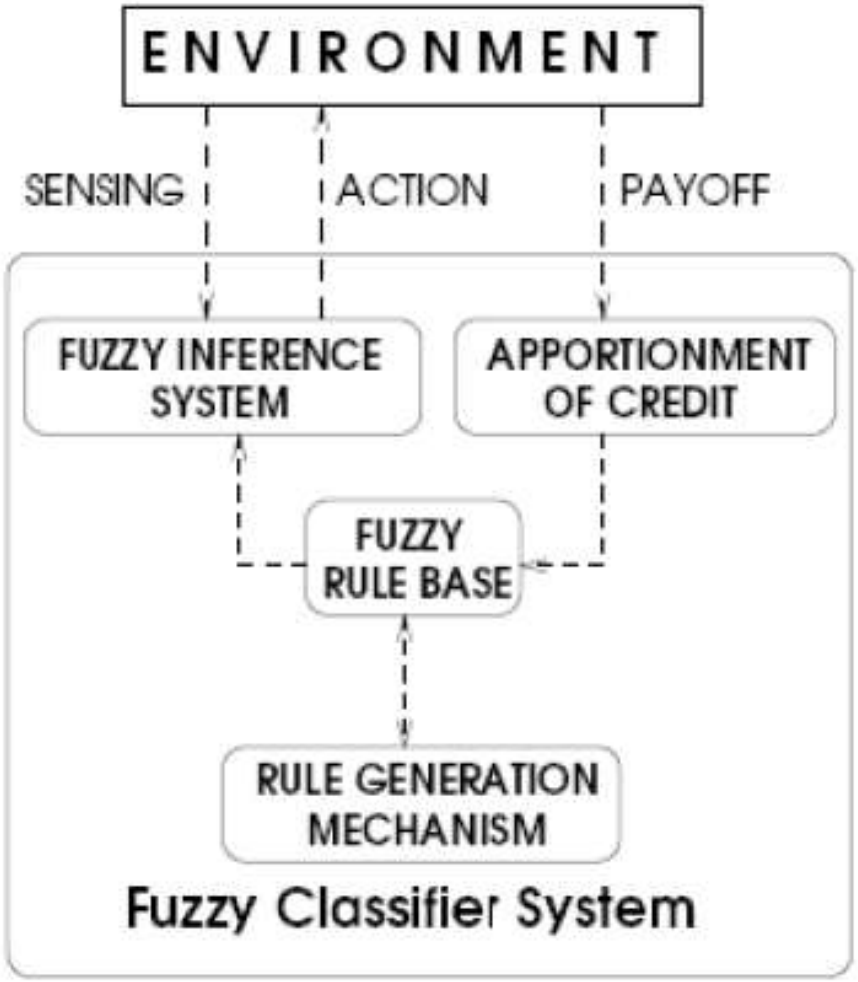
Genetic learning of the FRBS Rule Base

Michigan Learning Approach:

- Each chromosome encodes a single fuzzy rule and the derived RB is composed of the whole population
- Reinforcement mechanisms (reward (credit apportion) and weight penalization) are considered to adapt the rules through a GA
- Low weight (bad performing) rules are substituted by new rules generated by the GA
- The key question is to induce collaboration in the derived RB as the evaluation procedure is at single rule level (cooperation vs. competition problem (CCP))
- Mainly used in on-line learning (fuzzy control applications)

