An Introduction to Fuzzy Logic Part IV

Beatrice Lazzerini

Department of Information Engineering

University of Pisa

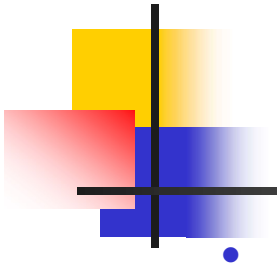
ITALY

Types of fuzzy rules

1. Mamdani fuzzy rule

$$if\ X_1\ is\ A_{i1}\ and\ \ldots\ and\ X_m\ is\ A_{im}\ then\ Y\ is\ B_i$$

- The rule consequent is a fuzzy set.
- Advantage:
  high interpretability
- Drawbacks:
  low accuracy
  high computational cost

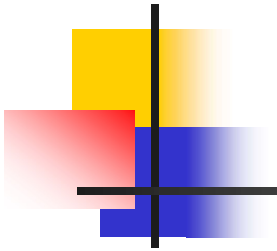


2. Takagi-Sugeno-Kang (TSK) fuzzy rule

$$if\ X_1\ is\ A_{i1}\ and\ \ldots\ and\ X_m\ is\ A_{im}\ then$$
$$Y\ is\ a_{i0} + a_{i1}\ X_1 + \ldots + a_{im}\ X_m$$

where a_{i0}, a_{i1}, …, a_{im} are real numbers.

- The rule consequent is a function – usually **linear** – of the input variables. Therefore, each rule can be considered as a *local linear model*.
- The system output is the weighted average of the output of each rule (better explained in the following slides).
- Advantage: greater accuracy than Mamdani systems
- Drawback: low interpretability



Coming back to the tourist-prediction example

- Let us convert the system to a TSK system by replacing the three linguistic consequents – *Low*, *Medium* and *High* – with corresponding, accurately calculated linear functions of the input variables
- The three rule consequents become:

Rule 1: **if** … **then** Tourists = $f_1(T,S) = 2T + 0.8S - 40$

Rule 2: **if** … **then** Tourists = $f_2(T,S) = T + S - 23$

Rule 3: **if** … **then** Tourists = $f_3(T,S) = 0.5T + 0.3S$

- With these consequents, the implication step produces the following predictions:

	Rule 1	Rule 2	Rule 3
Truth level μ_{Rule}	0.2	0.67	0.33
Estimated tourists	$f_1(19,60)=46\%$	$f_2(19,60)=56\%$	$f_3(19,60)=27.5\%$

- The aggregation step does not change these values. Hence, the defuzzifier produces:

$$output = \frac{\sum_{i=1}^{3} f_i(19,60)\cdot \mu_{\text{Rule}i}}{\sum_{i=1}^{3} \mu_{\text{Rule}i}}$$

$$= \frac{0.2\times 46\% + 0.67\times 56\% + 0.33\times 27.5\%}{0.2+0.67+0.33}$$

$$= 46.4\%$$

- **3. Singleton fuzzy rule**
- The rule consequent is a *constant value*.
- A singleton fuzzy rule can be considered as a special case of a Mamdani fuzzy rule or a special case of a TSK fuzzy rule. In fact, a constant value is equivalent to both a singleton fuzzy set and a linear function in which the coefficients of the input variables are 0.
- The singleton representation constitutes a compromise between the interpretability offered by Mamdani systems with their meaningful labels and the accuracy provided by TSK systems with their linear functions.
- Furthermore, thanks to the discrete representation of the output variable, the defuzzification process requires less computation than in the other two cases.

