INF FAKULTÄT FÜR INFORMATIK

Learning Fuzzy Systems

Prof. Dr. Rudolf Kruse

Building Fuzzy Systems

Fuzzy systems for real world applications cannot be generated automatically, but the learning tasks can be supported by several different learning methods

Hierarchie of Learning Tasks for Fuzzy Systems (e.g. for defining an input-output function)
Learn a single fuzzy set (e.g. parameters of a family of functions or membership degrees)
Learn a collection of fuzzy sets (e.g. a partition)
Learn a fuzzy rule (e.g. a if-then rule with Gödel Interpretation for a linguistic rule)
Learn a systems of fuzzy rules (e.g. an interpretable set of logical rule that is internally represented by a fuzzy relation)

Data driven methods for learning Fuzzy Systems (machine learning methods)
Supervised learning
Unsupervised learning
Semi supervised learning
Reinforcement learning
Active learning
and many others

Fuzzy systems for real world applications cannot be generated automatically, but this process can be supported by several software tools with learning methods:

- Tuning fuzzy sets and fuzzy partitions (e.g. parameter fitting, supervised)
- Finding useful fuzzy sets and fuzzy rules (e.g. clustering, unsupervised)
- Creating a suitable fuzzy systems architecture (structur identification, knowledge discovery, reinforcement learning, model selection, uncertainty quantification,…)
- Evolving Fuzzy Systems

Hybrid learning is currently most successful: The prior knowledge of the user is combined with intelligent data analysis methods from machine learning, e.g. deep learning.

Learning Fuzzy Rules by Fuzzy Clustering

Example: Transfer Passenger Analysis

German Aerospace Center (DLR) developed macroscopic passenger flow model for simulating passenger movements on airport's land side

For passenger movements in terminal areas: distribution functions are used today

Goal: build fuzzy rule base describing transfer passenger amount between aircrafts

These rules can be used to improve macroscopic simulation

Idea: find rules based on fuzzy *c*-means (FCM)

Attributes for Passenger Analysis

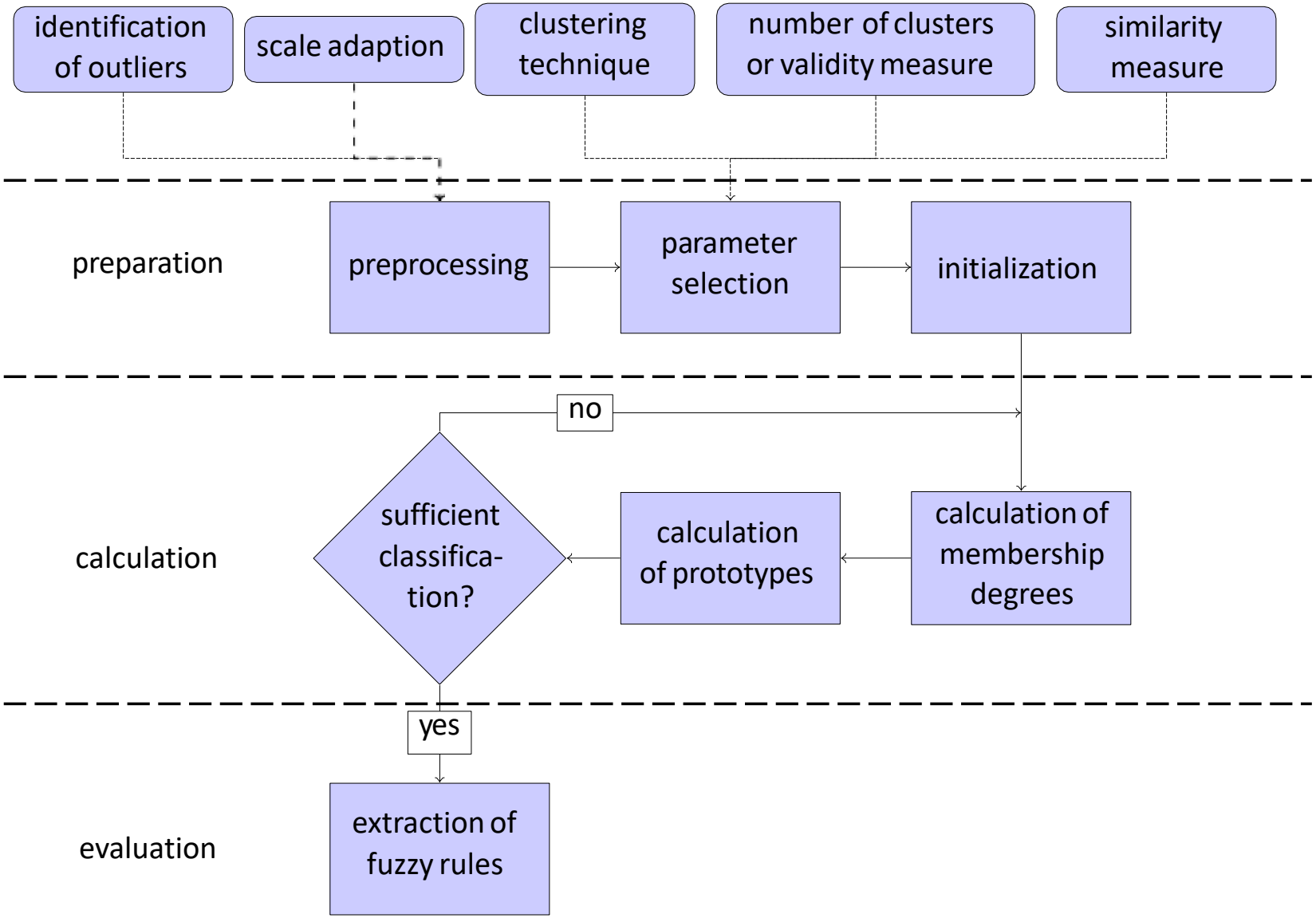
Maximal amount of passengers in certain aircraft (depending on type of aircraft)

Distance between airport of departure and airport of destination (in three categories: short-, medium-, and long-haul)