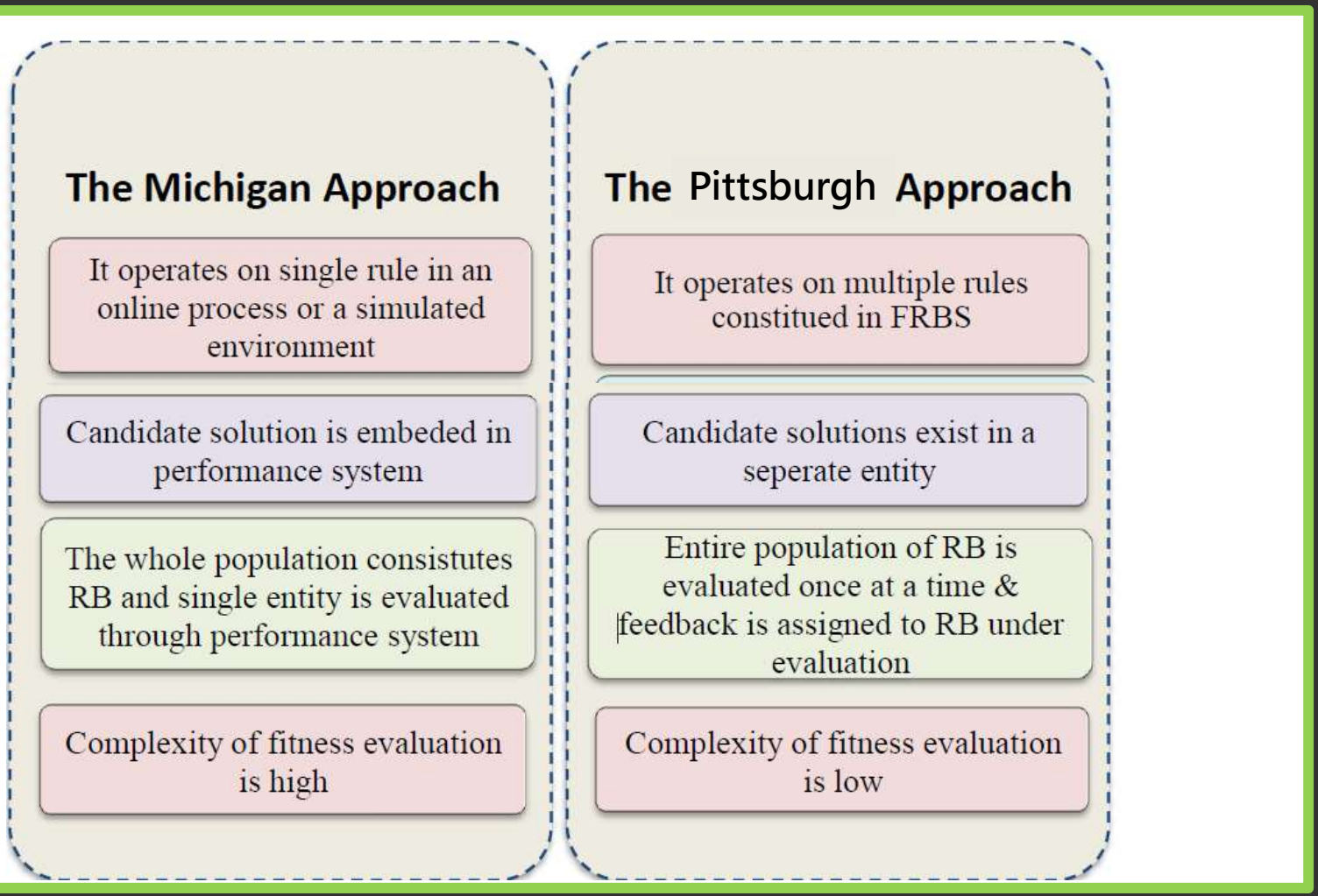
Genetic learning of the FRBS Rule Base

Michigan Learning Approach:



Reward or penalise individual rules, based on their contribution to the overall success or failure of the entire fuzzy classifier system

Pittsburgh vs Michigan Approaches



Genetic derivation of the FRBS Rule Base

Iterative Rule Learning Approach:

- Intermediate approach between the Michigan and Pittsburgh ones, based on partitioning the learning problem into several stages and leading to the design of multi-stage GFSs
- As in the Michigan approach, each chromosome encodes a single rule, but a new rule is learnt by an iterative fuzzy rule generation stage and added to the derived RB, in an iterative fashion, in independent and successive runs of the GA
- The evolution is guided by data covering criteria (rule competition). Some of them are considered to penalize the generation of rules covering examples already covered by the previously generated fuzzy rules (soft cooperation)

