Hybrid Systems

Combining FS, NN and EC

Dr. Petr Musilek

Department of Electrical and Computer Engineering
University of Alberta

Fall 2019

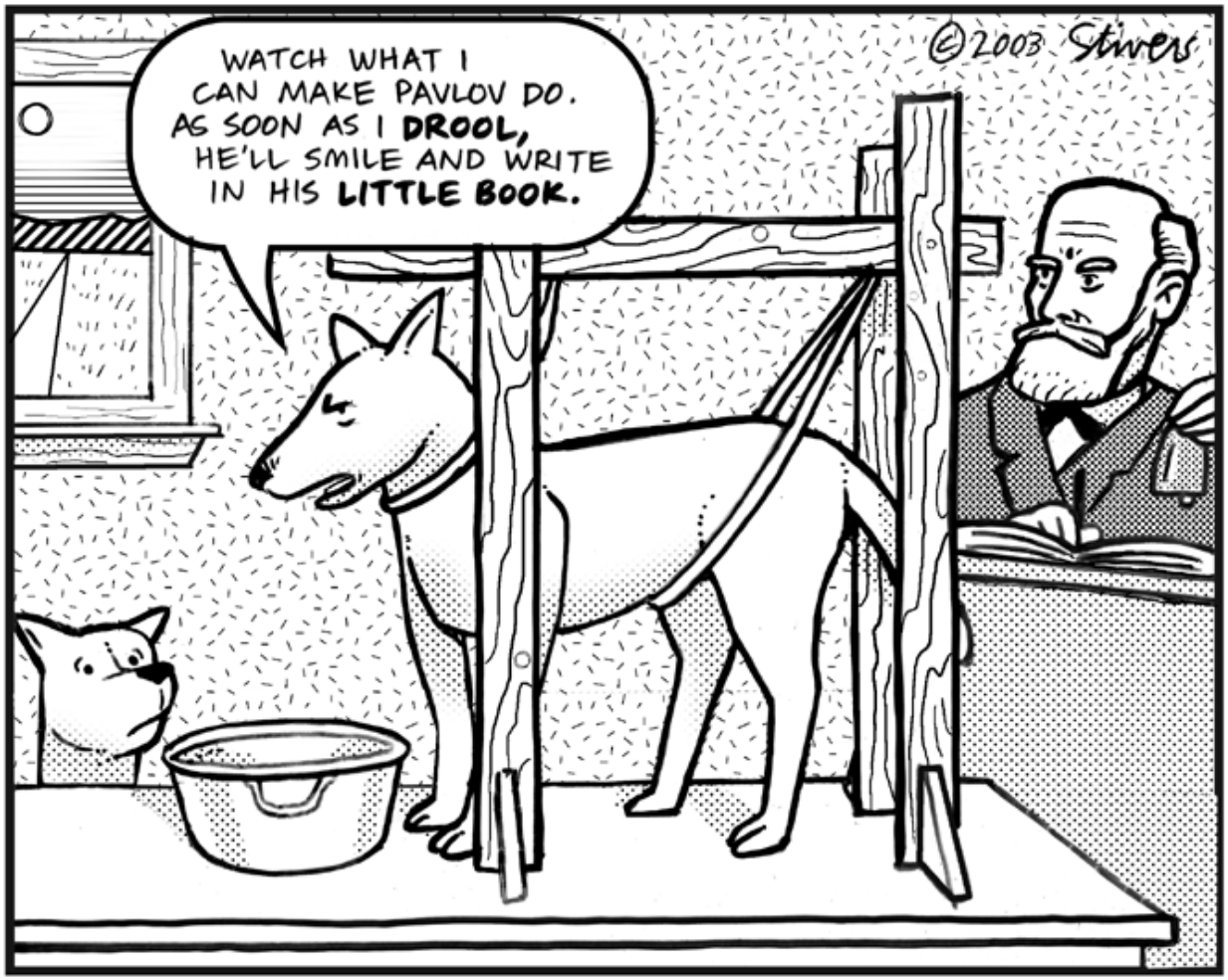
Neural Networks

Problem requirements:

- There must be sample data describing the problem
- If input-output data is available $\rightarrow$ Supervised Learning
- If input data only is available $\rightarrow$ Unsupervised Learning
- If input data and error measurement is available $\rightarrow$ Reinforcement Learning

Reinforcement Learning

We did not really talk about reinforcement learning (RL); here is a simple illustrative example:



Neural Networks

Pros and Cons:

- \+ No prior knowledge is required.
- – We can't interpret the solution, NN is a black-box.
- – We can't initialize NN with prior knowledge. (learning from scratch)
- \+ Fault tolerant to changes in input and network structure.
- – If problem data differs too much from training data $\rightarrow$ NN can't cope $\rightarrow$ retraining.
- – No guarantee for convergence $\rightarrow$ learn from scratch.
- \+ No need for mathematical (explicit, analytical) model.

Fuzzy Systems

Problem requirements:

- Prior knowledge about the problem in terms of IF-THEN rules.
- Defining suitable fuzzy sets describing the linguistic variables.

