Artificial Intelligence

Fuzzy Logic

Lesson 5: Neuro-Fuzzy Systems

Vincenzo Piuri

Università degli Studi di Milano

Contents

- Neuro-fuzzy systems
- From fuzzy rules to network structure
- Neuro-fuzzy control
- Neuro-fuzzy classification

Comparison of Neural Networks and Fuzzy System

Neural Networks	Fuzzy Systems
are low-level computational structures	deal with reasoning on a higher level
perform well when enough data are present	use linguistic information from domain experts
can learn	neither learn nor adjust themselves to new environment
are black-boxes for the user	are based on natural language

- Neuro-fuzzy systems shall combine the parallel computation and learning abilities of neural networks with the human-like knowledge representation and explanation abilities of fuzzy systems
- As a result, neural networks become more transparent, while fuzzy systems become capable of learning

Neuro-Fuzzy Systems (1)

- **Cooperative** models
  - A neural network and a fuzzy controller work independently
  - The neural network generates (offline) or optimizes (online) certain parameters

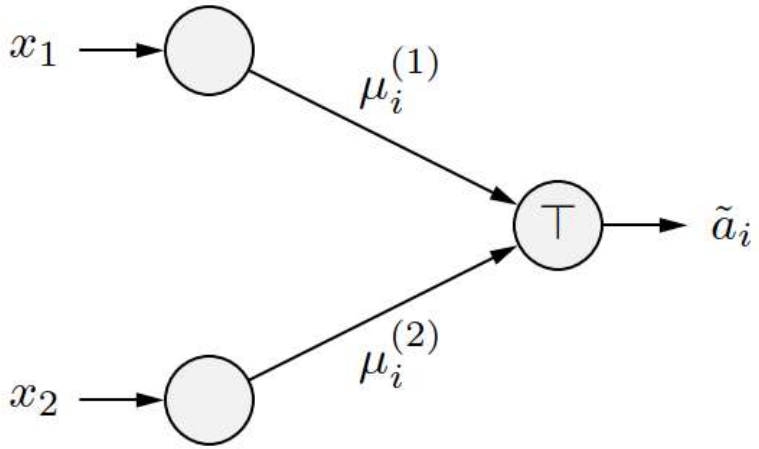
Neuro-Fuzzy Systems (2)

- **Hybrid** models
  - Combine the structure of a neural network and a fuzzy controller
    - Map fuzzy sets and fuzzy rules to a neural network structure
  - A hybrid neuro-fuzzy system can be interpreted as a neural network and can be implemented with the help of a neural network
  - Advantages
    - Integrated structure
    - No communication between two different models is needed
    - Both offline and online training are possible

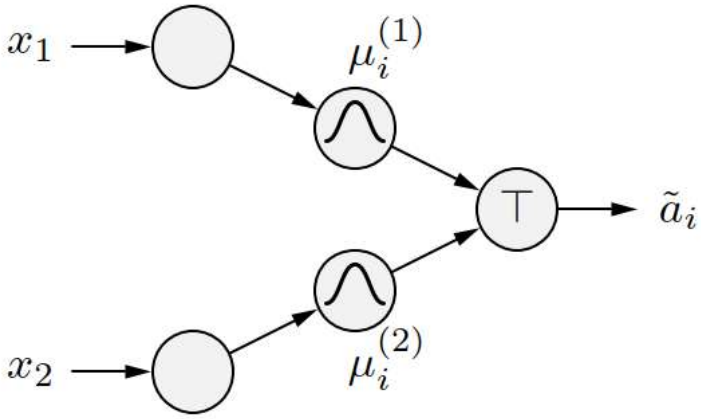
Neuro-Fuzzy Systems (3)

- The fuzzy sets appearing in the antecedents of fuzzy rules can be modeled in different ways

As connection weights

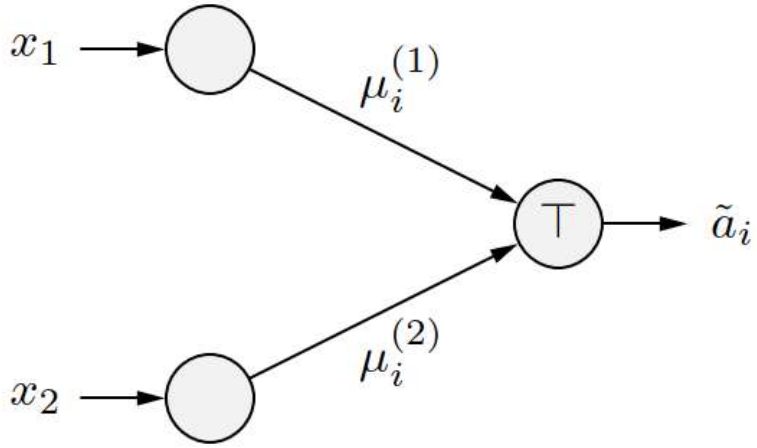


As activation functions



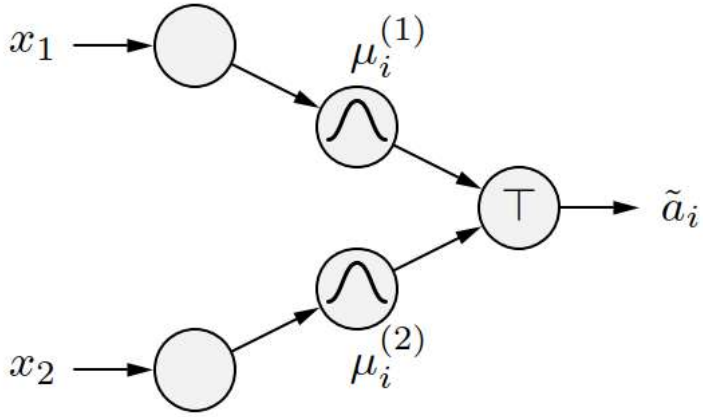
Neuro-Fuzzy Systems (4)

As connection weights



The neurons in the first hidden layer represent the rule antecedents and the connections from the input units represent the fuzzy sets of these antecedents