Generation Process: derives a preliminary set of rules representing the knowledge existing in the data set

Genetic derivation of the FRBS Rule Base

Iterative Rule Learning Approach:

- A second post-processing stage is considered to refine the derived RB by selecting the most cooperative rule set and/or tuning the membership functions (cooperation induction)
- Hence, the CCP is solved taking the advantages of both the Michigan and Pittsburgh approaches (small search space and good chances to induce cooperation)
- Mainly used in off-line learning (fuzzy modeling and classification applications)

Post-processing Process: refines the previous rule set in order to remove the redundant rules and select those fuzzy rules that cooperate in an optimal way

SLAVE (Structural Learning Algorithm in Vague Environment)

SLAVE generation process

Generation process:

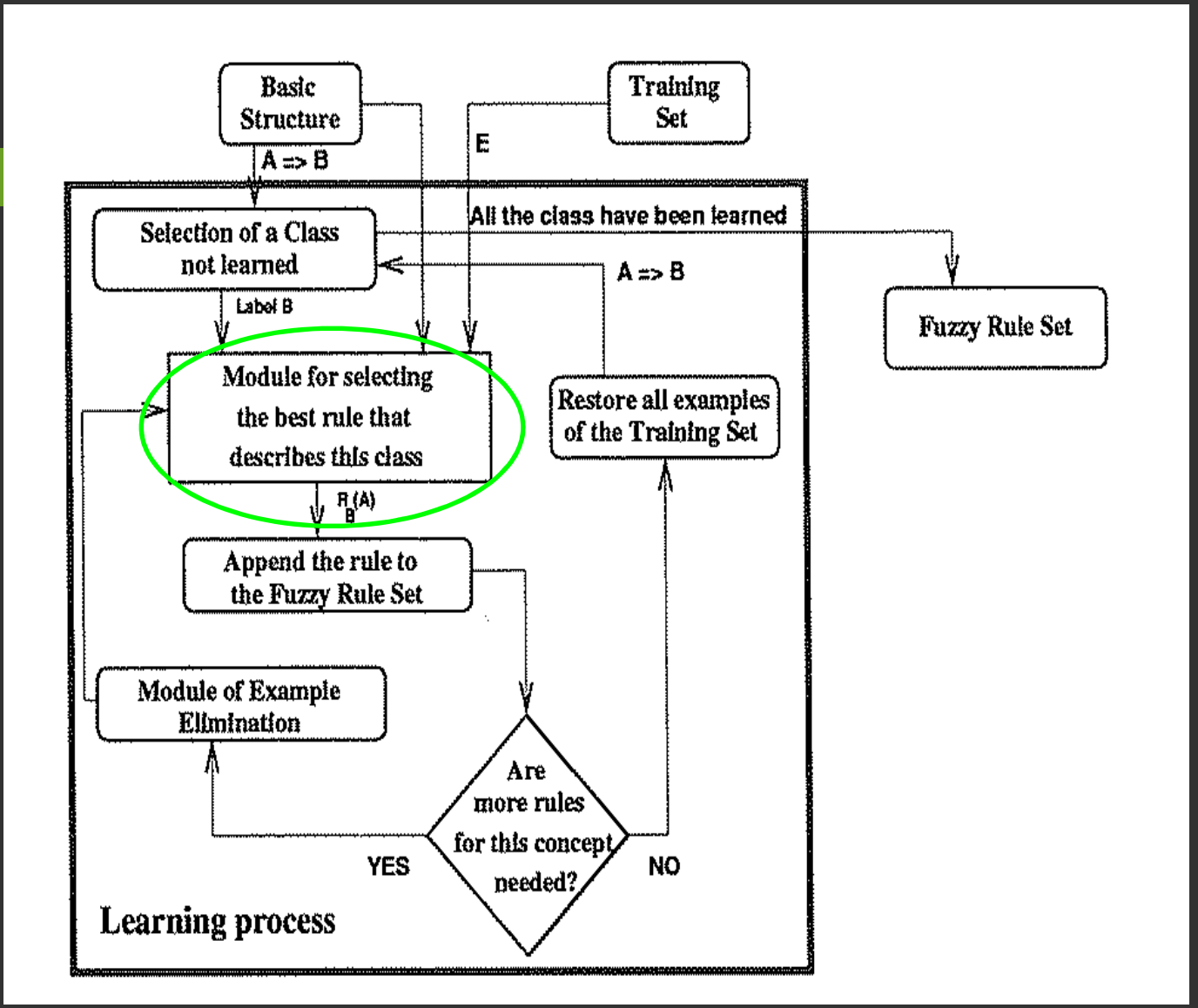
1. Use a GA to obtain a rule for the system