Fuzzy Systems

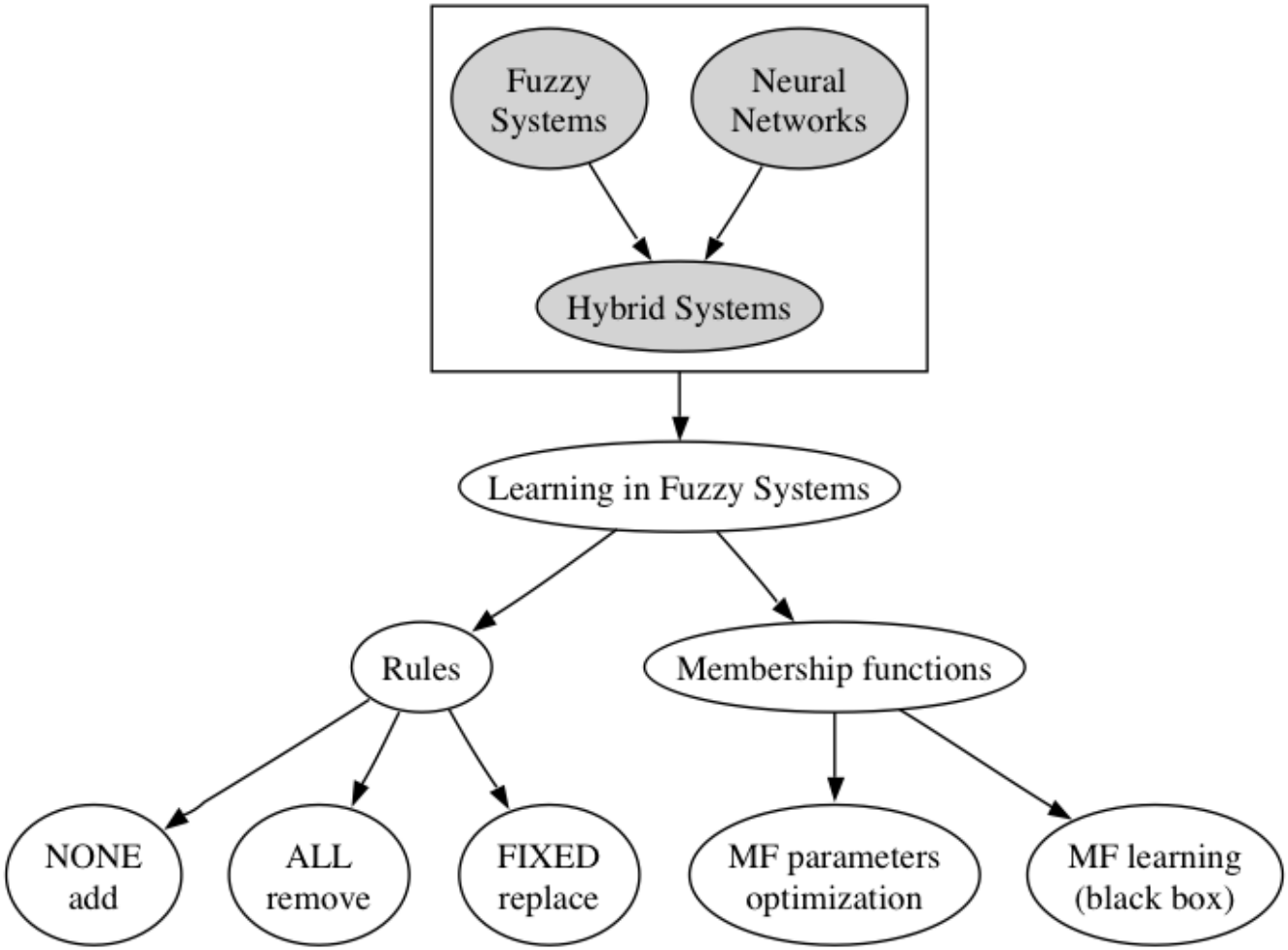
Pros and Cons

- \+ No need for mathematical model.
- \+ No need for training data.
- – Definition of fuzzy rules and fuzzy sets requires tuning.
- \+ Fault tolerant to small changes in input and system parameters.
- – Slight changes in MF's can cause changes in performance

Combining both approaches - Hybrid Systems

- Manual adaptation of FS parameters is not easy.
- Automatic adaptation could be done by taking advantage of NN learning capabilities.
- Can be used to learn new fuzzy rules and MF's, or to optimize existing ones.

Hybrid Systems: Approaches to Learning



Learning Fuzzy Rules (1/3)

First approach

- Start without rules and create new ones through learning.
- Create a new rule if a training pattern is not covered by existing rules.
- Drawbacks:
  - Can lead to large rule base (especially if MF's don't verlap.)
  - Can lead to inconsistent rule base $\rightarrow$ delete some rules.

Learning Fuzzy Rules (2/3)

Second approach

- Start with all possible rules.
- Delete unnecessary rules.
- Drawbacks:
  - Performance evaluation is needed for individual rules.
  - Complex with large number of variables.
  - It's possible to end with too few rules.

Learning Fuzzy Rules (3/3)

Third approach

- Start with a fixed number of rules (possibly random.)
- Replace rules during learning, checking for consistency at each step.
- Drawbacks:
  - Fixed number of rules.
  - Needs evaluation scheme for rule deletion.
  - Needs analysis procedure for acquiring new rules.

Learning Membership Functions (1/2)

- Less complex than adaptation of rule bases.
- MF's parameters can be optimized with respect to a global error.
- NN learning is used to meet certain conditions (e.g. MF's must overlap).

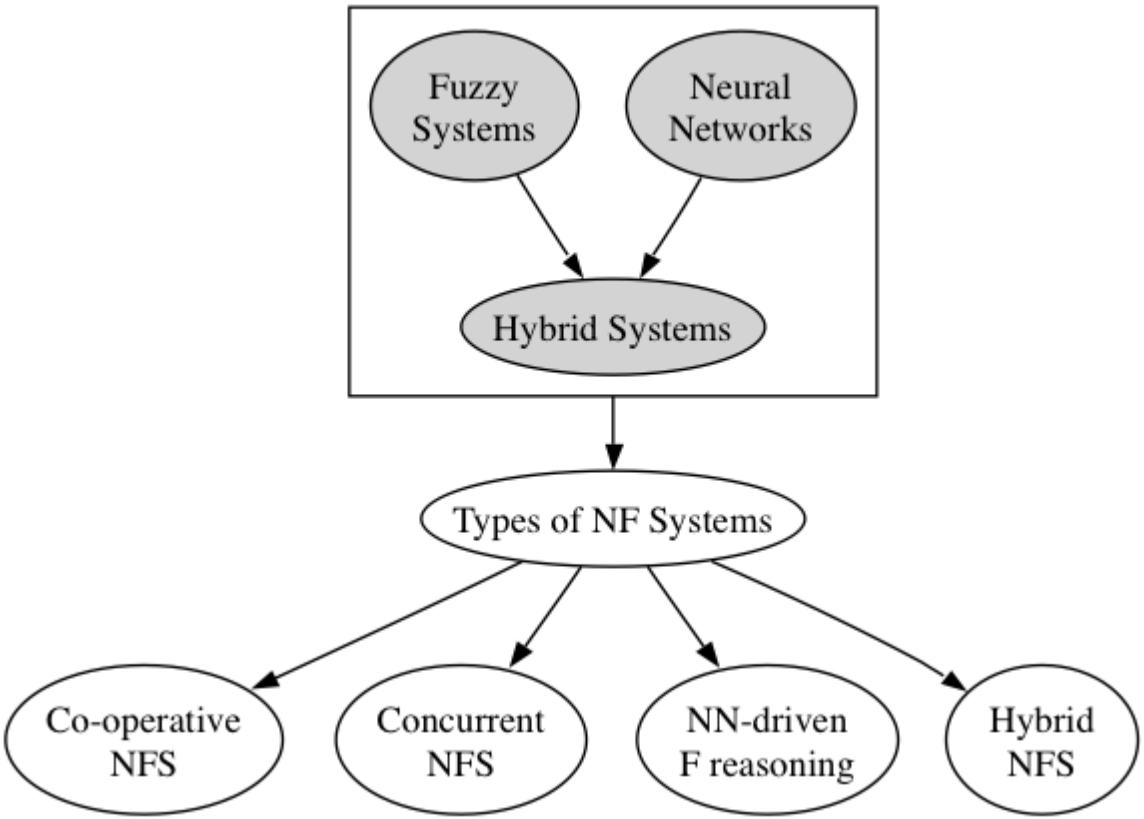
Learning Membership Functions (2/2)

Two approaches:

(i) MF's are described by parameters which are optimized in a learning process.