Coming back to the tourist-prediction example

- The Mamdani-type system is easily converted to a singleton-type system by replacing the fuzzy values *Low*, *Medium* and *High* with their corresponding centers of areas: 0%, 50% and 100%.
- With these new consequents, the implication step produces:

	Rule 1	Rule 2	Rule 3
Truth level μ_{Rule}	0.2	0.67	0.33
Estimated tourists	High=100%	Medium=50%	Low=0%

- The defuzzifier produces:

$$output = \frac{\sum_{i=1}^{3} f_i(19,60)\cdot \mu_{\text{Rule}i}}{\sum_{i=1}^{3} \mu_{\text{Rule}i}}$$

$$= \frac{0.2\times 100\% + 0.67\times 50\% + 0.33\times 0\%}{0.2+0.67+0.33}$$

$$= 44.4\%$$

- So far we have considered only membership functions that have been fixed, and somewhat arbitrarily chosen. Furthermore, we have only applied fuzzy inference to modeling systems whose rule structure is essentially predetermined by the user's interpretation of the characteristics of the variables in the model.

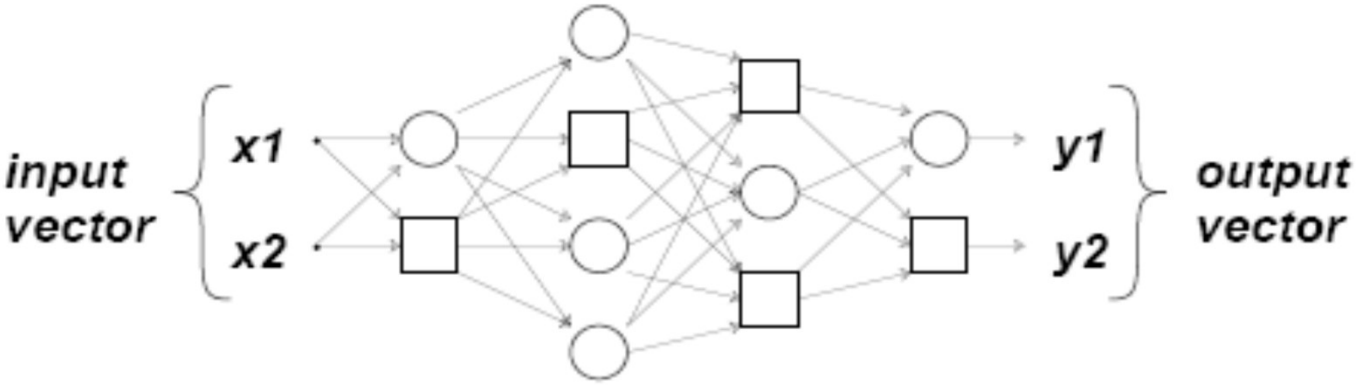
A new modeling scenario

- We have a collection of input/output data that we would like to use for modeling.
- We do not necessarily have a predetermined model structure based on the characteristics of the variables in our system.
- Rather than choosing the parameters associated with a given membership function arbitrarily, these parameters could be chosen so as to tailor the membership functions to the input/output data.

➔ We introduce Adaptive Networks

Adaptive networks

An *adaptive network* is a multi-layer feedforward network in which each node performs a particular function (*node function*) on incoming signals.



- A *square* node (*adaptive* node) has parameters.
- A *circle* node (*fixed* node) has no parameters.
- The links indicate only the flow direction of signals between nodes; no weights are associated with the links.
- In order to achieve a desired input-output mapping, the parameters are updated according to given training data and a gradient-descent learning procedure.

ANFIS (Adaptive-Network-based Fuzzy Inference Systems)

- *ANFIS* refers to a class of adaptive networks that are functionally equivalent to fuzzy inference systems.
- Suppose that the fuzzy inference system under consideration has two inputs *X* and *Y* and one output *f*. Suppose the rule base contains two fuzzy if-then rules of TSK type:

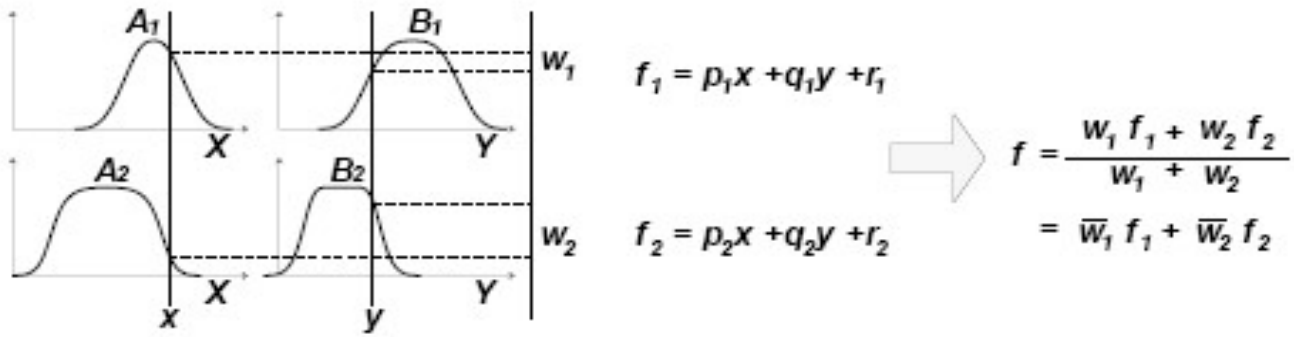
$$\text{Rule 1: } \textit{if } X \textit{ is } A_1 \textit{ and } Y \textit{ is } B_1 \textit{ then } f_1 = p_1 x + q_1 y + r_1$$

$$\text{Rule 2: } \textit{if } X \textit{ is } A_2 \textit{ and } Y \textit{ is } B_2 \textit{ then } f_2 = p_2 x + q_2 y + r_2$$

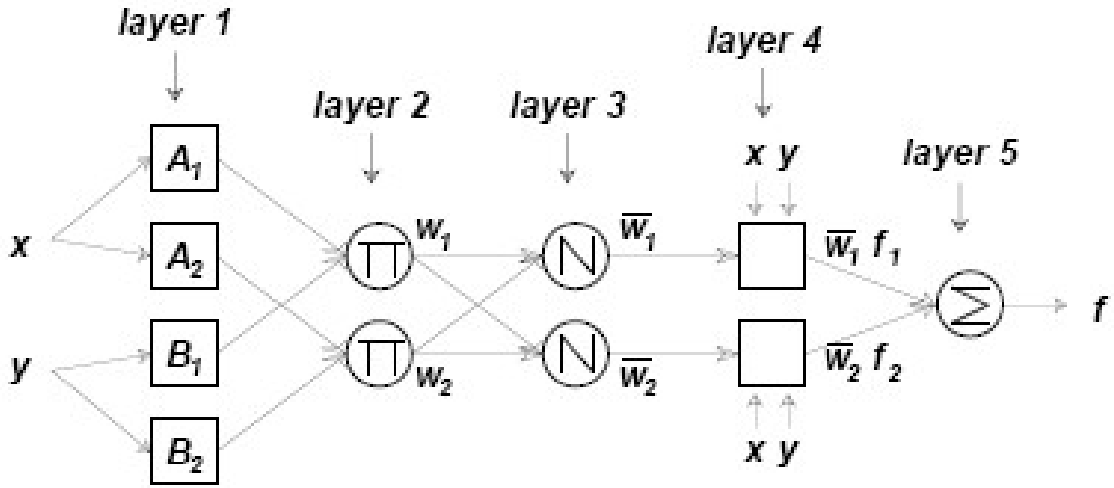
Rule 1: *if* X *is* A_1 *and* Y *is* B_1 *then* $f_1 = p_1 x + q_1 y + r_1$