As activation functions



The neurons in the first hidden layer represent the fuzzy sets and the neurons in the second hidden layer represent the rule antecedents

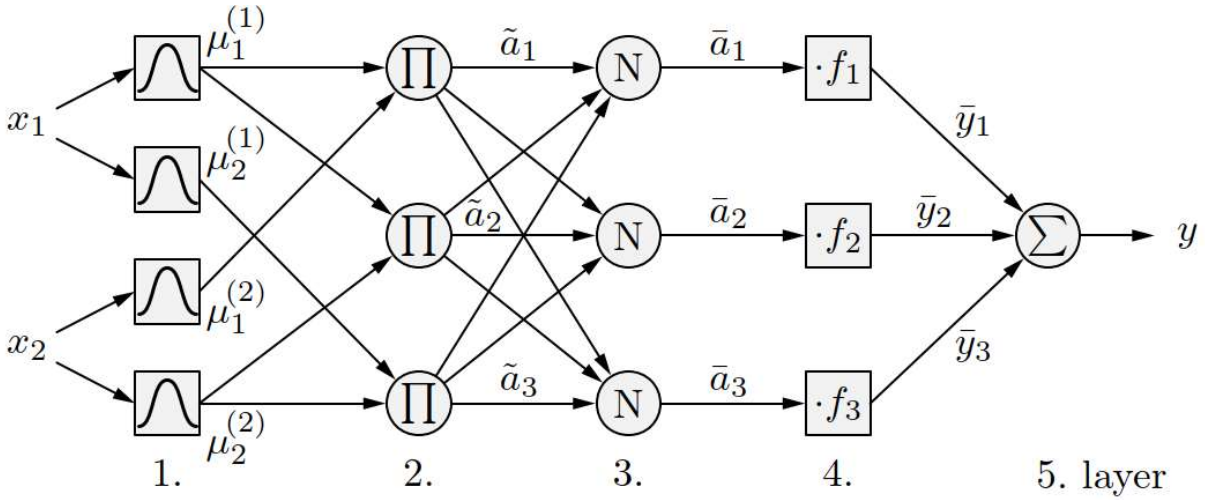
From Fuzzy Rules to Network Structure (1)

- A fuzzy rule base is turned into a network structure as follows
  1. For every input variable x_i: create a neuron in the input layer
  2. For every output variable y_i: create a neuron in the output layer
  3. For every fuzzy set $\mu_i^{(j)}$: create a neuron in the first hidden layer and connect it to the input neuron corresponding to x_i
  4. For every fuzzy rule R_i: create a (rule) neuron in the second hidden layer and specify a t-norm for computing the rule (antecedent) activation
  5. Connect each rule neuron to the neurons that represent the fuzzy sets of the antecedent of its corresponding fuzzy rule R_i

From Fuzzy Rules to Network Structure (2)

- Mamdani–Assilian controller
  6. Connect each rule neuron to the output neuron corresponding to the consequent domain of its fuzzy rule. As connection weight choose the consequent fuzzy set of the fuzzy rule
- Takagi–Sugeno–Kang controller
  6. For each rule neuron, create a sibling neuron that computes the output function of the corresponding fuzzy rule and connect all input neurons to it (arguments of the consequent function)
- Resulting network structure can be trained with error backpropagation

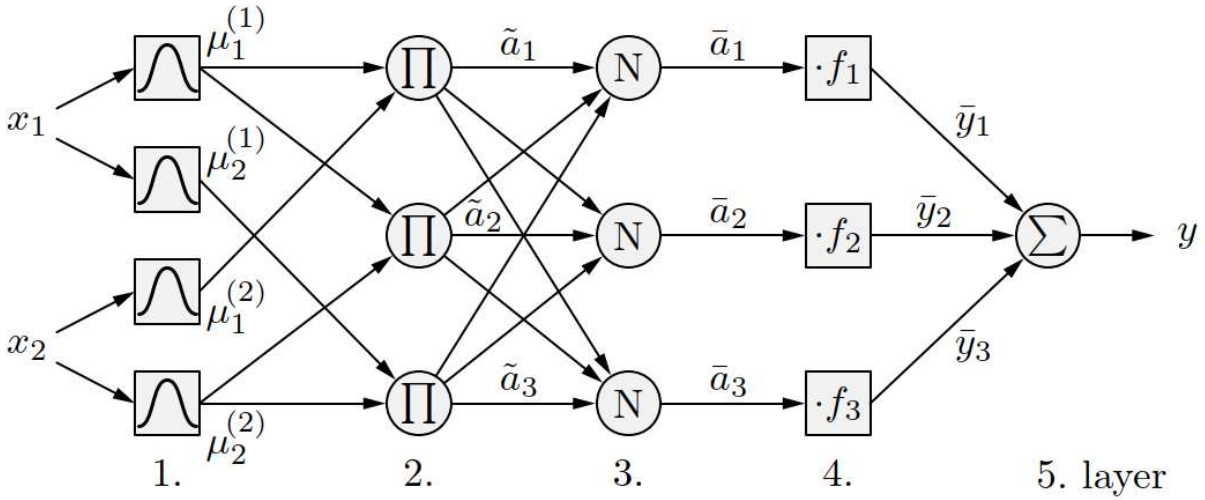
From Fuzzy Rules to Network Structure (3)



(Connections from inputs to output function neurons for f_1, f_2, f_3 not shown.)

- Adaptive Network-based Fuzzy Inference Systems (ANFIS)
  - This ANFIS network represents the fuzzy rule base (Takagi–Sugeno–Kang rules)

From Fuzzy Rules to Network Structure (4)



(Connections from inputs to output function neurons for f_1, f_2, f_3 not shown.)