(ii) NN learns to produce membership values for given inputs by using sample data. (disadvantage: MF's are not explicitly known – black box)

Hybrid Systems: Classification



Neuro-Fuzzy Systems

- Basic idea is to find fuzzy system parameters by means of learning using a neural network.
- Common way:
  - Represent the FS in a NN architecture.
  - Apply a learning algorithm (e.g. back-propagation)
  - Problem: NN learning usually depends on gradient descent method, and FS inference process is not differentiable (e.g. min, max)
  - Possible solution:
    a) Replace the FS functions with differentiable ones.
    b) Use better suited procedure other than gradient descent for learning.

Common Models

- ANFIS:
  - Implements a Sugeno-like FS in a NN architecture.
  - Uses back-propagation and LMS for learning.
  - Uses only differentiable functions [solution (a)]
- GARIC:
  - Uses special “soft-min” function which is differentiable [solution (a)]
- NEFCON, NEFCLASS, and NEFPROX:
  - Use Mamdani-type FS.
  - Use special learning algorithm [solution (b)]

Common Properties of Neuro-Fuzzy Systems

- Learning
  - Trained by learning algorithm derived from NN theory.
  - Operates on local info and causes local modifications.
  - Data-driven, not knowledge based.
- Interpretation
  - Can be interpreted as a system of fuzzy rules.
  - Can be created from scratch (using learning data) or initialized with prior knowledge (fuzzy rules).
- Semantics
  - Learning takes the semantics of a FS into account (which results in constraints on modifying system parameters.)

Common Properties of Neuro-Fuzzy Systems (continued)

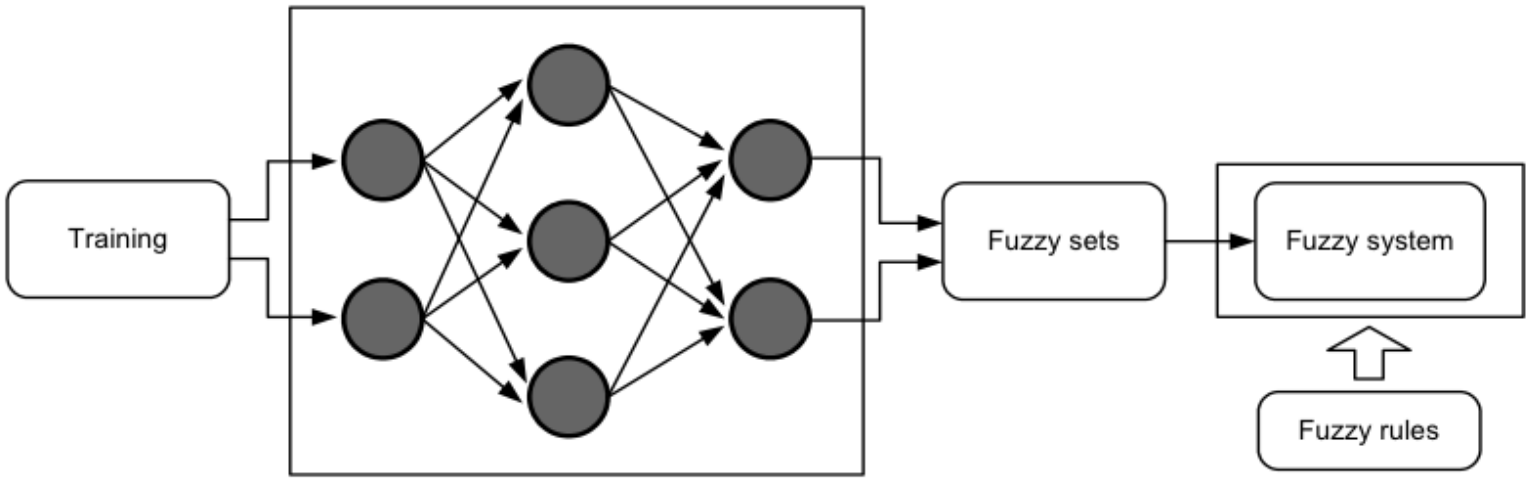
- Function approximation
  - NF system approximates an n-dimensional (unknown) function that is partially given by the training data.
  - The fuzzy rules represent vague samples (or vague prototype of training data.)
- NN-like
  - NF system can be viewed as special 3-layer feedforward network whose neurons are t-norms and t-conorms instead of activation functions.
    - 1st layer: Inputs
    - 2nd layer: Rules
    - 3rd layer: Outputs
    - Connection weights: Fuzzy sets

Types of Neuro-Fuzzy Systems

- Co-operative NF systems
  - NN and FS work independently of each other.
  - NN is used to determine certain parameters of the FS.
- Hybrid NF systems
  - FS is implemented (or interpreted) as a special kind of NN.
  - NN learning algorithm is used to tune the system.

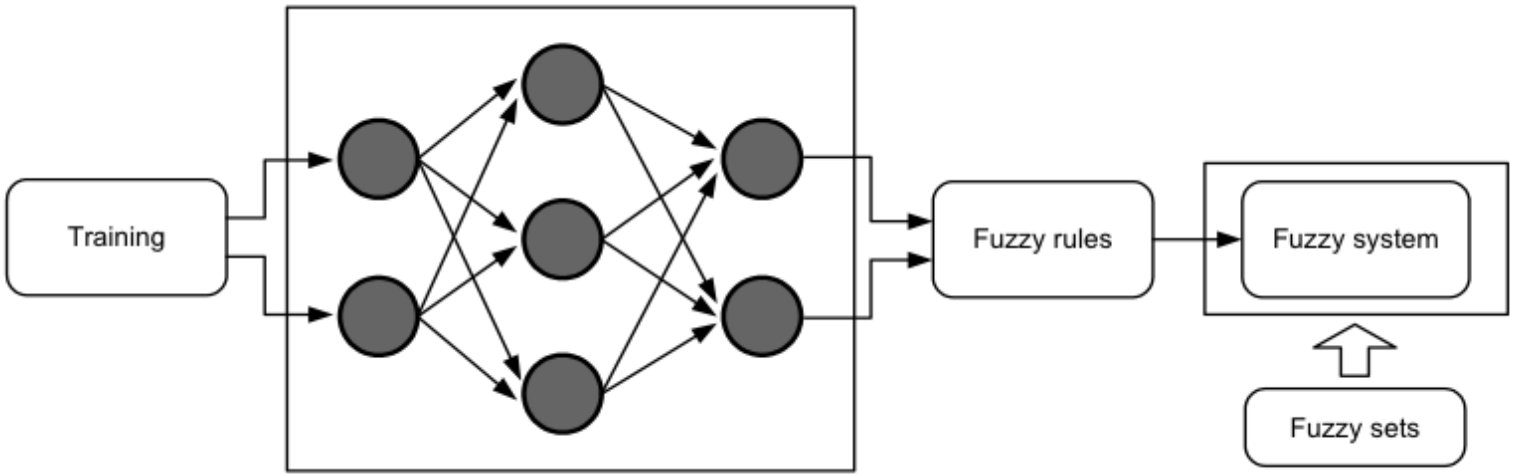
Co-operative NF systems (1/4)