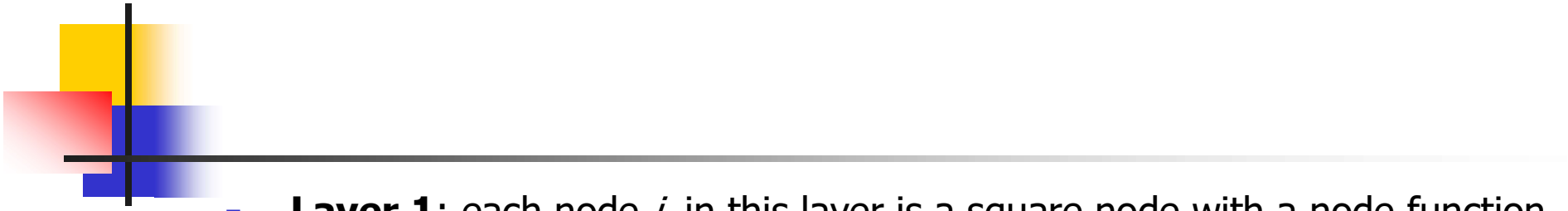
Rule 2: *if* X *is* A_2 *and* Y *is* B_2 *then* $f_2 = p_2 x + q_2 y + r_2$

- Fuzzy reasoning



- ANFIS architecture





- **Layer 1**: each node *i* in this layer is a square node with a node function

$$O_i^1 = \mu_{A_i}(x)$$

The parameters of the membership functions are called *premise parameters*.

- **Layer 2**: each node *i* in this layer is a circle node labeled $\prod$, which multiplies the incoming signals and sends the product (or other T-norm) out, e.g.,

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

Each node output represents the firing strength of a rule.

- **Layer 3**: each node *i* in this layer is a circle 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:

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

The outputs of this layer are called *normalized firing strengths*.

- **Layer 4**: each node *i* in this layer is a square node with a node function

$$O_i^4 = \overline{w}_i f_i = \overline{w}_i (p_i x + q_i y + r_i)$$

where $\overline{w}_i$ is the output of layer 3 and p_i, q_i, r_i are the so-called *consequent parameters*.

- **Layer 5**: the single node in this layer is a circle node labeled $\sum$ which computes the overall output as the summation of all incoming signals, i.e.,

$$O_1^5 = \text{overall output} = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

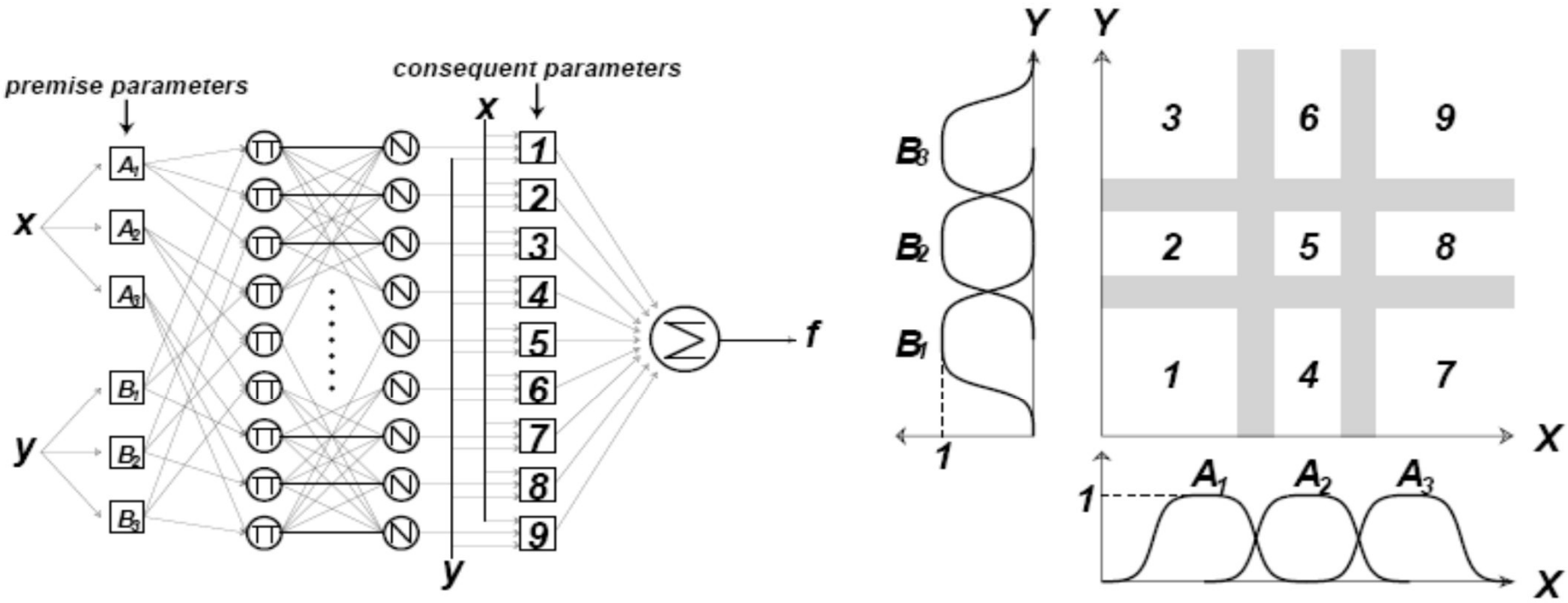
➔ Thus, we have built an adaptive network that is functionally equivalent to a TSK-type fuzzy inference system.