Time of departure

Percentage of transfer passengers in aircraft

General Clustering Procedure



Constraints for the Objective function

Fuzzy clustering

Noise clustering

Influence of outliers

Influence of Outliers

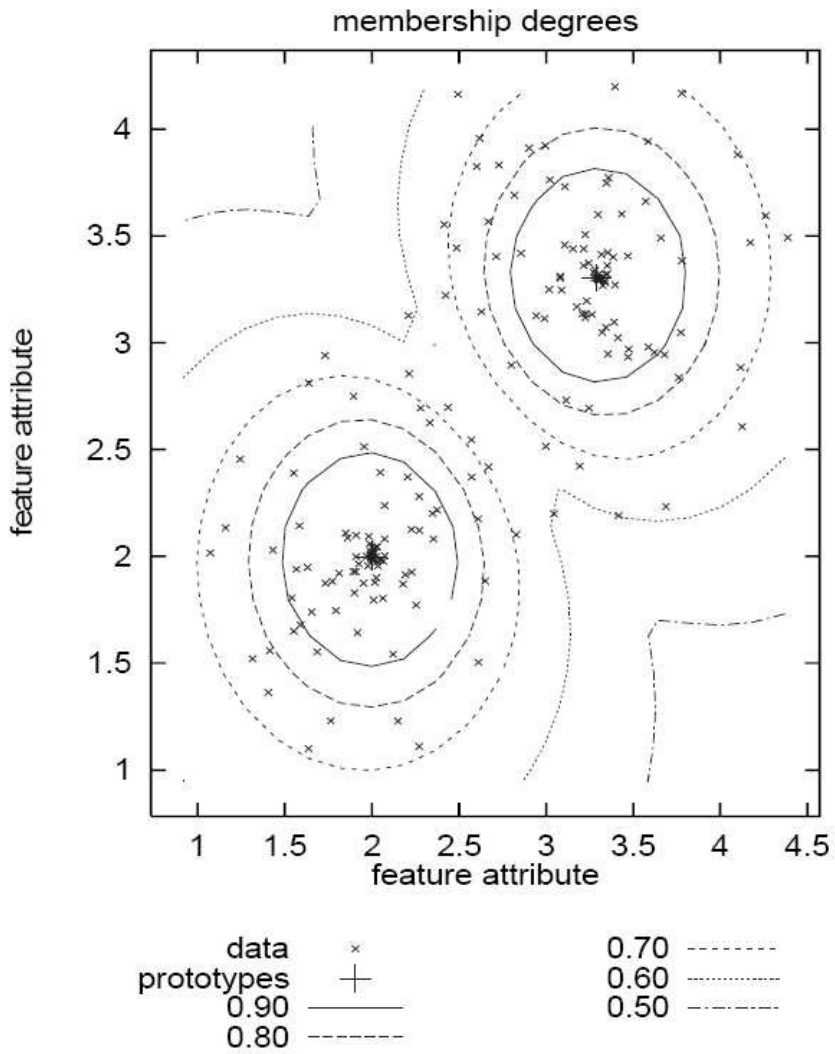
A weighting factor ω_j is attached to each datum $\boldsymbol{x}_j$

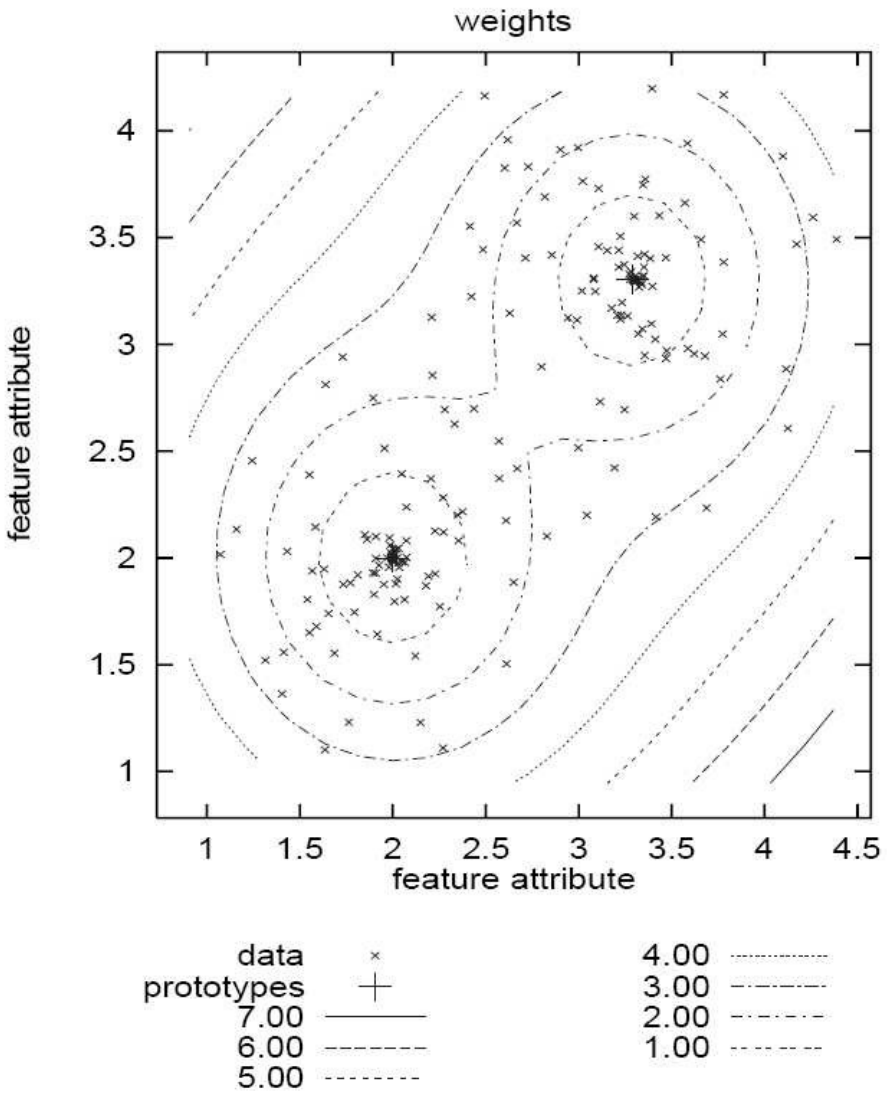
Weighting factors are adapted during clustering

Using concept of weighting factors:

- outliers in data set can be identified and
- Outlier's influence on partition is reduced

Membership Degrees and Weighting Factors





Minimize objective function

$$J(X, U, C) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{m} \cdot \frac{1}{\omega_j^q} \cdot d_{ij}^2$$

subject to

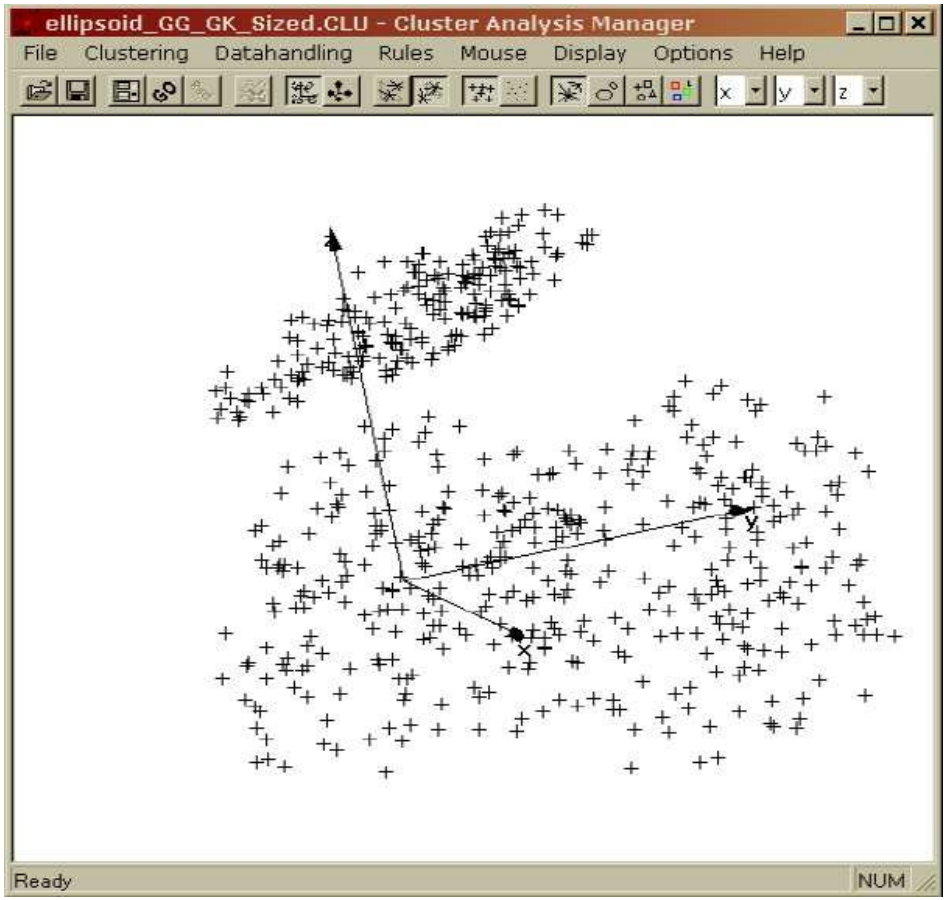
$$\forall j \in [n] : \sum_{i=1}^{c} u_{ij} = 1, \quad \forall i \in [c] : \sum_{j=1}^{n} u_{ij} > 0, \quad \sum_{j=1}^{n} \omega_j = \omega$$