- A NN determines MF's from learning data, either by tuning MF parameters or approximating the MF.
- The tuned fuzzy sets are then used with separately given fuzzy rules.



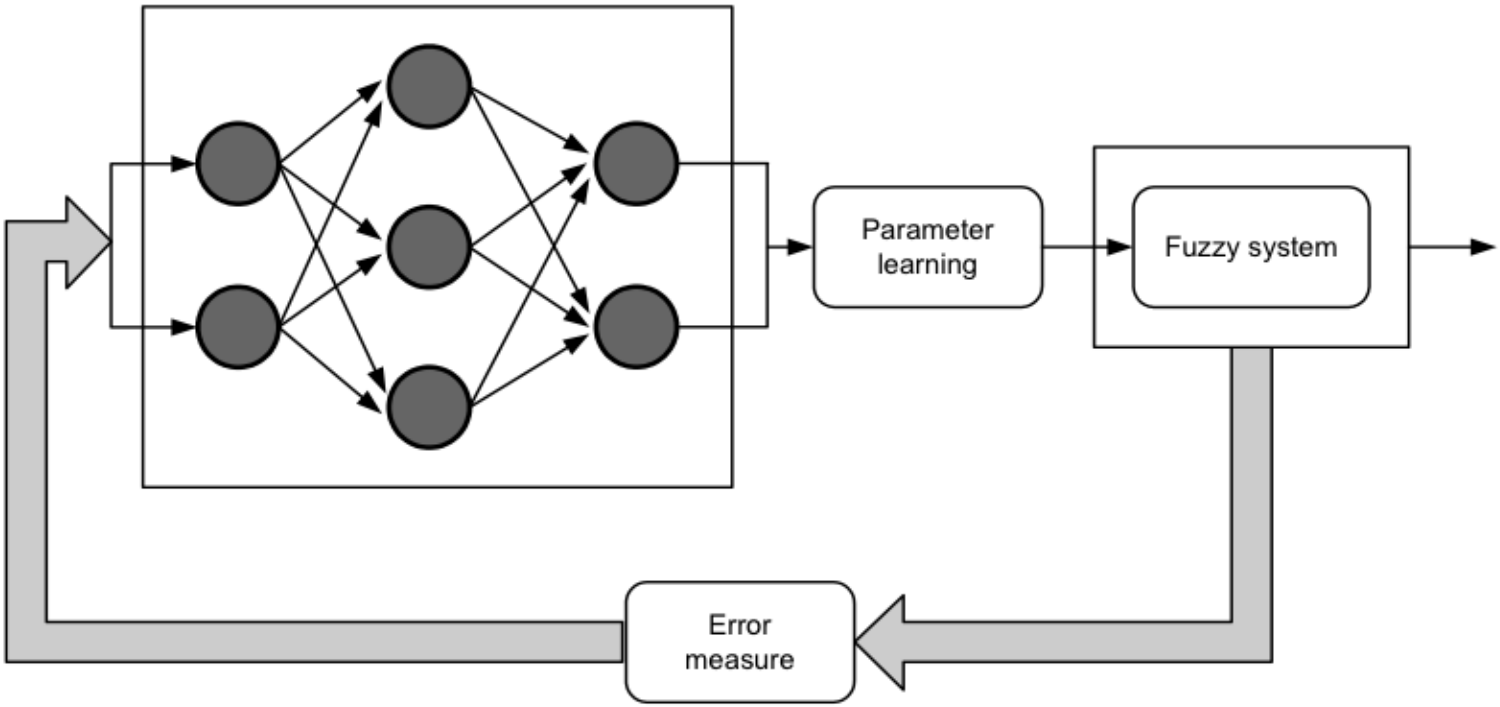
Co-operative NF systems (2/4)

- A NN determines fuzzy rules from training data (usually by clustering).
- MF's are specified separately and then used with the tuned rules.



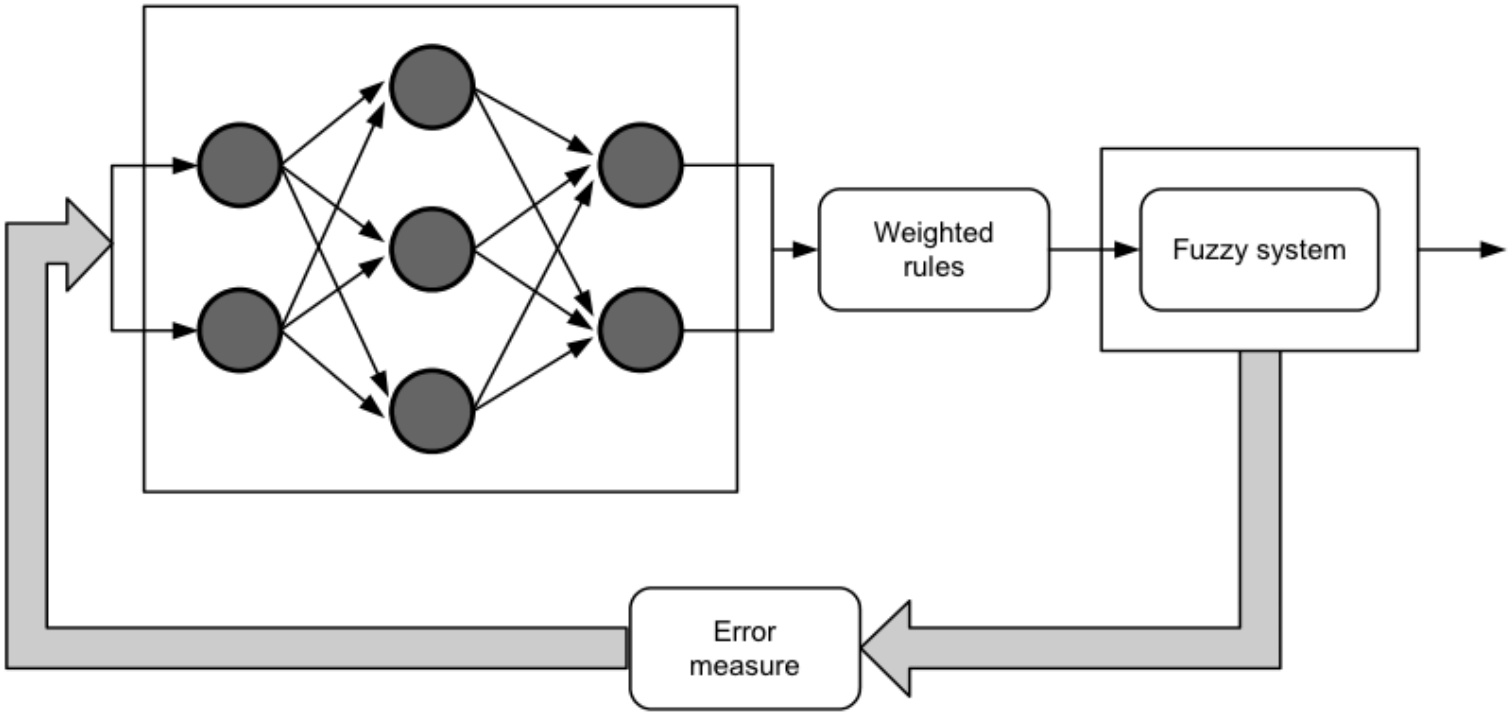
Co-operative NF systems (3/4)