2. Incorporate the rule into the final set of rules

3. Eliminate from the data set all those examples that are covered by this rule to a sufficient degree λ (λ-covered)

4. If the set of rules obtained is sufficient to represent the examples in the training set, the system ends up returning the set of rules as the solution. Otherwise return to step 1.

SLAVE generation process



The rule model in SLAVE

- A common approach to code individual rules is the use of the disjunctive normal form (DNF) represented in the form of a fixed length binary string

$$\ldots \text{ and } X_i = \{L_4, L_5, L_6\} \text{ and } \ldots$$

$$\ldots \text{ and } \{X_i \text{ is } L_4 \text{ or } X_i \text{ is } L_5 \text{ or } X_i \text{ is } L_6\} \text{ and } \ldots$$

0	0	0	1	1	1

The rule model in SLAVE

- A DNF fuzzy rule allows an antecedent variable to take a disjunction of linguistic terms from its domain as a value:

IF Femur_length is (medium or big-medium or big) and Head_diameter is (medium) and Foetus_sex is (male or female or unknown) THEN Foetus_weight is normal

Femur_length ={small,small-medium, medium, big-medium,big}

Head_diameter ={small,medium,big}

Foetus_sex ={male,female,unknown}

Foetus_weight = {low, normal, high}

IF Femur_length is (medium or big-medium or big) and Head_diameter is (medium) and ~~Foetus_sex is (male or female or unknown)~~ THEN Foetus_weight is normal

SLAVE parameters

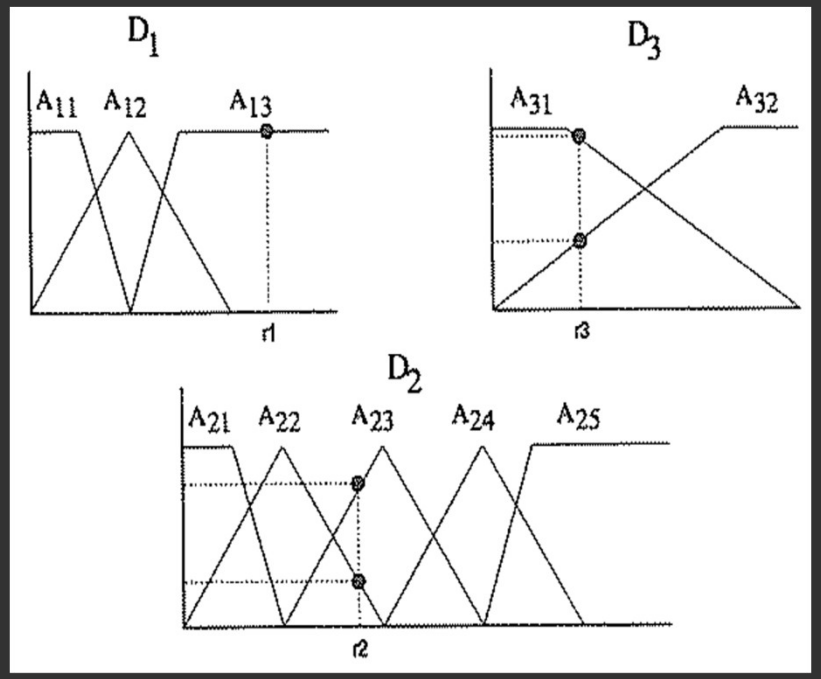
- ***Genetic operators:*** two-points crossover, uniform mutation. Can use proportional, linear ranking and elitist selection schemes.
- ***Generation of the initial population:*** The initial population is generated by randomly selecting examples from the class that must be learned (i.e. matches the rule consequent value) and obtaining for each one the most specific rule antecedent that best describes the example.