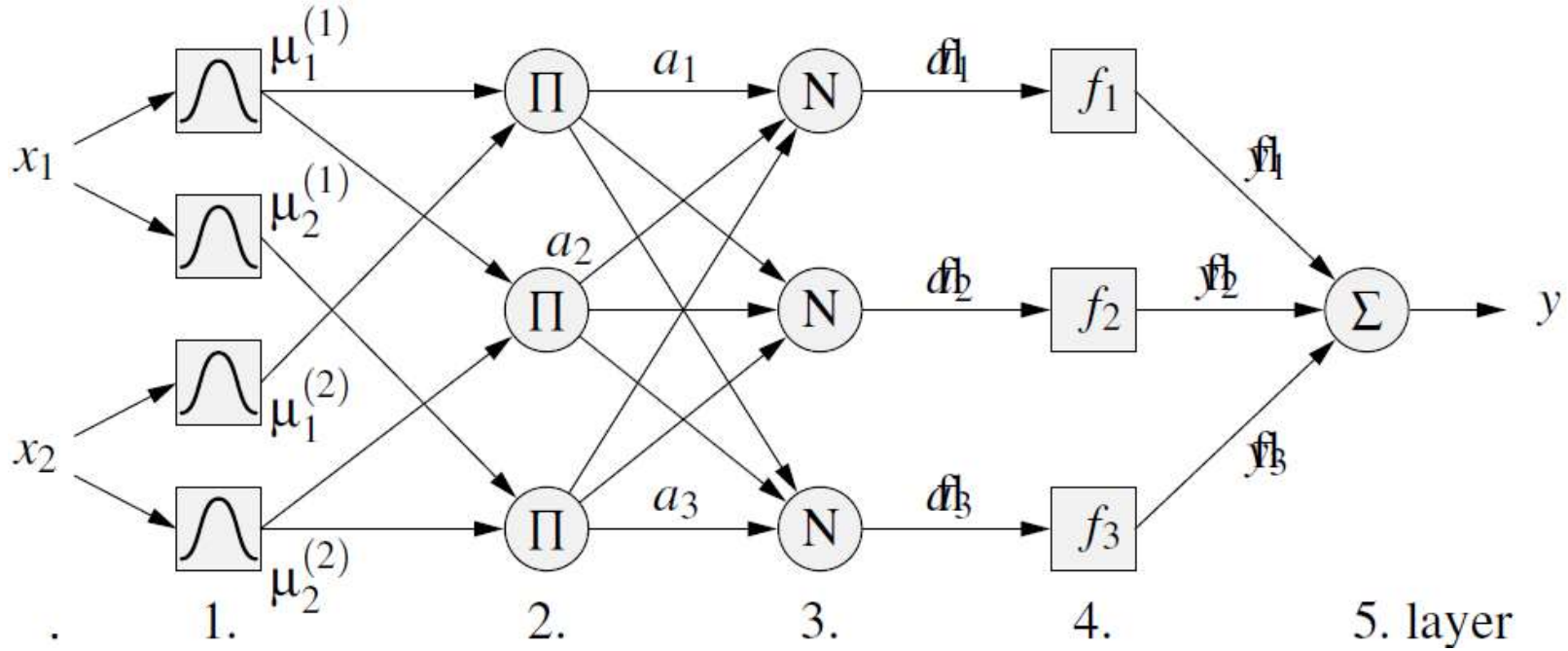
- Adaptive Network-based Fuzzy Inference Systems (ANFIS)

$$
\begin{aligned}
R_1\text{: } & \textbf{if } x_1 \text{ is } \mu_1^{(1)} \textbf{ and } x_2 \text{ is } \mu_1^{(2)}, \textbf{ then } y = f_1(x_1, x_2) \\
R_2\text{: } & \textbf{if } x_1 \text{ is } \mu_1^{(1)} \textbf{ and } x_2 \text{ is } \mu_2^{(2)}, \textbf{ then } y = f_2(x_1, x_2) \\
R_3\text{: } & \textbf{if } x_1 \text{ is } \mu_2^{(1)} \textbf{ and } x_2 \text{ is } \mu_2^{(2)}, \textbf{ then } y = f_3(x_1, x_2)
\end{aligned}
$$

ANFIS Model

- Adaptive-Network-based Fuzzy Inference System
- Integrates fuzzy rules into a neural structure

Example of an ANFIS Model



- This is a model with three fuzzy rules
  - R_1 : If x_1 is A_1 and x_2 is B_1 then $y = f_1(x_1, x_2)$
  - R_2 : If x_1 is A_2 and x_2 is B_2 then $y = f_2(x_1, x_2)$
  - R_3 : If x_1 is A_1 and x_3 is B_3 then $y = f_3(x_1, x_2)$

  with linear output functions $f_i = p_i x_1 + q_i x_2 + r_i$ in the antecedent part

ANFIS: Layer 1
The Fuzzification Layer

- Neurons represent fuzzy sets of the fuzzy rule antecedents
- The activation function of a membership neuron is set to the function that specifies the neuron's fuzzy set
- A fuzzification neuron receives a crisp input and determines the degree to which this input belongs to the neuron's fuzzy set
- Usually bell-curved functions are used, e.g.

$$\mu_{\mathrm{i}}^{(j)} x_j = \frac{1}{1 + \left(\frac{x_j - a_i}{b_i} \right)^{2c_i}}$$

  where a_i, b_i, c_i are parameters for center, width, and slope, respectively
- The output of a fuzzification neuron thus also depends on the membership parameters