- The system learns parameters online during use of the FS to adapt MF's.
- Fuzzy rules and initial MF's must be given.
- An error measure must be specified that guides learning.



Co-operative NF systems (4/4)

- A NN determines rule weights either online or offline.
- Rule weights are multiplied by rule outputs.
- Often destroys the semantics of rules.

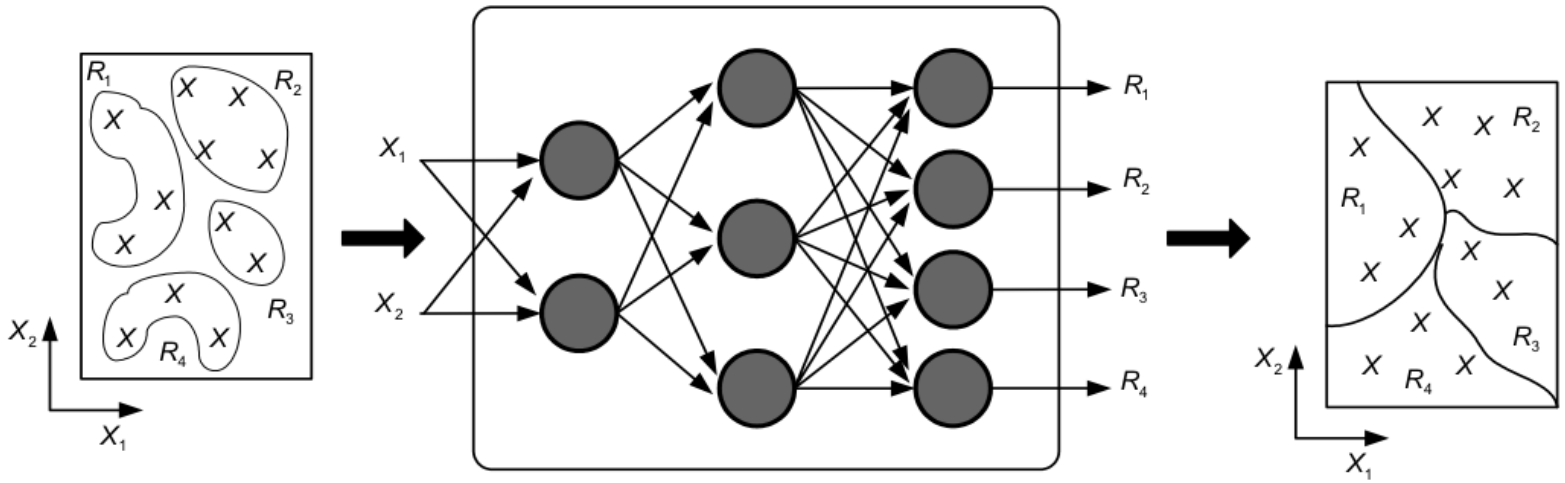


Concurrent neural/fuzzy systems

- Use a NN as a preprocessor or postprocessor of a Fuzzy system.
- These types do not optimize the FS, but improve the overall performance of the combined system.
- Useful if the inputs to a FS have to be combined, or the output is to be combined with other parameters first; i.e. inputs or outputs can not be processed directly.
- Sometimes not considered as neuro-fuzzy systems

Neural Network-driven Fuzzy Reasoning

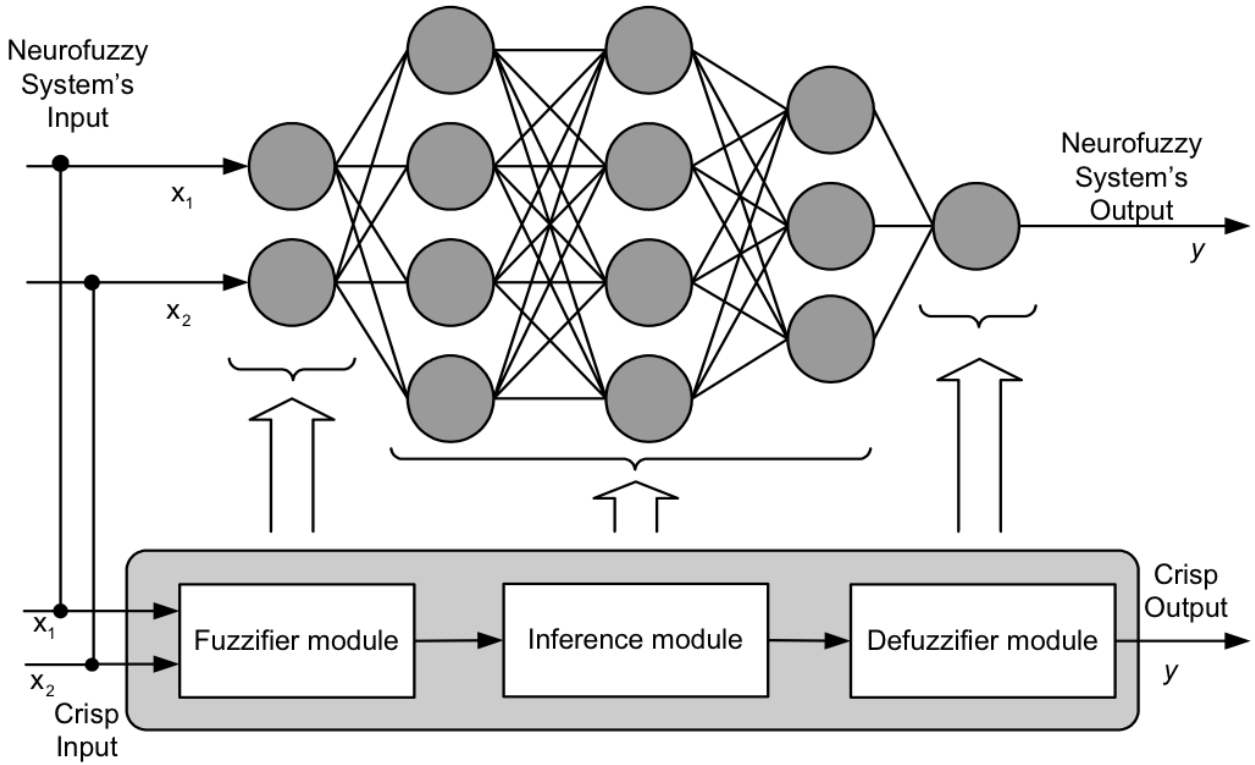
- Large number (¿4) of linguistic variables - construction of FRB can be difficult