Hybrid learning algorithm

- A faster hybrid learning algorithm, which combines the gradient method and the least squares estimate (LSE), can be used instead of a gradient-descent learning procedure.
- Each epoch of the hybrid learning algorithm consists of two steps:
- in the *forward step*, given the values of the premise parameters, we supply the input data and the functional signals go forward till layer 4 to calculate each node output, and the consequent parameters are identified by the *least squares estimate*;
- in the *backward step*, the error signals propagate backward and the premise parameters are updated by the *gradient descent*.

-	forward pass	backward pass
premise parameters	fixed	gradient descent
consequent parameters	least squares estimate	fixed
signals	node outputs	error rates

Example

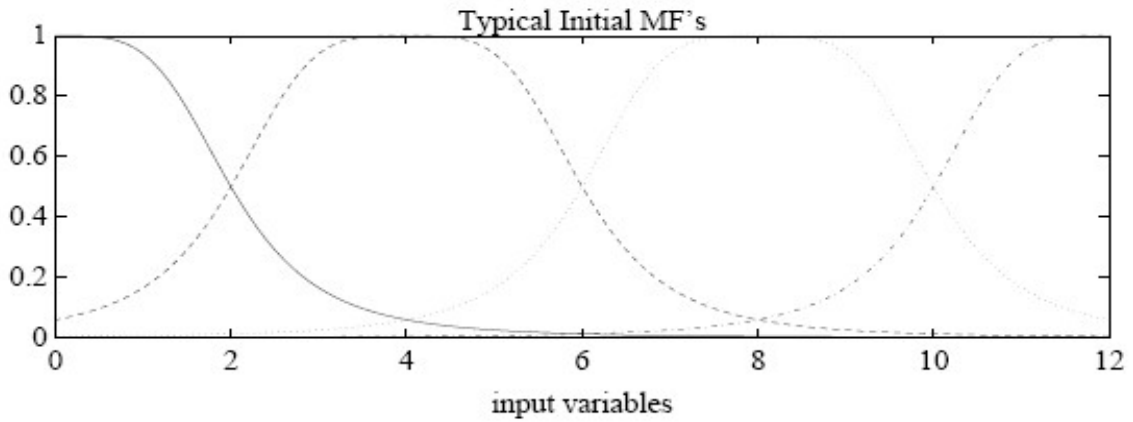


2-input ANFIS with 9 rules

3 membership functions for each input ➔ the input space is partitioned into 9 fuzzy subspaces, each governed by a rule: the antecedent of a rule defines a subspace while the consequent specifies the output within this subspace.

Typical initial membership function setting

- The number of membership functions is chosen empirically and/or by trial and error. Typically, the initial values of MFs are equally spaced. Moreover, the fuzzy system provides a smooth transition and sufficient overlapping from one linguistic label to another.