ANFIS: Layer 2
The Fuzzy Rule Layer

- Each neuron corresponds to a single Takagi-Sugeno fuzzy rule
- A fuzzy rule neuron receives inputs from the fuzzification neurons that represent fuzzy sets in the rule antecedents
- It calculates the firing strength of the corresponding rule.
- The intersection is usually implemented by the product
- So, the firing strength $\widetilde{a}_i$ of rule R_i is

$$\widetilde{a}_i = \prod_{j=1}^{k} \mu_{\mathrm{i}}^{(j)} x_j$$

ANFIS: Layer 3
The Normalization Layer

- Each neuron in this layer receives the firing strengths from all neurons in the rule layer
- The normalized firing strength of a given rule is calculated here
- It represents the contribution of a given rule to the final result
- Thus, the output of neuron i in layer 4 is determined as

$$\bar{a}_i = a_i = net_i = \frac{\tilde{a}_i}{\sum_j \tilde{a}_j}$$

ANFIS: Layers 4 and 5 Defuzzification and Summation

- Each neuron in layer 4 is connected to the respective normalization neuron, and also receives the raw input values $\boldsymbol{x}$
- A defuzzification neuron calculates the weighted consequent value of a given rule as

$$\overline{y_i} = a_i = \text{net}_i = \overline{a_i} f_i(x_1, \ldots, x_n)$$

- The single neuron in layer 5 calculates the sum of outputs from all defuzzification neurons and produces the overall ANFIS output

$$y = f(\boldsymbol{x}_i) = a_{\text{out}} = \text{net}_{\text{out}} = \sum_i \overline{y_i} = \frac{\sum_i \widetilde{a_i} f_i(x_1, \ldots, x_n)}{\sum_i \widetilde{a_i}}$$

How does ANFIS learn? The Forward Pass (1)

- ANFIS uses a hybrid learning algorithm that combines least-squares and gradient descent
- Each learning epoch is composed of one forward and one backward pass
- In the forward pass, a training set of input-output tuples $(\boldsymbol{x}_k, y_k)$ is presented to the ANFIS, neuron outputs are calculated on the layer-by-layer basis, and rule consequent parameters are identified by least squares
- Goal: minimize mean squared error

$$e = \sum_{i=1}^{m} \left| y(k) - f\left(\boldsymbol{x}(k)\right) \right|^2$$

How does ANFIS learn? The Forward Pass (2)

- r_{ij} : parameters of output function f_i , $x_i(k)$: input values, $y(k)$: output value of *k*-th training pair, $\overline{a_i}(k)$: relative control activation
- Then we obtain

$$y(k) = \sum_i \overline{a_i}(k) y_i(k) = \sum_i \overline{a_i}(k) \left(\sum_{j=1}^{n} r_{ij} x_j(k) + r_{i0} \right), \qquad \forall i \, , k$$

- Therefore, with $\hat{x}_i(k) := [1, x_1(k), \ldots, x_n(k)]^T$ we obtain the overdetermined linear equation system

$$\boldsymbol{y} = \overline{\boldsymbol{a}} \boldsymbol{R} \boldsymbol{X}$$

  for $m > (n + 1) \cdot r$ with m number of training points, r number of rules, n number of input variables

- The consequent parameters are adjusted while the antecedent parameters remain fixed

How does ANFIS learn? The Backward Pass

- In the backward pass, the error is determined in the output units based on the new calculated output functions
- Also, with the help of gradient descent, the parameters of the fuzzy sets are optimized
- Back propagation is applied to compute the "error" of the neurons in the hidden layers
- It updates the parameters of these neurons by the chain rule

ANFIS: Summary

- Forward and backward passes improves convergence
- Reason: Least squares already has an optimal solution for the parameters of the output function w.r.t. the initial fuzzy sets
- Unfortunately ANFIS has no restrictions for the optimization of the fuzzy sets in the antecedents
  - So, after optimization the input range might not be covered completely with fuzzy sets
  - Thus definition gaps can appear which have to be checked afterwards
- Fuzzy sets can also change, independently form each other, and can also exchange their order and so their importance, too
- We have to pay attention to this, especially if an initial rule base was set manually and the controller has to be interpreted afterwards

Learning Fuzzy Sets (1)

- Gradient descent procedures are only applicable, if a differentiation is possible, e.g. for Sugeno-type fuzzy systems

- Applying special heuristic procedures that do not use any gradient information can facilitate the learning of Mamdani-type rules

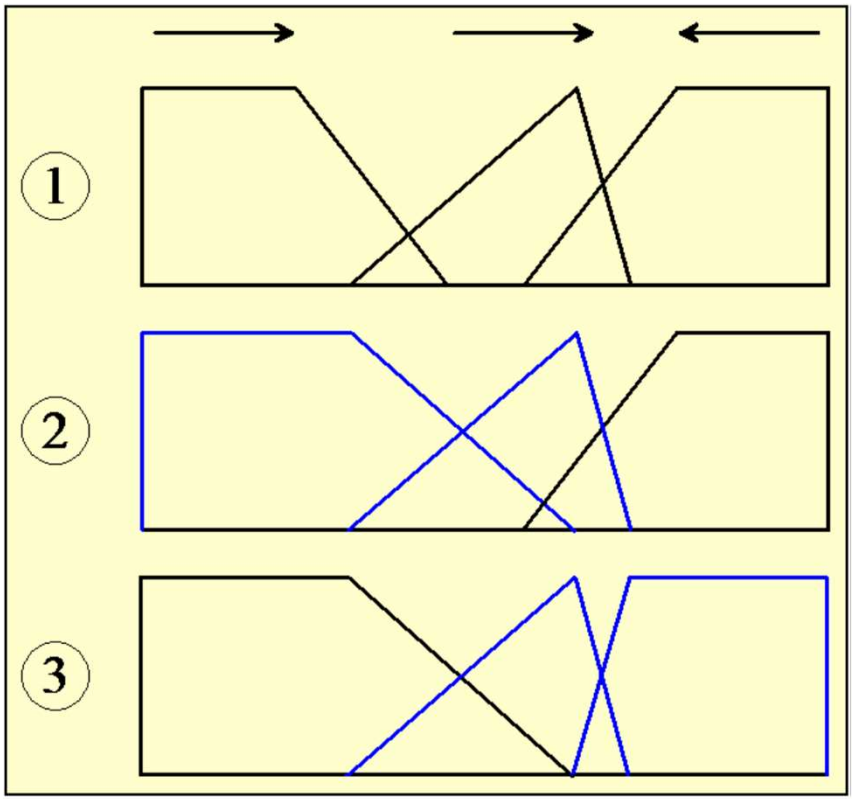
- Learning algorithms are based on the idea of backpropagation but constrain the learning process to ensure interpretability