- Can be used with automatically partitioned input/output
- Reasoning can be adapted to changes in environment

Hybrid Neuro-Fuzzy Systems

- Architecture integrating NN and FL-based system in appropriate (parallel) structure
- One entity, as opposed to cooperative or concurrent NFS



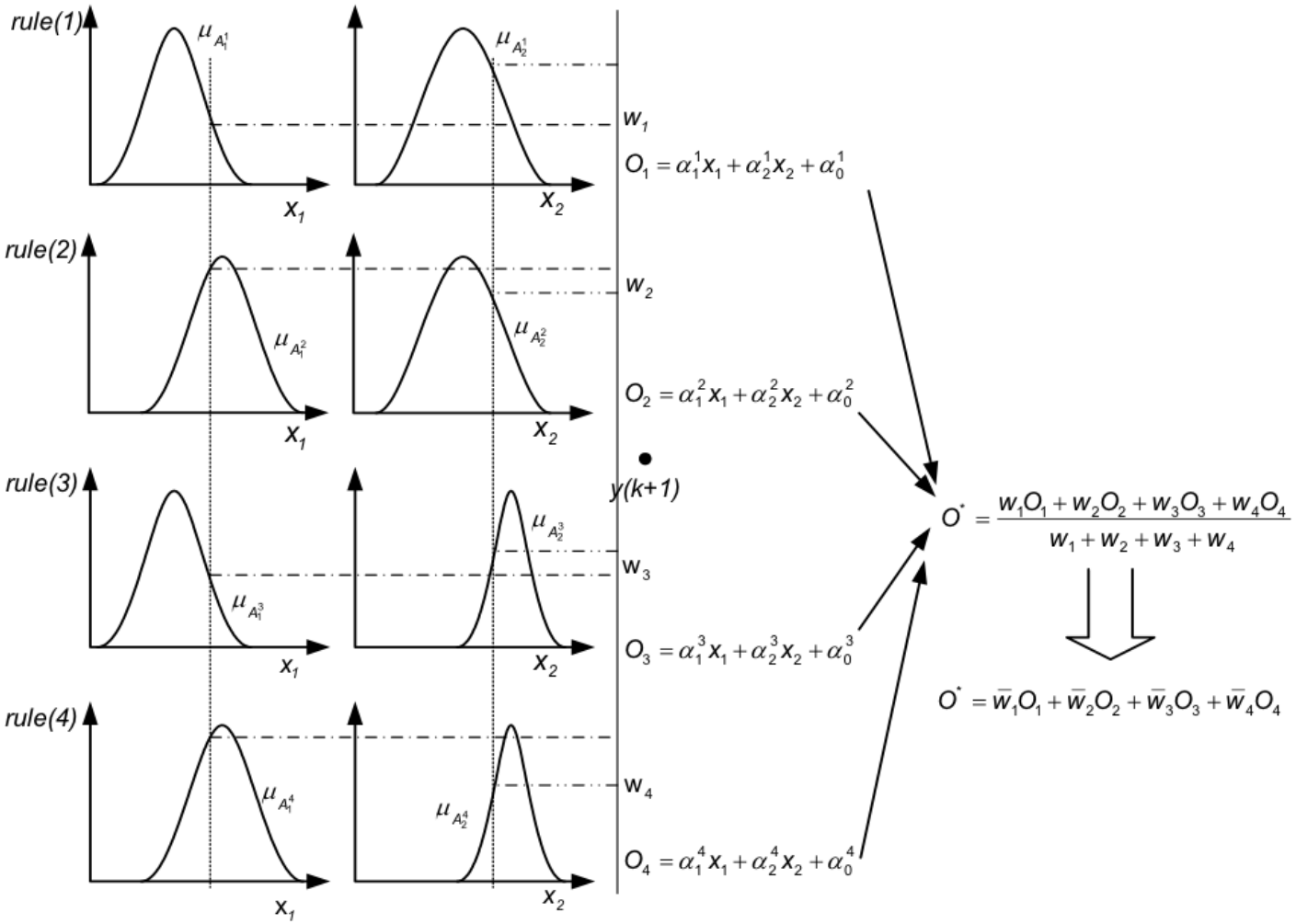
ANFIS

- Adaptive Network based Fuzzy Inference System
- Sugeno-tupe fuzzy system with rules in form