Crisp input vector of the example: $(r1, r2, r3)$

X_1 *is* A_{13} *and* X_2 *is* A_{23} *and* X_3 *is* A_{31}

C= (0010010010)

Note: the consequent part is specified by the iterative covering method



SLAVE parameters

•***Fitness function:*** to generate fuzzy classification rules that give the consistency criterion priority over the positive examples.

$$F(R_i) = \begin{cases} n^+(R_i), & \text{if } R_i \text{ is } k\text{-consistent} \\ 0, & \text{otherwise} \end{cases}$$

- *Positive example*: matches the antecedents as well as the consequent

- *Negative example:* matches the antecedents and not the consequent

- $n^+(R_i)$, number of positive examples.

- R_i is *k*-consistent when its associated number of negative examples is less than a percentage *100.k* of the number of the positive examples.

SLAVE parameters

- ***Termination criteria:*** the GA ends returning the best rule of the last population if:

- the number of iterations is greater than the fixed limit.

- the fitness function of the best rule in the population does not increase its value within a fixed number of iterations and rules with this consequent have already been obtained.

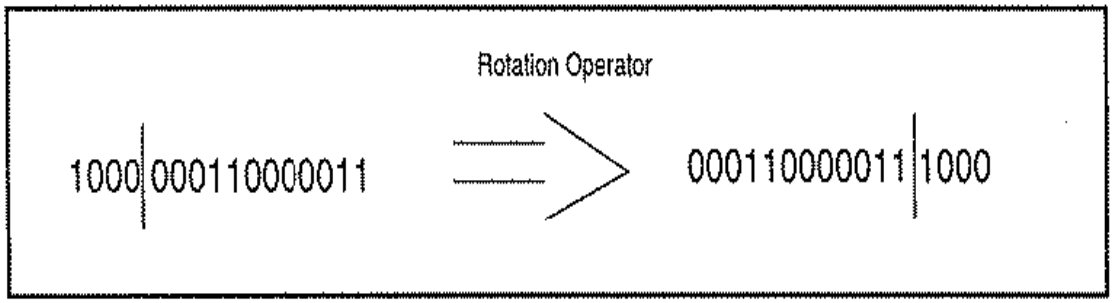
- when no rule has been obtained for this concept yet, the generating process continues, until the current best rule eliminates at least one example from the training set.

SLAVE new genetic operators

Introducing new genetic operators with the intention of:

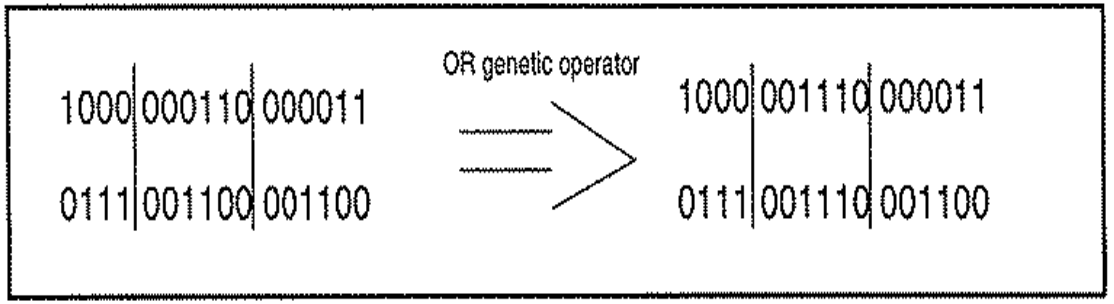
1. Improving the diversity in the genetic population
2. Increasing the understanding of the rules that are obtained