Learning Fuzzy Sets (2)

- Mandatory constraints: Fuzzy sets must
  - stay normal and convex
  - not exchange their relative positions (they must not "pass" each other)
  - always overlap
- Optional constraints
  - Fuzzy sets must stay symmetric
  - The membership degrees must add up to 1
- A learning algorithm must enforce these constraints

Constraints for Training Fuzzy Sets

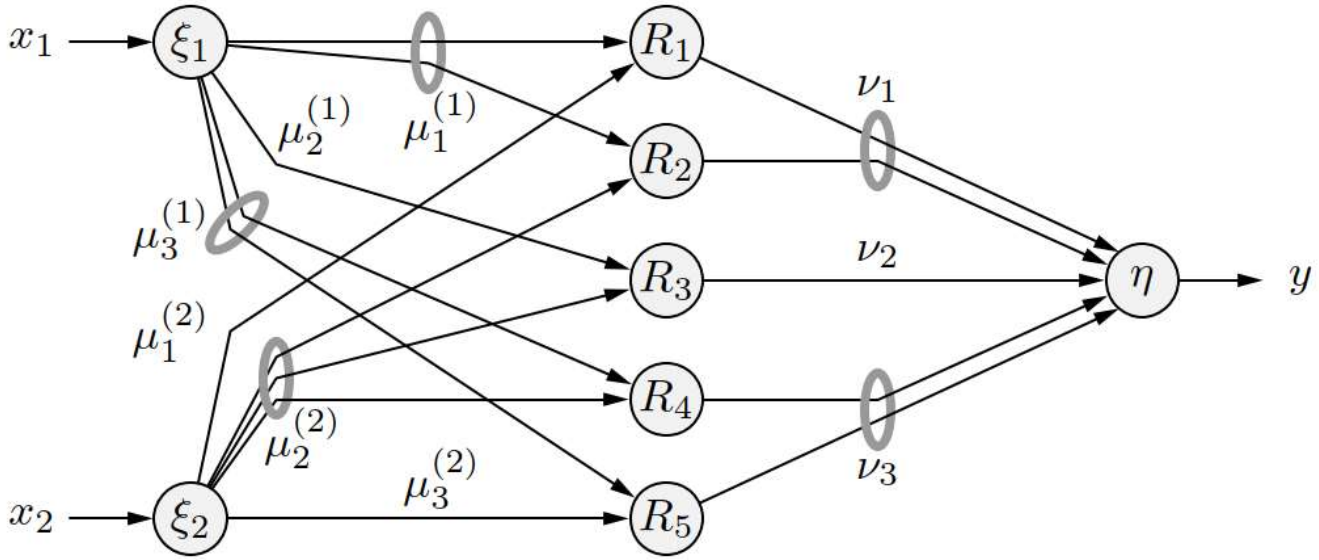
- valid parameter values
- non-empty intersection of adjacent fuzzy sets
- keep relative positions
- maintain symmetry
- complete coverage (degrees of membership add up to 1 for each element)



Correcting a partition after modifying the parameters

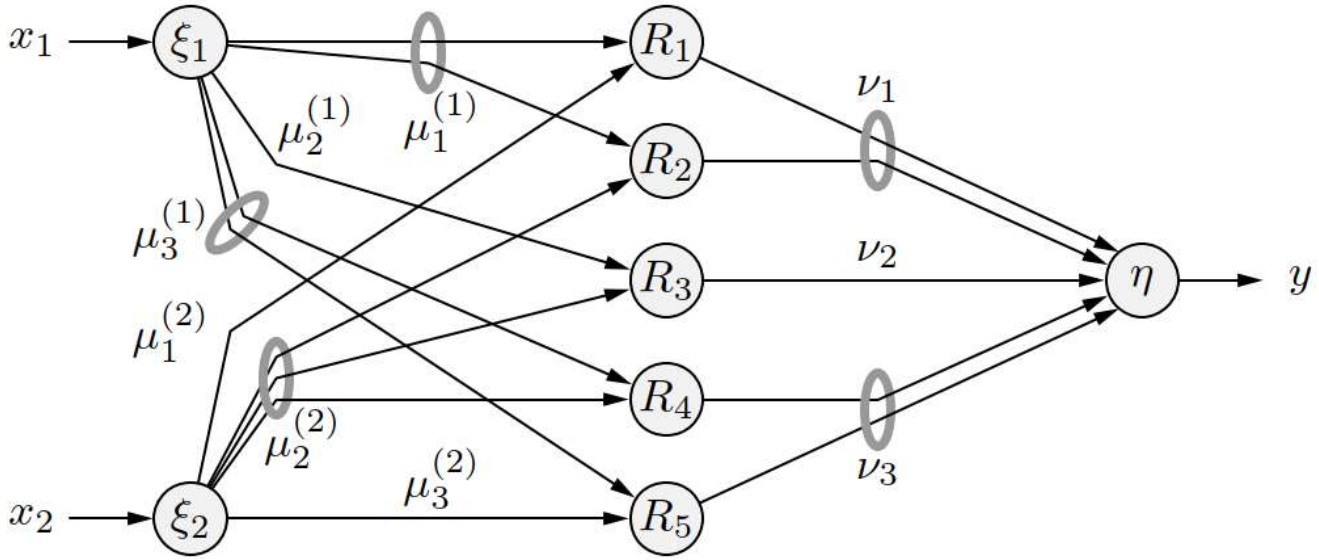
Neuro-Fuzzy Control

Neuro-Fuzzy Control (1)



- Neuro-Fuzzy Control (NEFCON)
  - This NEFCON network represents the fuzzy rule base (Mamdani–Assilian rules)

Neuro-Fuzzy Control (2)