q determines emphasis put on weight adaption during clustering

Update equations for memberships and weights, resp.

$$u_{ij} = \frac{d_{ij}^{\frac{2}{1-m}}}{\sum_{k=1}^{c} d_{kj}^{\frac{2}{1-m}}}, \qquad \omega_j = \frac{\left(\sum_{i=1}^{c} u_{ij}^m d_{ij}^2\right)^{\frac{1}{q+1}}}{\sum_{k=1}^{n} \left(\sum_{i=1}^{c} u_{ik}^m d_{ik}^2\right)^{\frac{1}{q+1}}} \cdot \omega$$

Determining the Number of Clusters



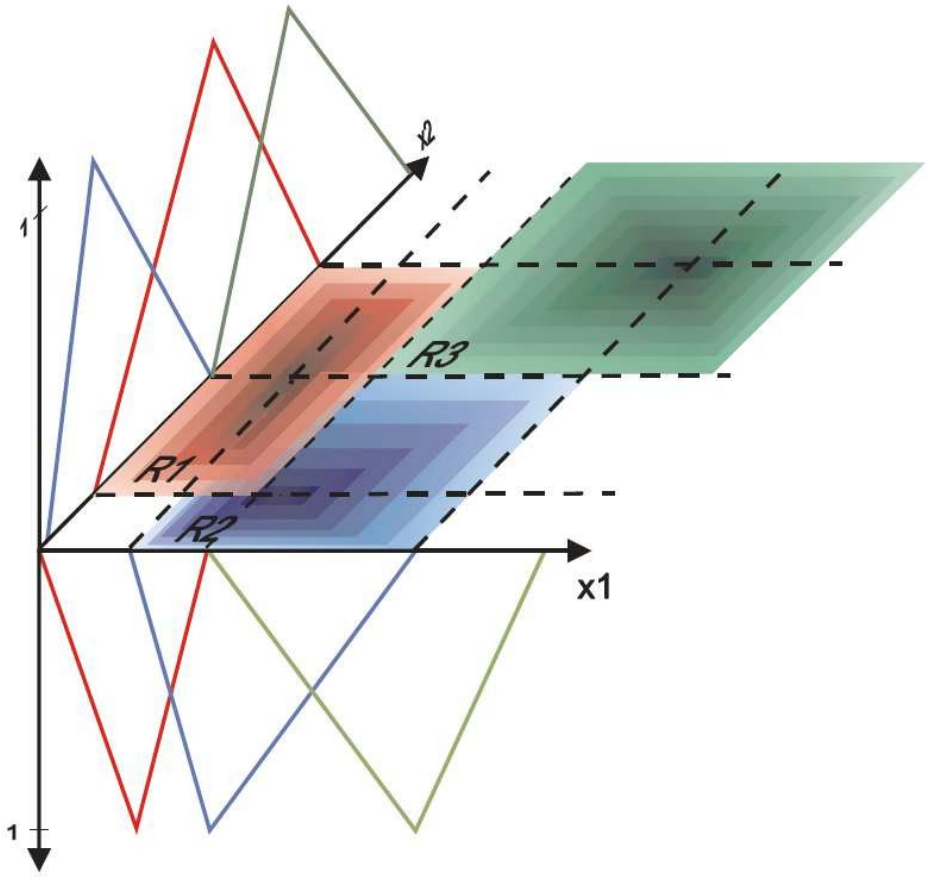
Here, validity measures evaluating whole partition of data

Getting: global validitymeasures

Clustering is run for varying number of clusters

Validity of resulting partitions is compared

Fuzzy Rules and Induced Vague Areas

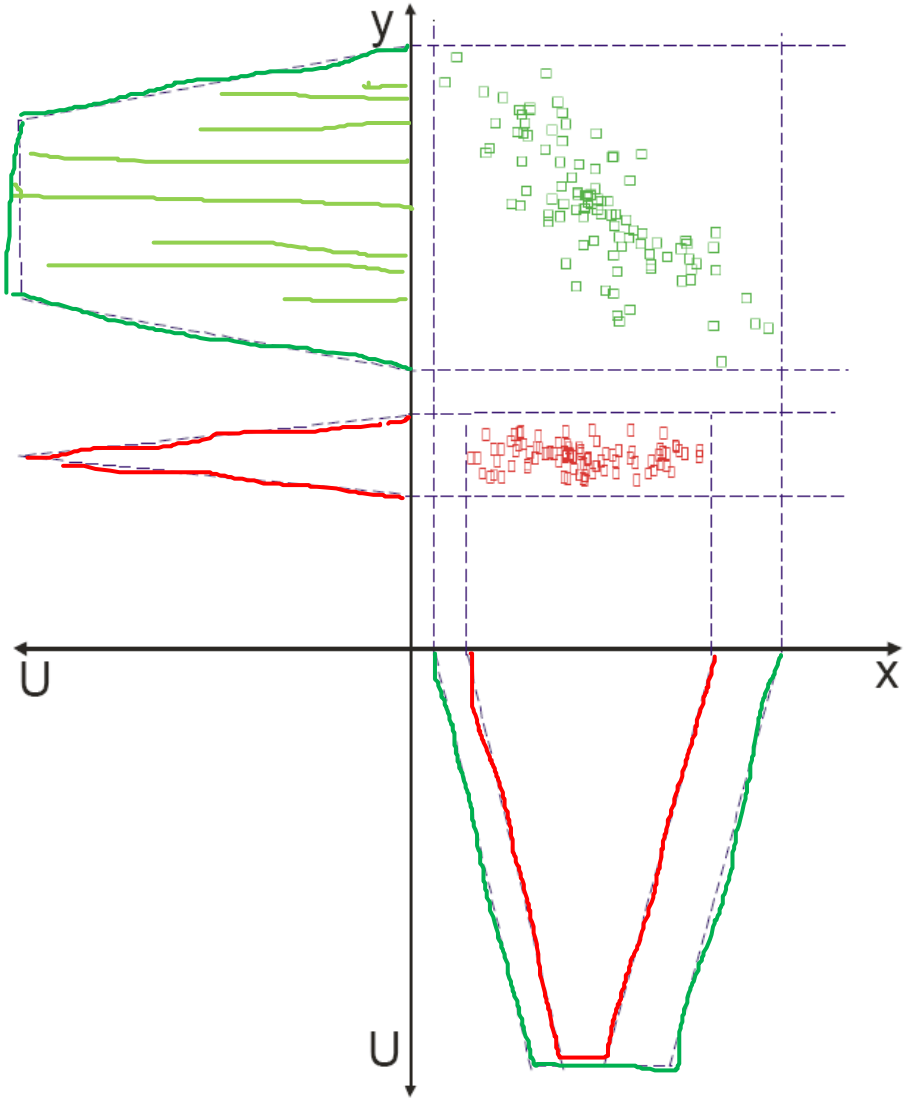


Intensity of color indicates firing strength of specific rule

Vague areas = fuzzy clusters where color intensity indicates membership degree

Tips of fuzzy partitions in single domains = projections of multidimensional cluster centers