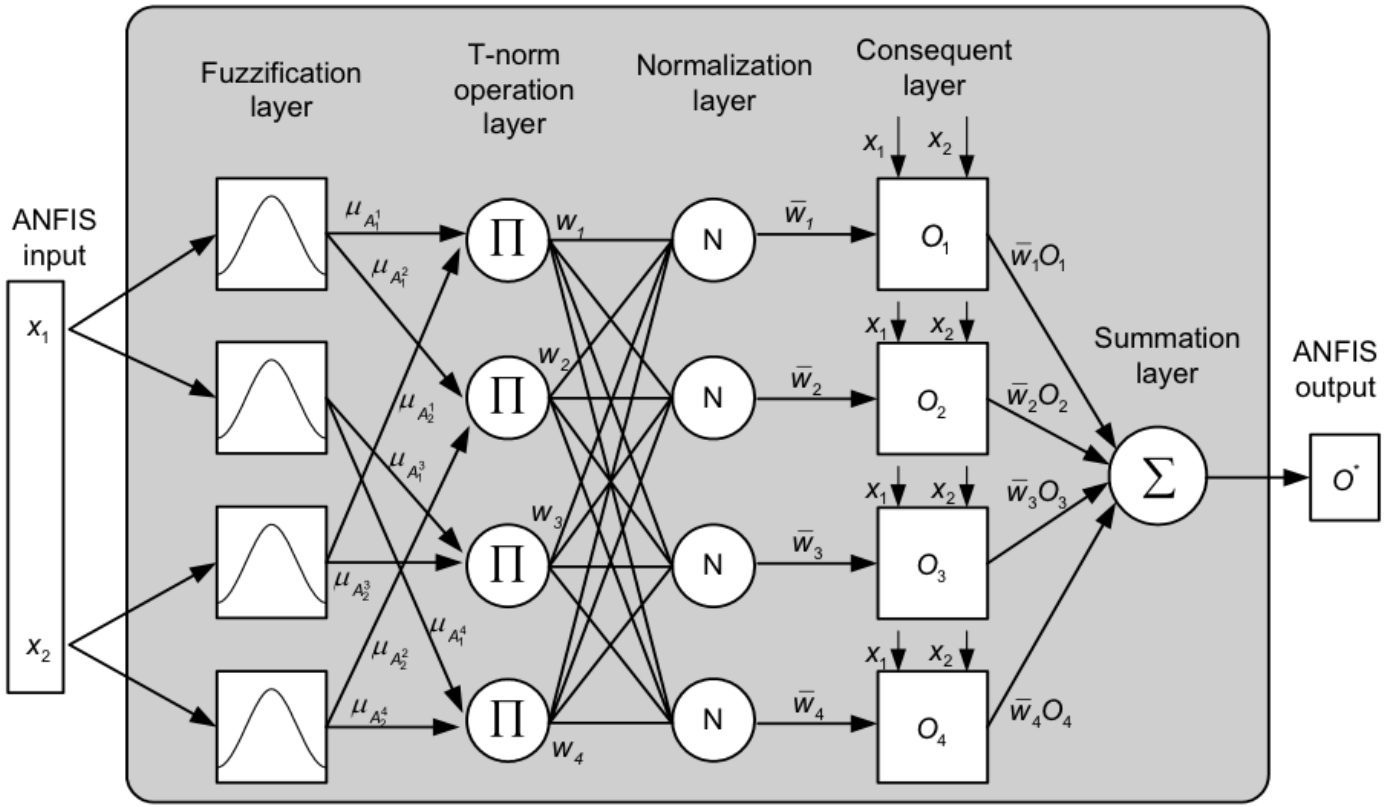
R_p: IF x_1 is A_1^p ANDx_2 is A_2^p AND $\ldots$ AND x_n is A_n^p
THEN $O_p = \alpha_0^p + \alpha_1^p x_1 + \ldots + \alpha_n^p x_n$

- 5-layer feed-forward network
  - Fuzzification
  - **t**-norm operation
  - Normalization
  - Consequent
  - Aggregation

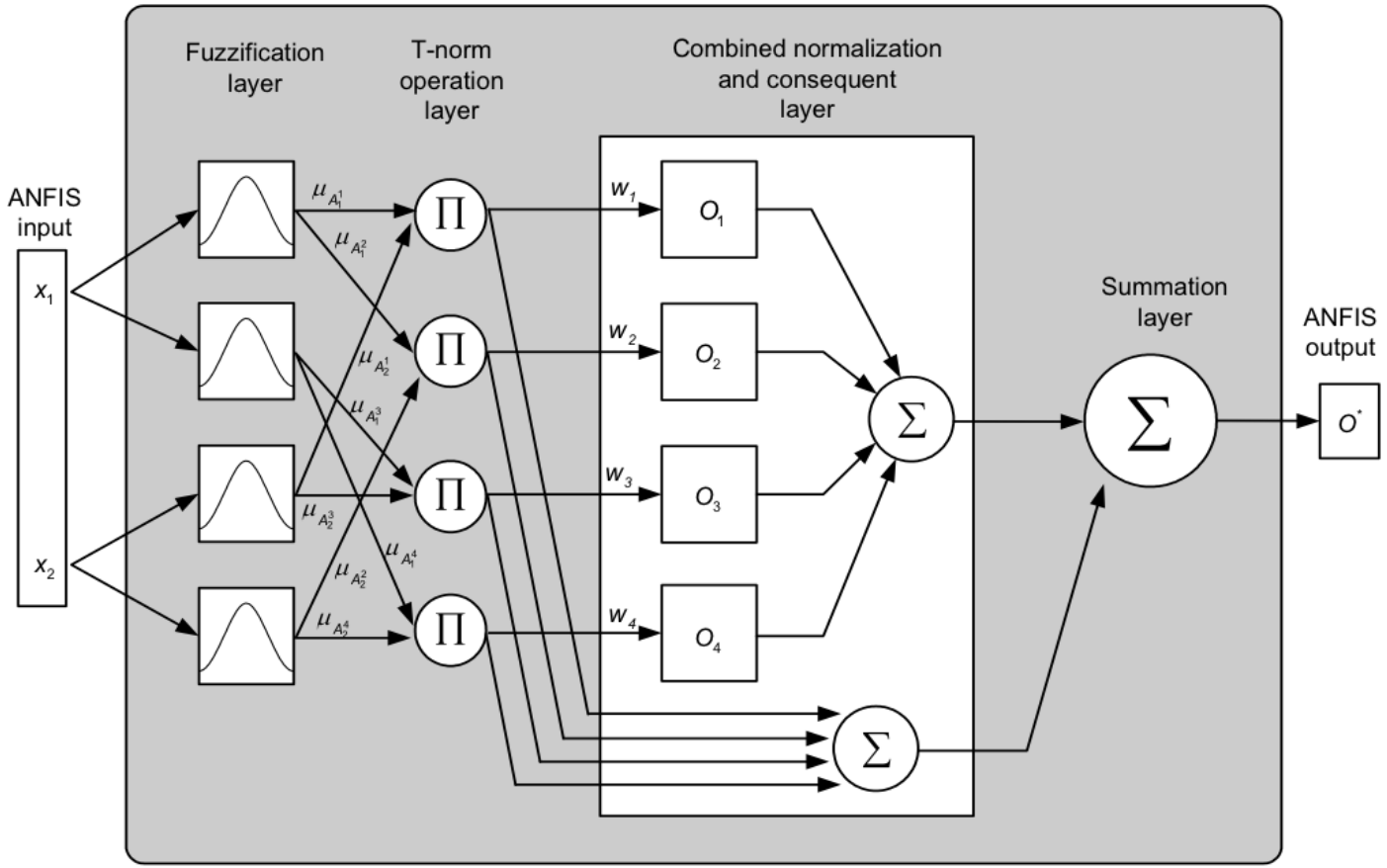
ANFIS example: 2 inputs, 1 output, 4 1st-order rules