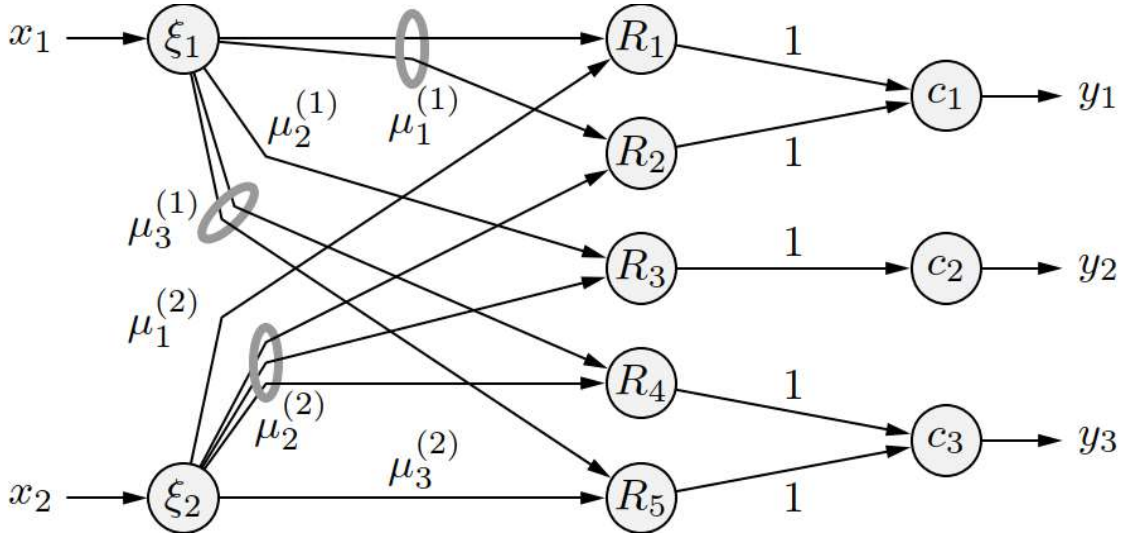


- Neuro-Fuzzy Control (NEFCON)

$$
\begin{array}{lllllll}
R_1\text{:} & \textbf{if} & x_1 \text{ is } \mu_1^{(1)} & \textbf{and} & x_2 \text{ is } \mu_1^{(2)}, & \textbf{then} & y \text{ is } \nu_1 \\
R_2\text{:} & \textbf{if} & x_1 \text{ is } \mu_1^{(1)} & \textbf{and} & x_2 \text{ is } \mu_2^{(2)}, & \textbf{then} & y \text{ is } \nu_1 \\
R_3\text{:} & \textbf{if} & x_1 \text{ is } \mu_2^{(1)} & \textbf{and} & x_2 \text{ is } \mu_2^{(2)}, & \textbf{then} & y \text{ is } \nu_2 \\
R_4\text{:} & \textbf{if} & x_1 \text{ is } \mu_3^{(1)} & \textbf{and} & x_2 \text{ is } \mu_2^{(2)}, & \textbf{then} & y \text{ is } \nu_3 \\
R_5\text{:} & \textbf{if} & x_1 \text{ is } \mu_3^{(1)} & \textbf{and} & x_2 \text{ is } \mu_3^{(2)}, & \textbf{then} & y \text{ is } \nu_3
\end{array}
$$

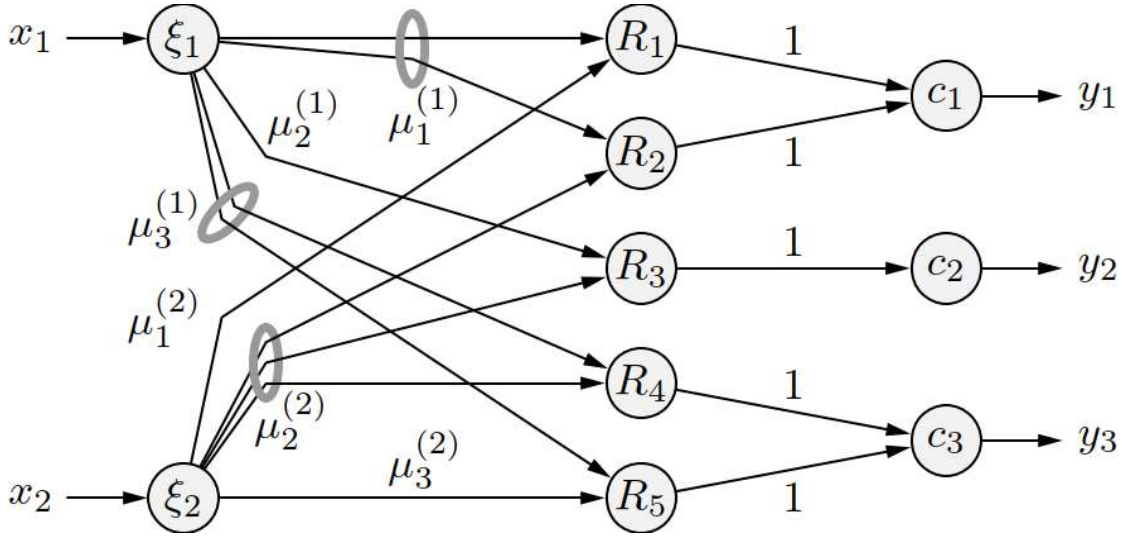
Neuro-Fuzzy Classification

Neuro-Fuzzy Classification (1)



- Neuro-Fuzzy Classification (NEFCLASS)
  - This NEFCLASS network represents fuzzy rules that predict classes

Neuro-Fuzzy Classification (2)