Finding fuzzy partitions and fuzzy rules



Projection of cluster elements with membership degree

Choosing fitting simple fuzzy sets

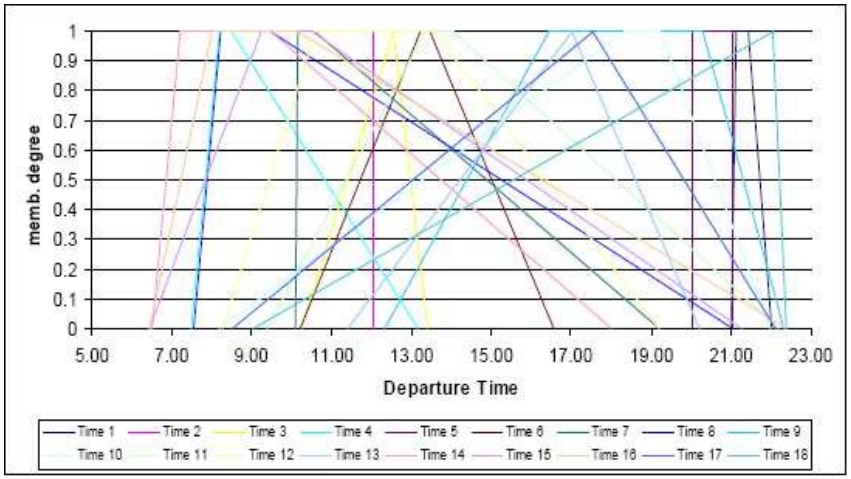
Combining similar fuzzy sets

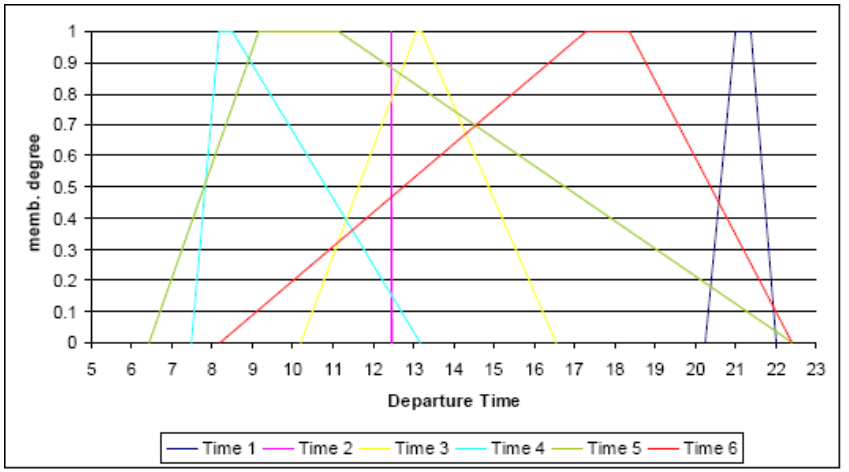
Rules with same input sets are

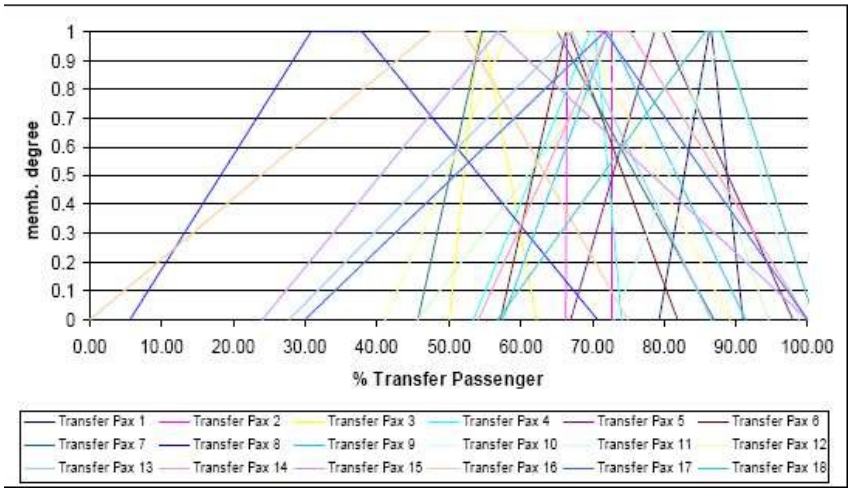
- Combined if they also have same output set(s) or
- Otherwise removed from rule set

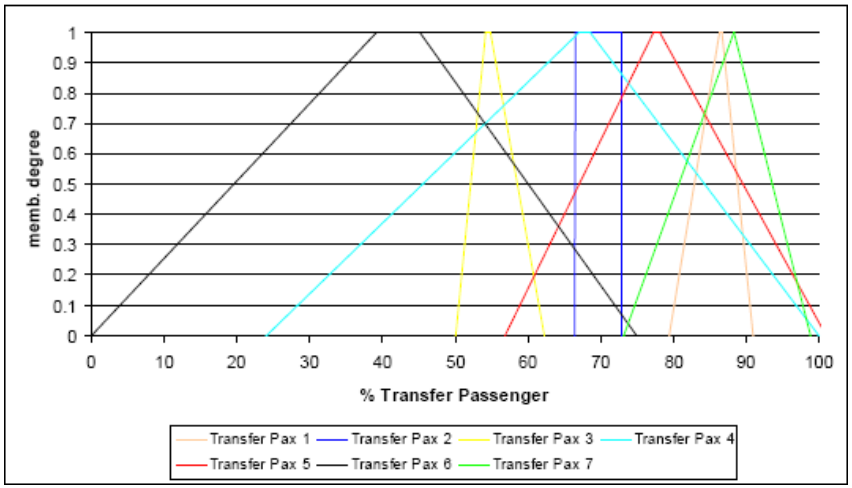
Results

FCM with c = 18, outlier and size adaptation, Euclidean distance:









resulting fuzzy sets

simplified fuzzy sets

Evaluation of the Rule Base

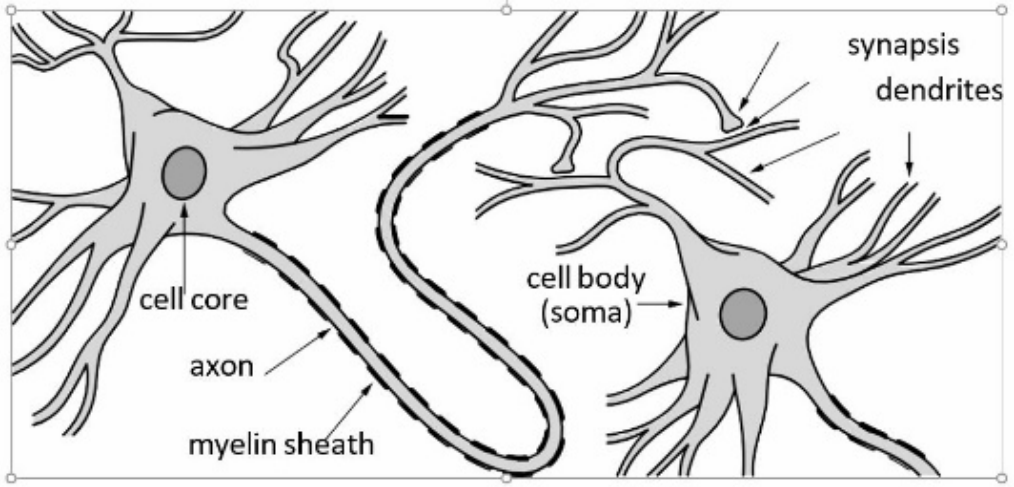
rule	max. no. of pax	De st.	depart.	% transfer pax
1	paxmax1	R1	time1	tpax1
2	paxmax2	R1	time2	tpax2
3	paxmax3	R1	time3	tpax3
4	paxmax4	R1	time4	tpax4
5	paxmax5	R5	time1	tpax5
. . .	. . .	. . .	. . .	. . .

rules 1 and 5: aircraft with relatively small amount of maximal passengers (80-200), short- to medium-haul destination, and departing late at night usually have high amount of transfer passengers(80-90%)

rule 2: flights with medium-haul destination and small aircraft (about 150 passengers), starting about noon, carry relatively high amount of transfer passengers (ca. 70%)