Modeling a two-input nonlinear function

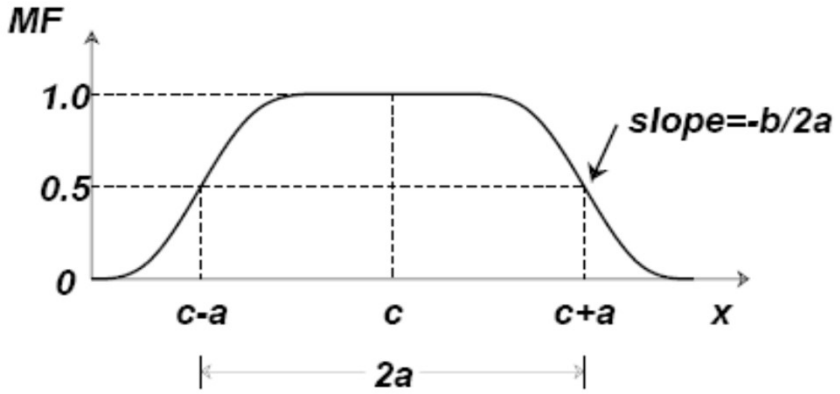
- Use ANFIS to model a nonlinear *sinc* equation

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

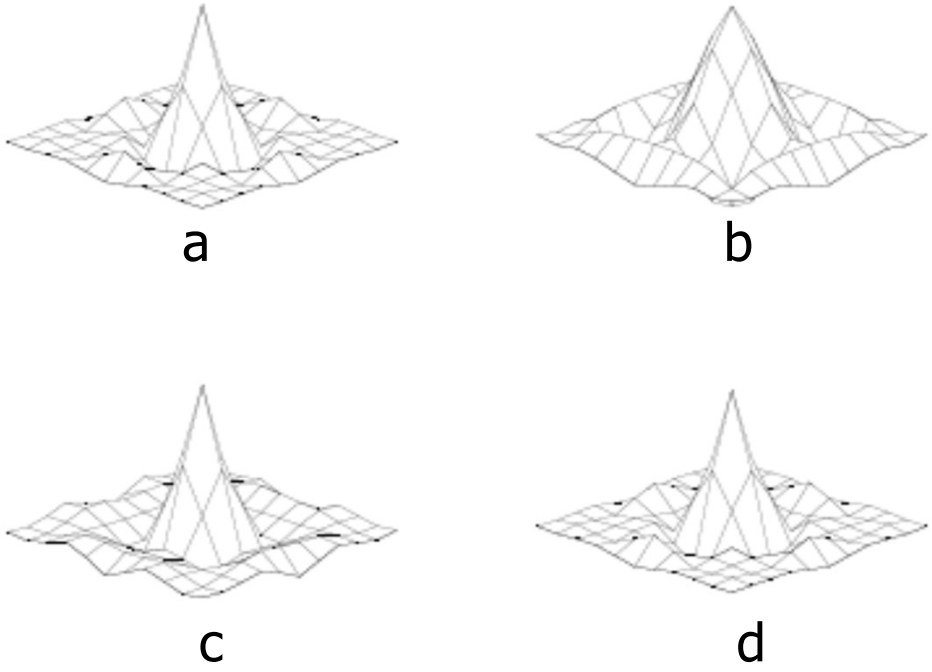
- From the grid points of the range $[-10,10]\times[-10,10]$, 121 training data pairs are obtained first.

- As a membership function, we use the **bell function**

$$\mu_A(x) = \frac{1}{1 + \left[\left(\frac{x-c}{a}\right)^2\right]^b}$$



- The bell function contains 3 parameters:
  - *c* is the center of the function,
  - *a* is the half width,
  - *b* (together with *a*) controls the slope at the crossover points (where MF value = 0.5).
- The used ANFIS contains 16 rules, with 4 membership functions for each input variable ➔ # fitting parameters = 72 (24 premise parameters and 48 consequent parameters).



- Training data (a) and reconstructed surfaces at 0.5 (b), 99.5 (c) and 249.5 (d) epochs.

  (Note that since the error is computed after the forward pass, the epoch numbers always end with ".5". Further, the surface after 0.5 epochs is only due to the identification of consequent parameters).

Initial and final membership functions