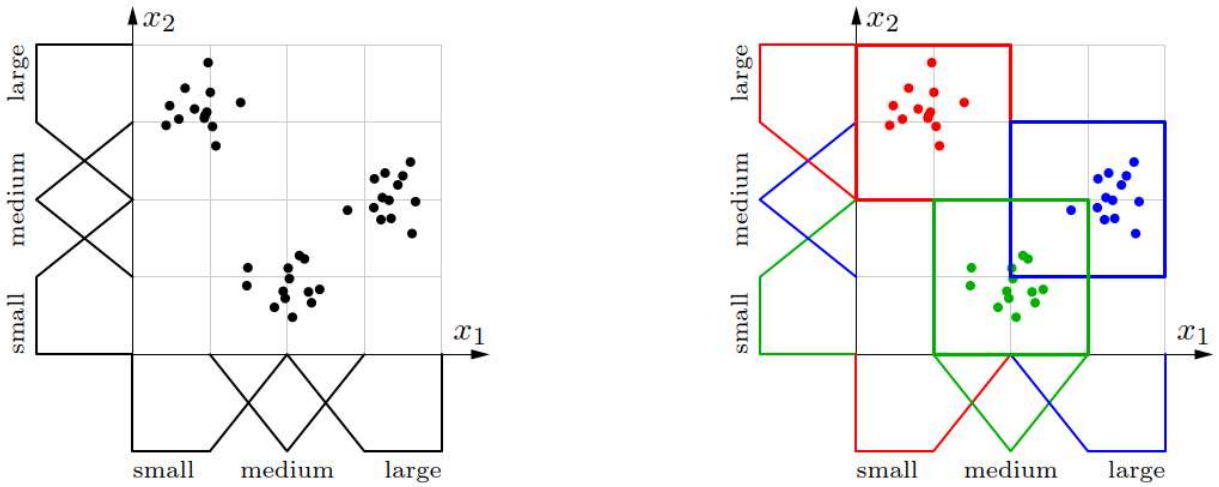


- Neuro-Fuzzy Classification (NEFCLASS)

$$
\begin{array}{lllllll}
R_1\text{:} & \textbf{if} & x_1 \text{ is } \mu_1^{(1)} & \textbf{and} & x_2 \text{ is } \mu_1^{(2)}, & \textbf{then} & \text{class } c_1 \\
R_2\text{:} & \textbf{if} & x_1 \text{ is } \mu_1^{(1)} & \textbf{and} & x_2 \text{ is } \mu_2^{(2)}, & \textbf{then} & \text{class } c_1 \\
R_3\text{:} & \textbf{if} & x_1 \text{ is } \mu_2^{(1)} & \textbf{and} & x_2 \text{ is } \mu_2^{(2)}, & \textbf{then} & \text{class } c_2 \\
R_4\text{:} & \textbf{if} & x_1 \text{ is } \mu_3^{(1)} & \textbf{and} & x_2 \text{ is } \mu_2^{(2)}, & \textbf{then} & \text{class } c_3 \\
R_5\text{:} & \textbf{if} & x_1 \text{ is } \mu_3^{(1)} & \textbf{and} & x_2 \text{ is } \mu_3^{(2)}, & \textbf{then} & \text{class } c_3
\end{array}
$$

Neuro-Fuzzy Classification (3)

- NEFCLASS first fuzzy partitions the domain of each variable
  - The boundary fuzzy sets are "shouldered" (membership 1 to the boundary)
  - Based on the initial fuzzy partitions, the initial rule base is selected



Neuro-Fuzzy Classification (4)

- In order to reduce/limit the number of initial rules, their performance is evaluated
  - Sort the rules in the initial rule base by their performance
  - Choose either the best r rules or the best r/c rules per class (c is the number of classes)
  - The number r of rules in the rule base is either provided by a user or is automatically determined in such a way that all patterns are covered

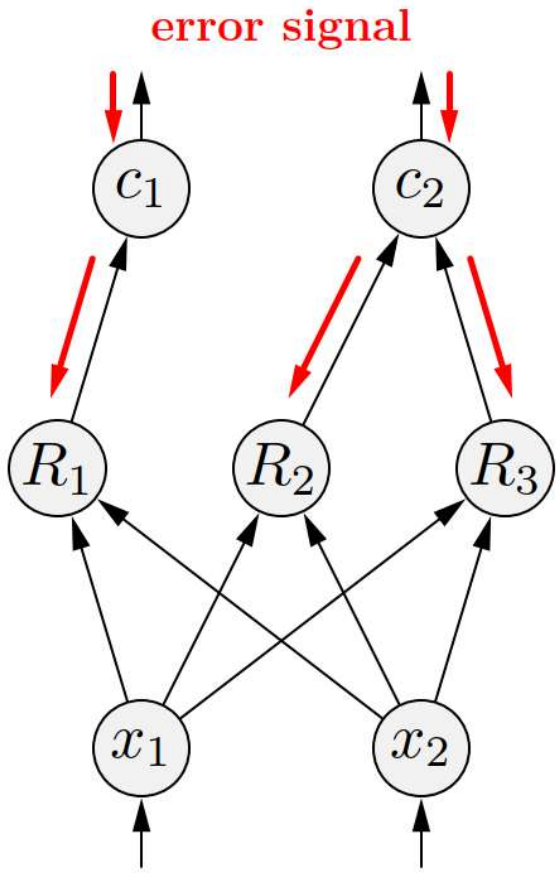
Neuro-Fuzzy Classification (5)

- Let $o_k^{(l)}$ the decided output and $out_k^{(l)}$ the actual output of the k-th output neuron (class c_k)
- Fuzzy error (k-th output class)

$$e_k^{(l)} = 1 - \gamma\left(\varepsilon_k^{(l)}\right),$$

- Error signal

$$\delta_k^{(l)} = \mathrm{sgn}\left(\varepsilon_k^{(l)}\right)\, e_k^{(l)}.$$



Neuro-Fuzzy Classification (6)