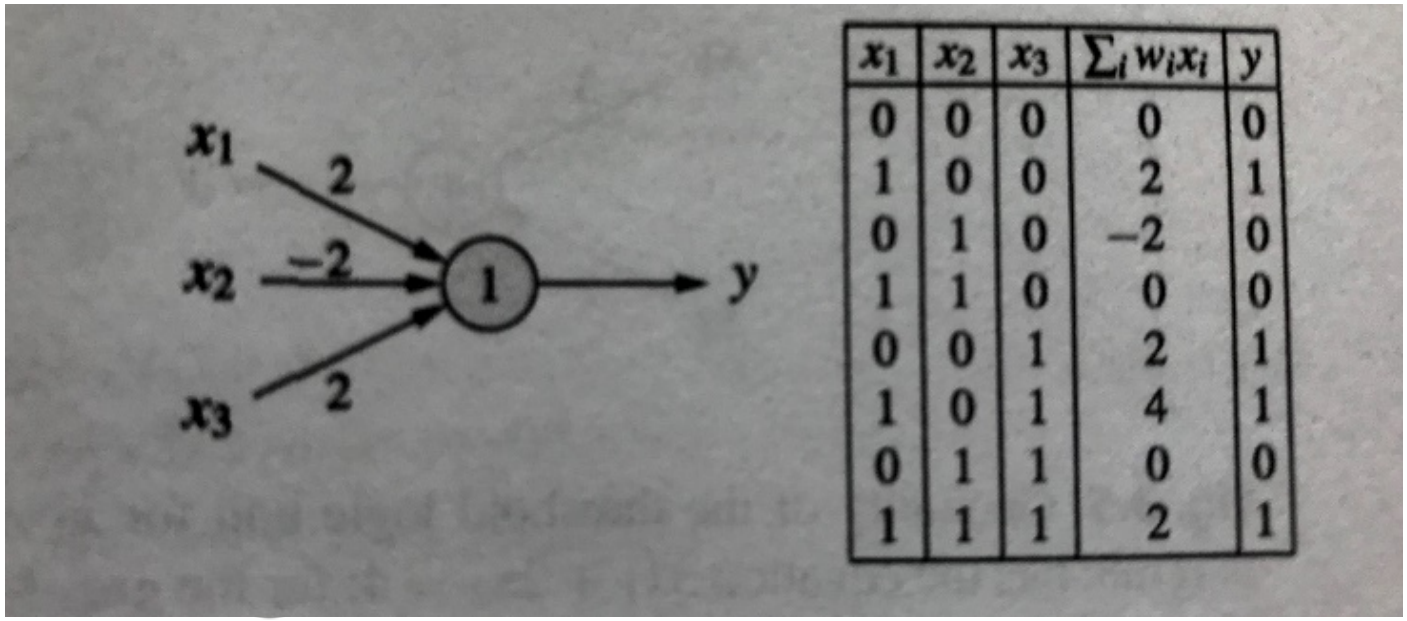
Reminder: Neural Network

Neuron



humans have
100000000000 neurons
with up to
10000 inputs each

Artificial Neuron
=
Input-output function

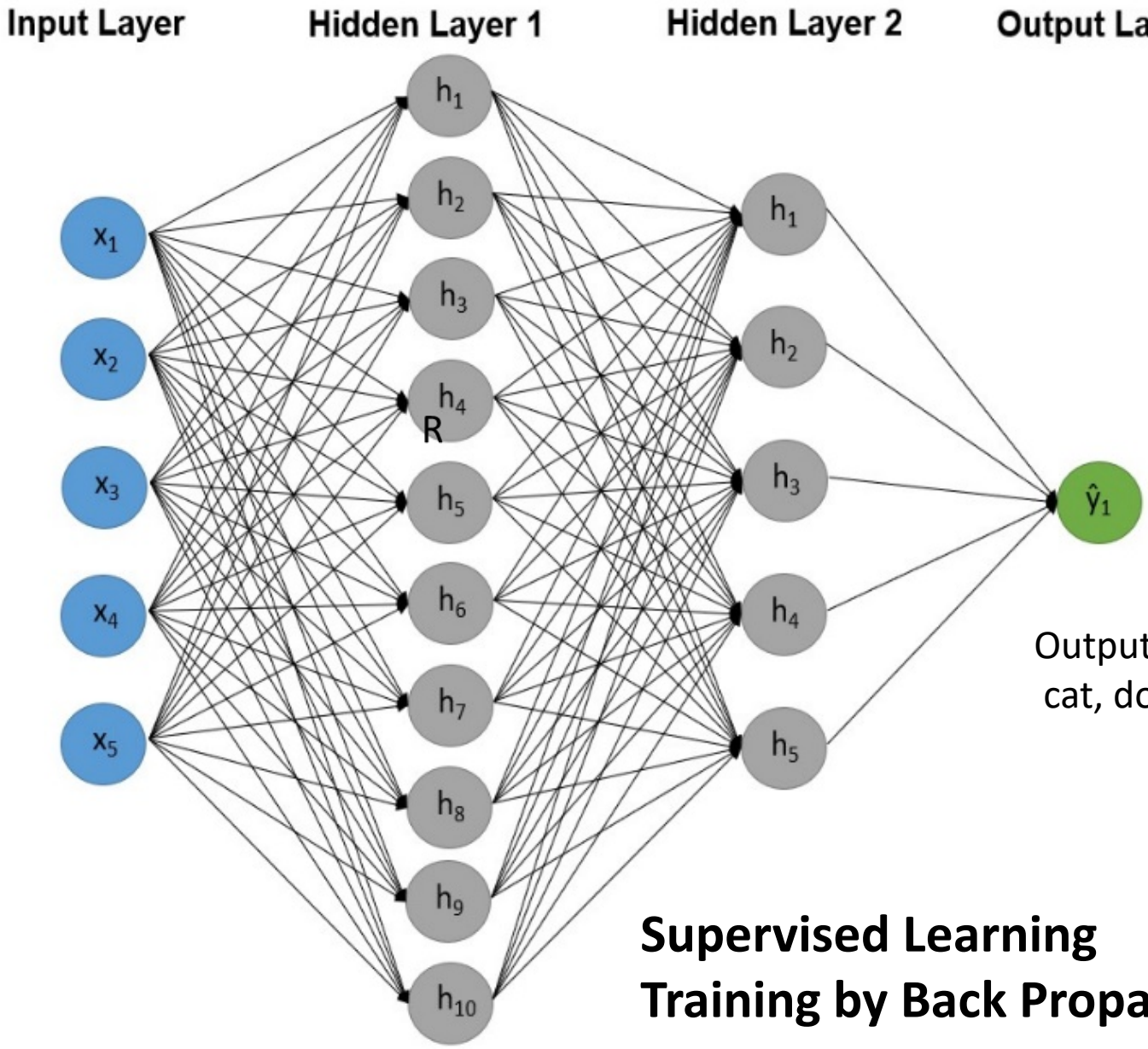


x_1	x_2	x_3	$\sum_i w_i x_i$	y
0	0	0	0	0
1	0	0	2	1
0	1	0	−2	0
1	1	0	0	0
0	0	1	2	1
1	0	1	4	1
0	1	1	0	0
1	1	1	2	1

Neural Network: Medium Size

400*266 pixel

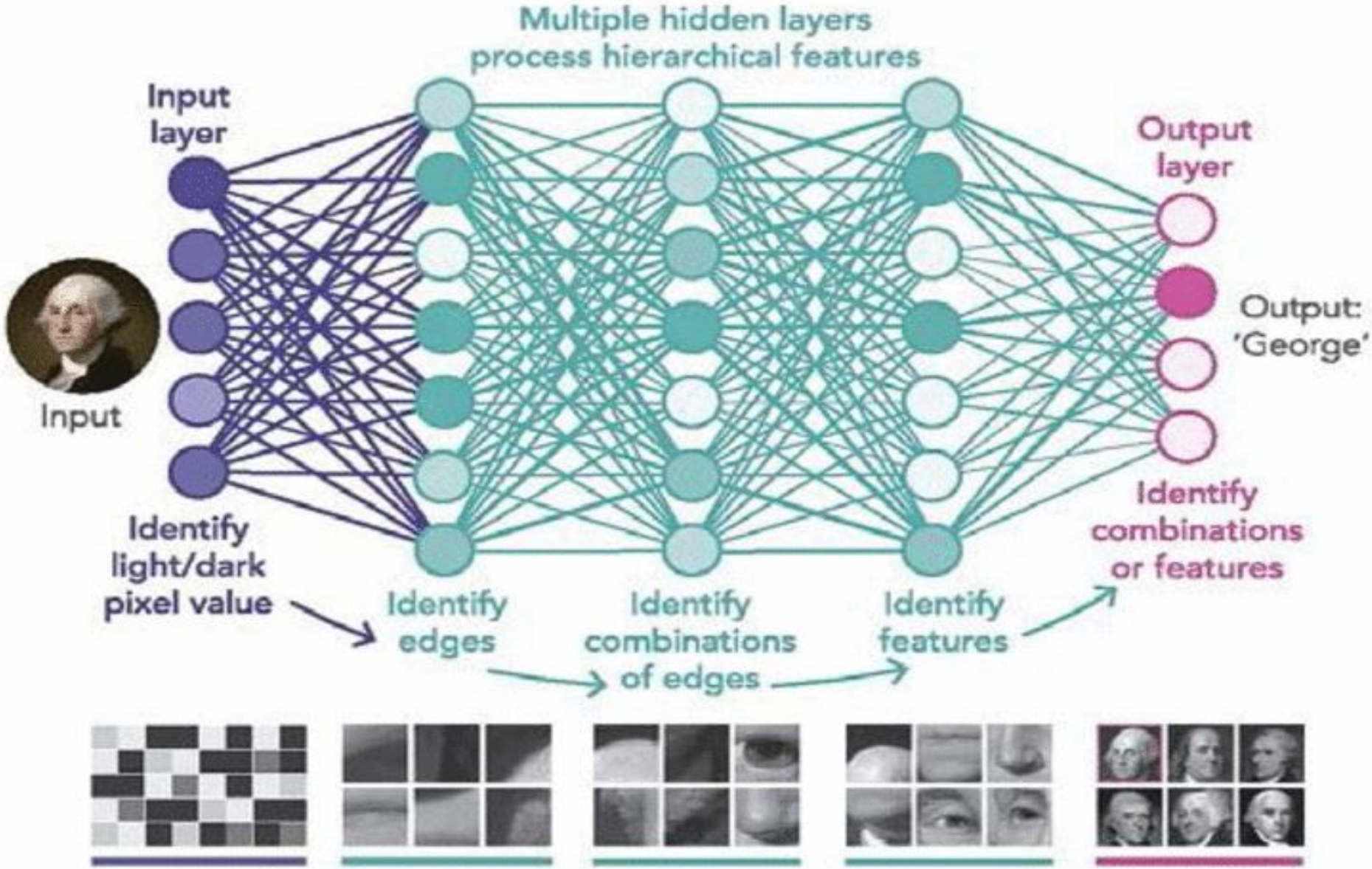
Per pixel one input neuron



Output:
cat, dog,…

Supervised Learning
Training by Back Propagation

Neural Network: Deep Learning



Object Recognition in Pictures

Imagenet Large Scale Visual Recognition Challenge since 2010

Look for 200 classes (chair, table, person, bike, ...)

Pictures with ca. 500 x 400 pxels, 3 color channels

Neural network with ca. 600.000 neurons in the first layer

200 neurons in the last layer

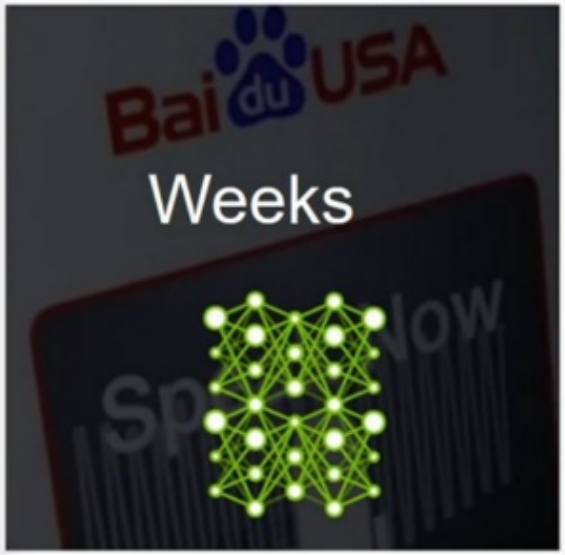
Winner: Classification error 2% (better than humans, typically 5%)

Training is the a big Challenge of AI

7 ExaFLOPS
60 Million Parameters

2015 - Microsoft ResNet
Superhuman Image Recognition

20 ExaFLOPS
300 Million Parameters



2016 - Baidu Deep Speech 2
Superhuman Voice Recognition

100 ExaFLOPS
8700 Million Parameters

2017 - Google Neural Machine Translation
Near Human Language Translation

Courtesy NVidia

Fastest Accelerator Cards Today ~ 100 Tera Flops/sec

Comparison of Neural Networks and Fuzzy Systems

Neural Networks	Fuzzy Systems
are distributed, connected low-level computational structures	deal with reasoning on a higher level
Data driven: Perform well when enough data are available	Knowledge driven: Use information from domain experts
Designed for Learning – lots of methods exist	Designed for interpretable knowledge representation
are complex and black-boxes for the user	are simple, understandable and explainable

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

Neuro-Fuzzy Models

Neuro-Fuzzy Systems with supervised learning optimize the fuzzy sets of a given rule base by observed input-output tuples.