Rotation operator (1):



OR operator (1):

(crossover)



SLAVE new genetic operators

Generalization operator (2):

Antecedents of variable X_2: **00111**/**00101**; **fitness(00111)=fitness(00101)**

$$\ldots \text{ and } X_2 \text{ is } \{A_{23}, A_{24}, A_{25}\} \text{ and } \ldots$$
$$\ldots \text{ and } X_2 \text{ is higher than or equal to } A_{23} \text{ and } \ldots$$

$$\ldots \text{ and } X_2 \text{ is } \{A_{23}, A_{25}\} \text{ and } \ldots$$

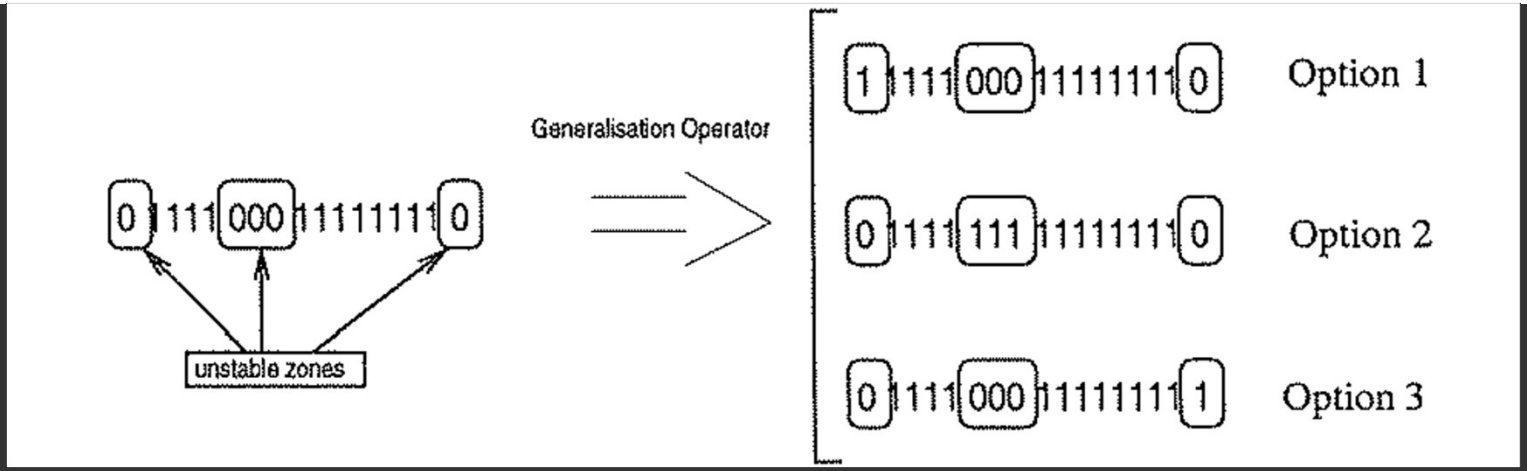
Red more understandable description from a descriptive point of view but the GA selects randomly one of them.

This operator tries to maximize the number of stable variables in the rule, in the sense that a variable is considered stable if the terms associated to it form a unique, consecutive sequence without gaps.