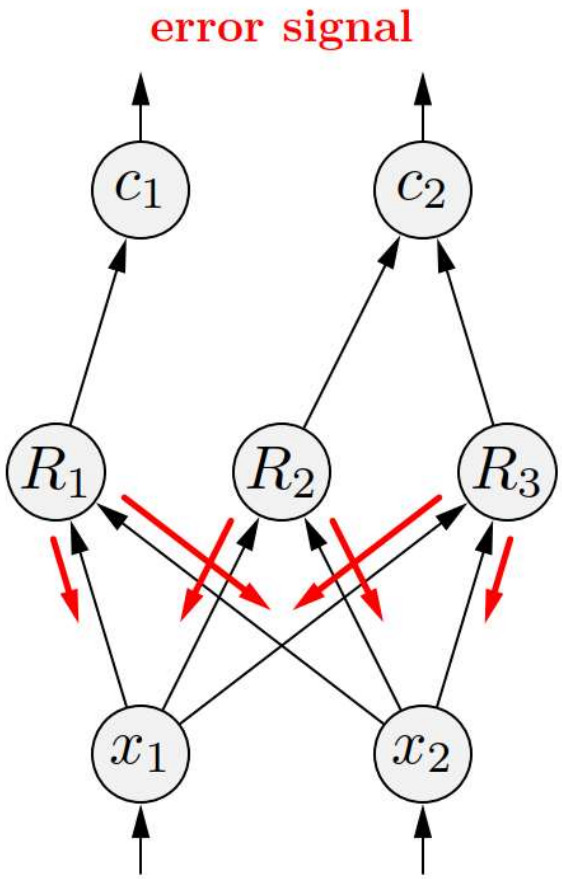
- Rule error signal (rule R_i for c_k)

$$\delta_{R_i}^{(l)} = \mathrm{out}_{R_i}^{(l)} \left(1 - \mathrm{out}_{R_i}^{(l)}\right) \delta_k^{(l)},$$

- Find input variable x_j such that

$$\mu_i^{(j)}\left(\imath_j^{(l)}\right) = \tilde{a}_i\left(\vec{\imath}^{(l)}\right) = \min_{\nu=1}^{d} \mu_i^{(\nu)}\left(\imath_\nu^{(l)}\right),$$

- Adapt parameters of the fuzzy set $\mu_i^{(j)}$



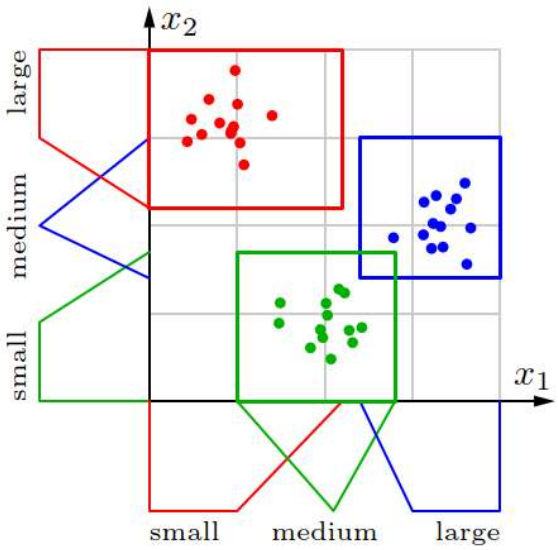
Neuro-Fuzzy Classification (7)

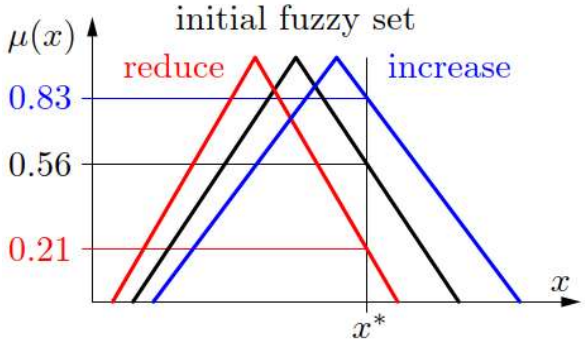
- Training the fuzzy sets (e.g., triangular fuzzy sets)

$$\mu_{a,b,c}(x) = \begin{cases} \frac{x-a}{b-a} & \text{if } x \in [a,b), \\ \frac{c-x}{c-b} & \text{if } x \in [b,c], \\ 0 & \text{otherwise.} \end{cases}$$

- Parameter changes

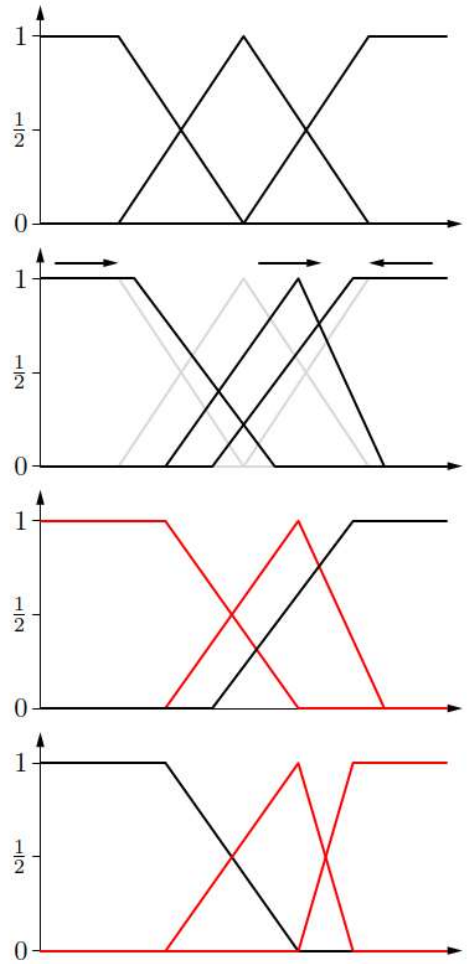
$$\begin{aligned} \Delta b &= +\eta \cdot \delta_{R_i}^{(l)} \cdot (c-a) \cdot \operatorname{sgn}\left(\imath_j^{(l)} - b\right) \\ \Delta a &= -\eta \cdot \delta_{R_i}^{(l)} \cdot (c-a) + \Delta b \\ \Delta c &= +\eta \cdot \delta_{R_i}^{(l)} \cdot (c-a) + \Delta b \end{aligned}$$





Neuro-Fuzzy Classification (8)

- Restrictions
  - Valid parameter values
  - Non-empty intersections of neighboring fuzzy sets
  - Preserve relative positions
  - Preserve symmetry
  - Partition of unity (membership degrees sum to 1 everywhere)



Neuro-Fuzzy Classification (9)