Requirement: An existing (fuzzy) rule base must exist.

If no initial rule base is available, we might apply fuzzy clustering to the input data for that.

In the following, we discuss two typical example for a neuro-fuzzy system with supervised learning: ANFIS and NEFCLASS

The ANFIS Model

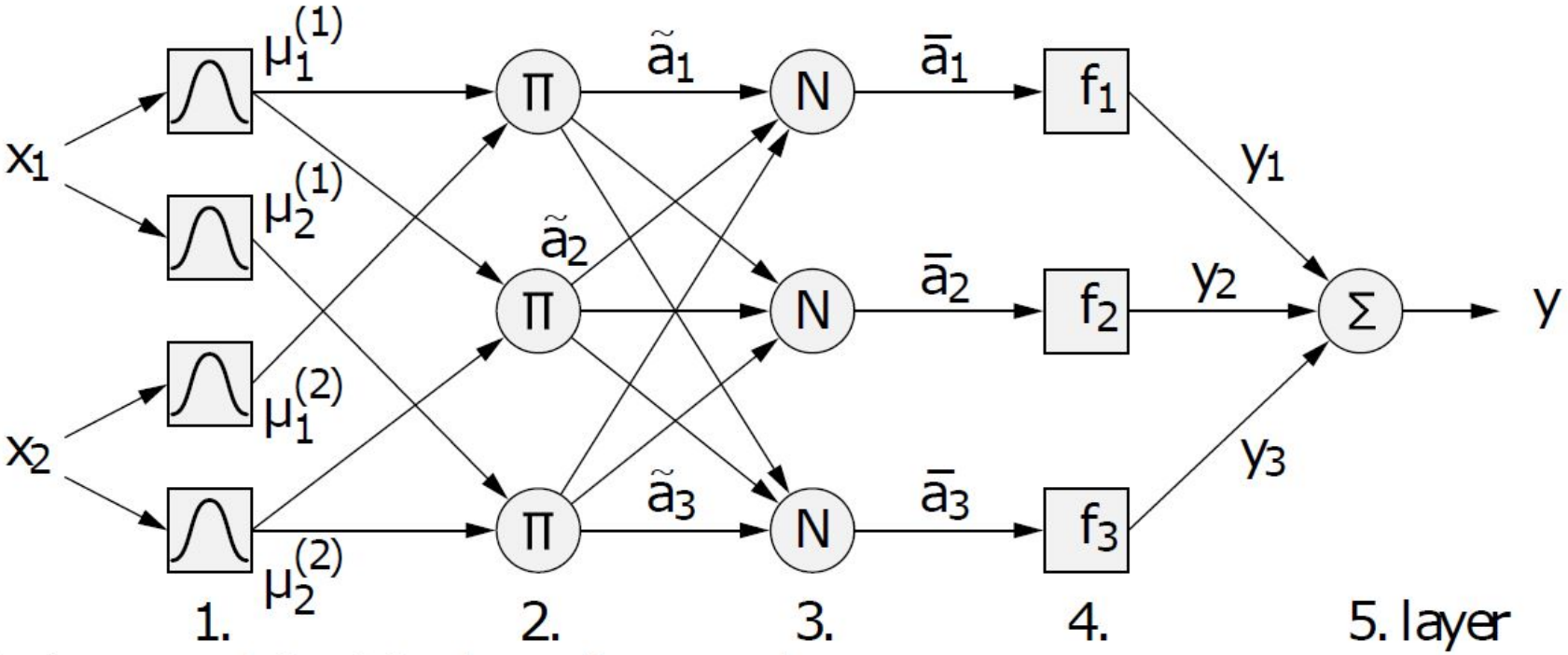
The neuro-fuzzy system *ANFIS* (Adaptive-Network-based Fuzzy Inference System)

It has been integrated in many controllers and simulation tools, *e.g.* Matlab.

The ANFIS model is based on a hybrid structure, *i.e.* it can be interpreted as neural network and as fuzzy system.

The model uses the fuzzy rules of a TSK controller.

Example of an ANFIS Model



This is a model with three fuzzy rules:

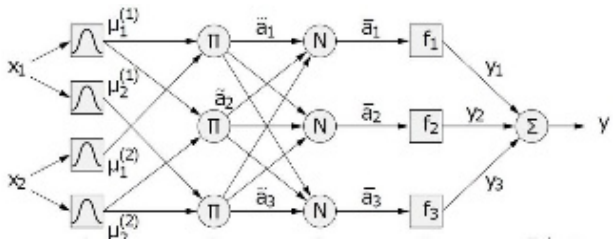
$$R_1: \quad \text{If } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } B_1 \text{ then } y = f_1(x_1, x_2)$$

$$R_2: \quad \text{If } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } B_2 \text{ then } y = f_2(x_1, x_2)$$

$$R_3: \quad \text{If } x_1 \text{ is } A_2 \text{ and } x_2 \text{ is } B_2 \text{ then } y = f_3(x_1, x_2)$$

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



Here, 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.

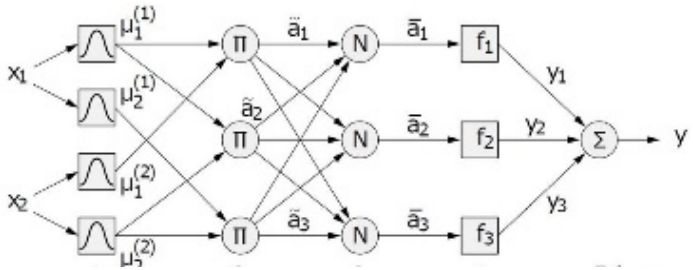
A fuzzification neuron receives a crisp input and determines the degree to which this input belongs to the neuron's fuzzy set.

Usually bell-curved functions are used: $$\mu_i^{(j)}(x_j) = \frac{1}{1 + \left(\frac{x_j - a_i}{b_i}\right)^{2c_i}}$$

where a_i , b_i , c_i are parameters for center, width, and slope, resp.

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 TSK fuzzy rule.

A fuzzy rule neuron receives inputs from the fuzzification neurons

that represent fuzzy sets in the rule antecedents.

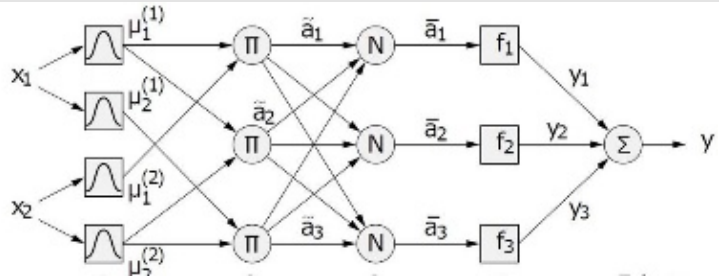
It calculates the firing strength of the corresponding rule.