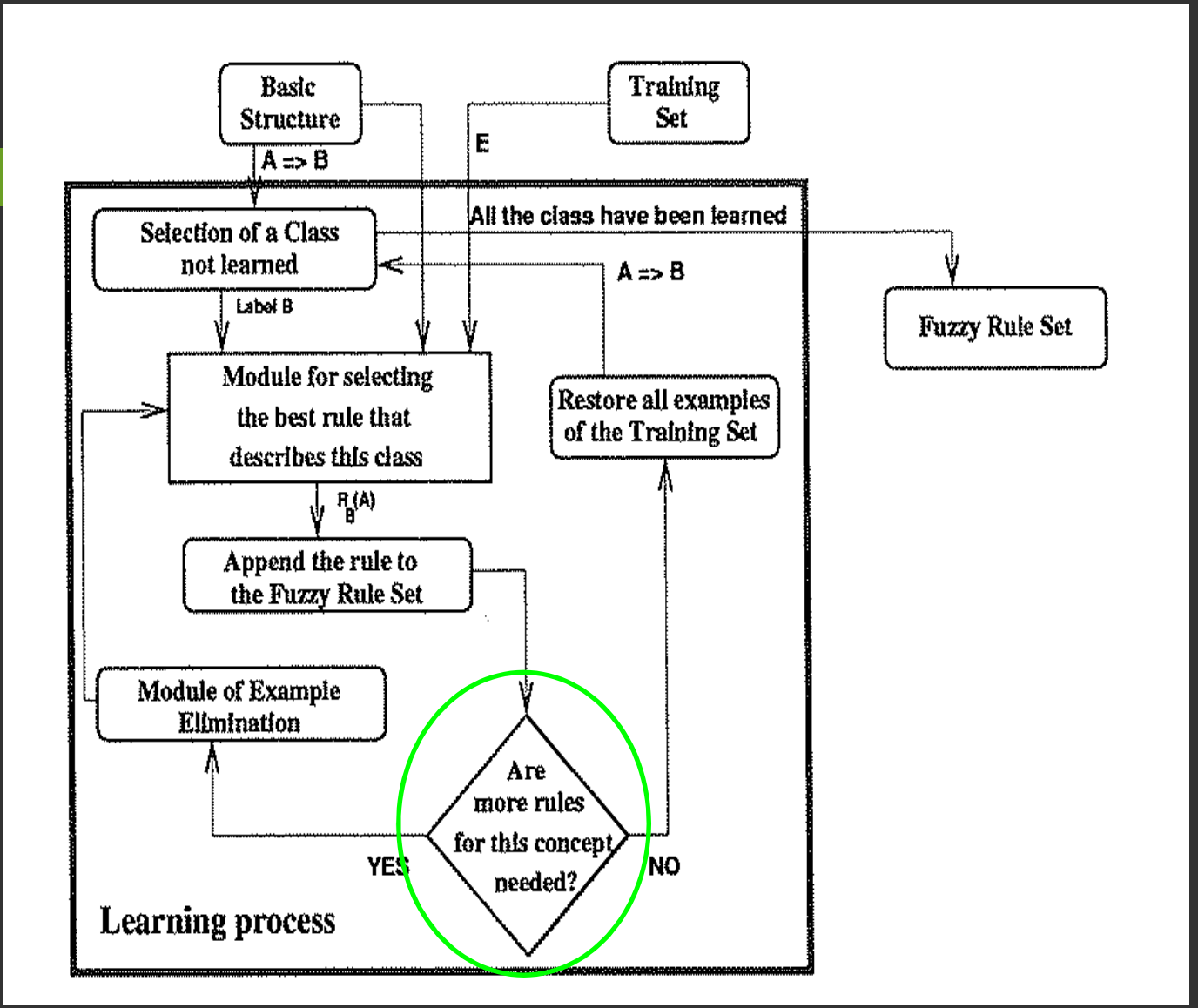
SLAVE new genetic operators

The generalization operator works as follows:

1. *A variable of a DNF rule antecedent encoded in the individual is selected at random.*
2. *If this variable is unstable, do the following:*
   - 2.1 *Detect its unstable regions and randomly select one of them.*
   - 2.2 *Replace all the '0'-bits in that region with a consecutive sequence of '1'-bits.*
   - 2.3 *If the fitness value of the original chromosome is smaller than or equal to that of the new chromosome, substitute the former by the latter*



SLAVE iterative covering method



SLAVE iterative covering method

• The termination criteria in the SLAVE covering method is based on the *weak completeness condition for a class.*

• The SLAVE covering method might terminate generating fuzzy rules for a concept (associated to the output) even though there remain class examples not yet covered by the current fuzzy rule set.