ANFIS example: architecture



ANFIS variant: 4-layer network

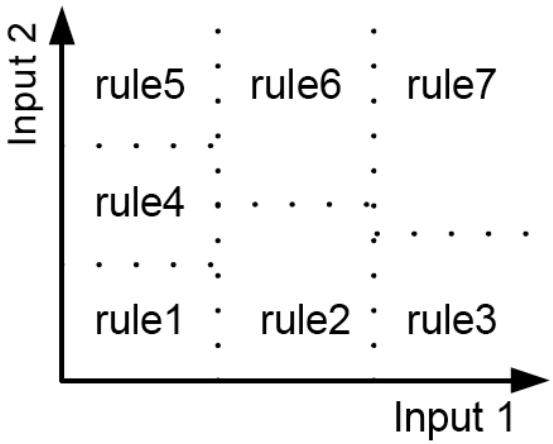


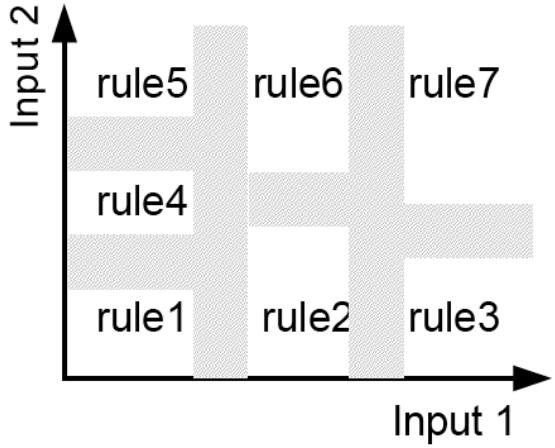
Construction of Neuro-Fuzzy Systems

- Structure identification pahse: I/O space partitioning
  - Each partition/patch represents a rule
- Parameter learning phase
  - Backpropagation learning
  - Hybrid learning

NFS Construction - Structure

- Each partition/patch represents a rule (crisp vs. fuzzy)

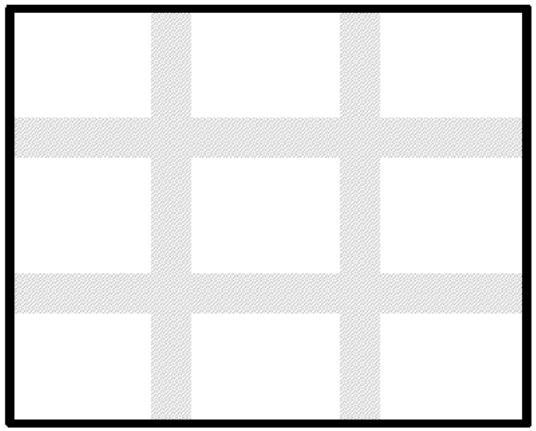


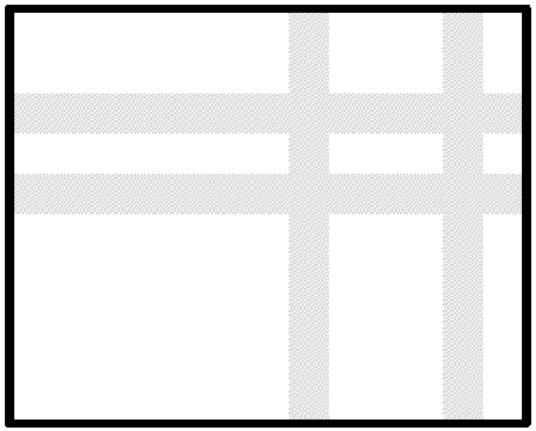


- Partitioning approaches
  - Grid partitioning
  - Clustering
  - Scatter partitioning

Grid Partitioning

- Fixed vs. adaptive grid partitioning





+ Rules can be generated using a (relatively) small number of linguistic variables.
– Size of rule-base grows exponentially (so called *curse of dimensionality*).

Clustering

- Hard vs.soft partitioning (cf. crisp vs. fuzzy partition)

Most important approach: fuzzy c-means which represents $X = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^d$ using c-clusters with centers at locations

$$v_i = \frac{\sum_{j=1}^{n} (\mu_{ij} x_j)}{\sum_{j=1}^{n} \mu_{ij}}$$

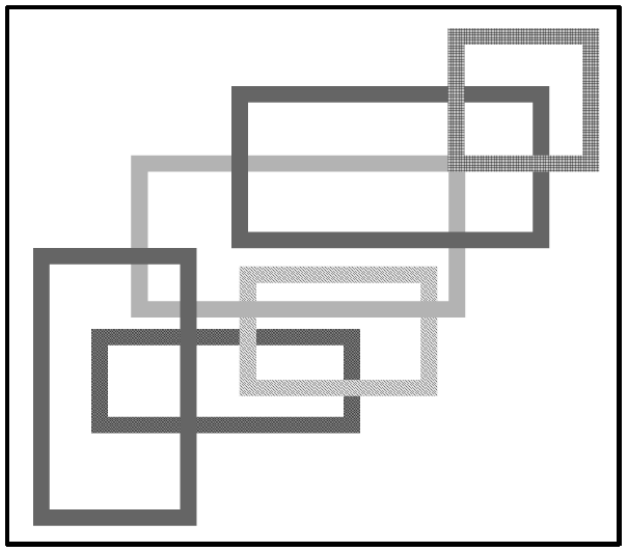
These locations are optimized using the objective function

$$J(U, V) = \sum_{j=1}^{n} \sum_{i=1}^{c} \mu_{ij}^m \|x_j - v_i\|^2$$

where U is partition matrix composed of memberships of each data point x_j in each cluster v_i.

Scatter Partitioning

- Antecedent (IF–part) of rules can be positioned at arbitrary locations of input space



\+ Limits the number of generated fuzzy rules

– Finding suitable number of rules and their positions and widths is difficult

Parameter learning phase

Learning algorithms - strategies for optimizing weights/biases of NNs, or MFs and rules of Fuzzy rule-based systems

- Backpropagation learning: requires differentiability, suffers from known problems (local minima, plateaus)
- Hybrid learning: combination of two or more learning algorithms (RBF, SOM, LMS, BP, ...)