• A set of rules $\{R_1, R_2, \ldots, R_s, R_{s+1}\}$, with output class C satisfies the weak completeness condition if and only if,

$$\begin{cases} R_{s+1} \text{ is not } k\text{-consistent} \\ or \\ R_{s+1} \text{ is } k\text{-consistent and } E_\lambda(R_{s+1}) = \emptyset. \end{cases}$$

examples λ-covered by R_{s+1}

• The iterative covering method terminates upon the first rule R_{S+1} that satisfies the weak completeness condition, without adding it to the final rule set.

SLAVE post-processing process

- The post-processing algorithm tries to promote the cooperation among the fuzzy rules in order to improve the accuracy of the FRBs and to simplify the fuzzy rule set.

- The post-processing algorithm is an heuristic algorithm that uses the ***hill-climbing*** optimization strategy (local search) and it is composed of operations of generalization, addition and elimination of rules that are repeated until the global error and the number of rules cannot be decreased.

- It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.

SLAVE Examples

- The IRIS database [11]. In this database we try to classify three different classes of plants using four predictive variables, using 150 examples. All the predictive variables have a continuous domain and we have considered five uniformly distributed linguistic labels for discretizing each range.
- The INFARCTION database [22, 15]. This database is made up by 14 different variables for determining the nominal diagnostic variable. There are 2 nominal variables and the rest of them are continuous. These variables are divided into two different type of tests, morphological methods and biochemical analysis. These tests are accepted as the best markers for the postmortem diagnosis of myocardial infarction. The databases has 78 examples and contains missing values. We have considered seven uniformly distributed linguistic labels for discretizing the continous variables.
- WINE recognition data. These data are the results of a chemical analysis of wines from the same region but different types of grapes, using 13 continuous variables and 178 examples. We have used seven uniformly distributed linguistic labels for discretizing the range of each variable.

SLAVE Examples

		IRIS	INFARCTION	WINE
C4.5	accuracy	92.7	81.4	93.2
C4.5rules	accuracy	94.4	82.2	93.5
the original SLAVE	accuracy	95.72	83.22	88.54
	rules	4.2	11.8	21.8
	time	1.00	1.00	1.00
only OR	accuracy	95.72	83.41	89.13
	rules	4.2	8.6	18.3
	time	0.94	0.36	0.48
OR and Generalization	accuracy	95.72	86.4	90.44
	rules	4.4	6.8	16.6
	time	0.46	0.16	0.29
OR and Rotation	accuracy	95.72	89.04	93.32
	rules	4.2	8.4	16
	time	0.46	0.17	0.28
OR, Generalization, Rotation	accuracy	95.72	90.03	93.83
	rules	4.4	7.4	16.2
	time	0.45	0.16	0.27

Bibliography

- G.J. Klir and B. Yuan. Fuzzy sets and fuzzy logic. Theory and Applications. Prentice Hall, 1995
- A. Celikyilmaz and I.B. Türksen. Modeling uncertainty with Fuzzy Logic. Springer, 2009
- L.A. Zadeh and R.A. Aliev. Fuzzy logic theory and applications. Part I and Part II. World Scientific, 2019
- J.M. Mendel. Uncertain Rule-Based Fuzzy Systems. Introduction and New Directions. Springer, 2017
- D. Rutkowska. Neuro-Fuzzy architectures and hybrid learning. Physica-Verlag, 2002
- M.R. Berthold and D.J. Hand, editors. Intelligent Data Analysis: An Introduction. Springer, 2003
- Genetic fuzzy systems. O. Cordón, F. Herrera, F. Hoffman, L. Magdalena, 2001
- SLAVE: A genetic learning system based on an iterative apporach. A. Gonzalez, R. Perez. IEEE Transactions on Fuzzy Systems, Volume: 7, Issue: 2, 1999