In NFS, the intersection is usually implemented by the product.

So, the firing strength $\tilde{a_i}$ of rule R_i is

$$\tilde{a}_i = \prod_{j=1}^{k} \mu_i^{(j)}(x_j).$$

ANFIS: Layer 3 – The Normalization Layer



Each neuron in this layer receives the firing strengths from all neurons in the rule layer.

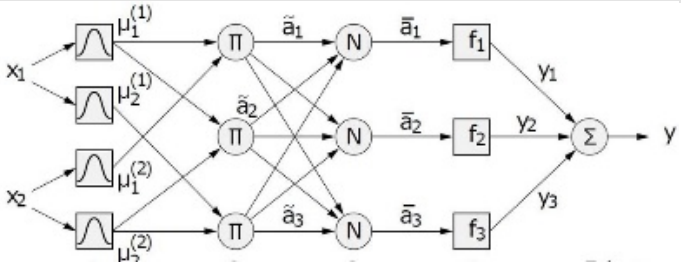
The normalised firing strength of a given rule is calculated here. It

represents the contribution of a given rule to the final result.

Thus, the output of neuron *i* in layer 4 is determined as

$$\bar{a}_i = a_i = \mathrm{net}_i = \frac{\tilde{a}_i}{\sum_j \tilde{a}_j}.$$

ANFIS: Layers 4 and 5 – Defuzzification and Summation



Each neuron in layer 4 is connected to the respective normalisation neuron, and also receives the raw input values ***x***.

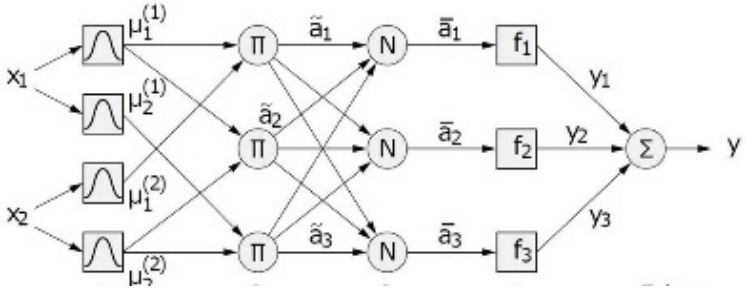
A defuzzification neuron calculates the weighted consequent value of a given rule as

$$\bar{y}_i = a_i = \mathrm{net}_i = \bar{a}_i f_i(x_1, \ldots, x_n).$$

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

$$y = f(\boldsymbol{x}_i) = a_{\mathrm{out}} = \mathrm{net}_{\mathrm{out}} = \sum_i \bar{y}_i = \frac{\sum_i \tilde{a}_i f_i(x_1, \ldots, x_n)}{\sum_i \tilde{a}_i}.$$

How does ANFIS learn?



Supervised Learning for Parameter Tuning

Goal: minimize mean squared error iteratively

ANFIS uses a hybrid learning algorithm that combines least-squares and gradient descent (backpropagation)

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.

How does ANFIS learn? – The Forward Pass

r_{ij}: parameters of output function f_i, $x_i(k)$: input values, $y(k)$: output value of k-th training pair, $\bar{a}_i(k)$: relative control activation

Then we obtain

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

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

$$\mathbf{y} = \bar{\mathbf{a}}\mathbf{R}\mathbf{X}$$

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.

The NEFCLASS Model

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 such as NEFCLASS are based on the idea of backpropagation but constrain the learning process to ensure interpretability.

NEFCLASS has been used in lots of applications, e.g. at British Telecom.

Example: Wisconsin Breast Cancer Dataset

The *Wisconsin Breast Cancer Dataset* stores results from patients tested for breast cancer.

These data can be used to train and evaluate classifiers.

For instance, decision support systems must tell if unseen data indicate malignant or benign case?

A surgeon must be able to check this classification for plausibility.

We are looking for a simple and interpretable classifier.

Example: WBC Data Set

699 cases (16 cases have missing values).

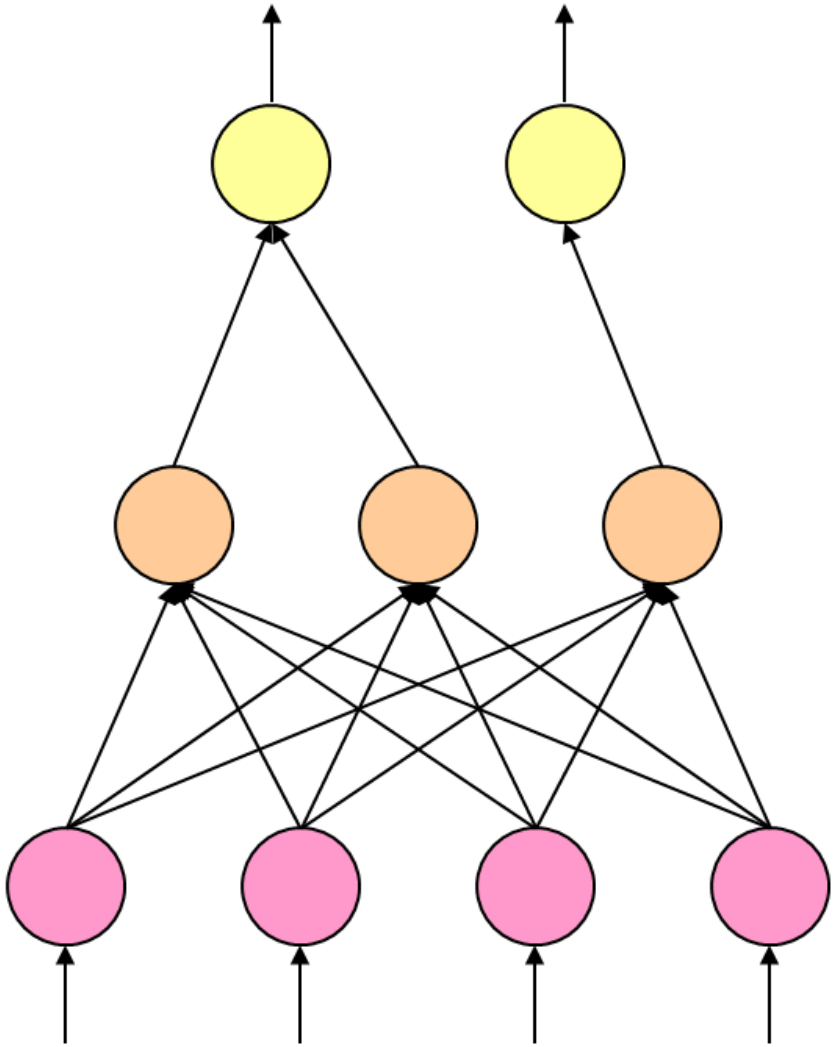
2 classes: benign (458), malignant (241).

9 attributes with values from $\{1, \ldots, 10\}$ (ordinal scale, but usually interpreted numerically).

In the following, x_3 and x_6 are interpreted as nominalattributes.

x_3 and x_6 are usually seen as “important” attributes.

NEFCLASS: Neuro-Fuzzy Classifier



output variables

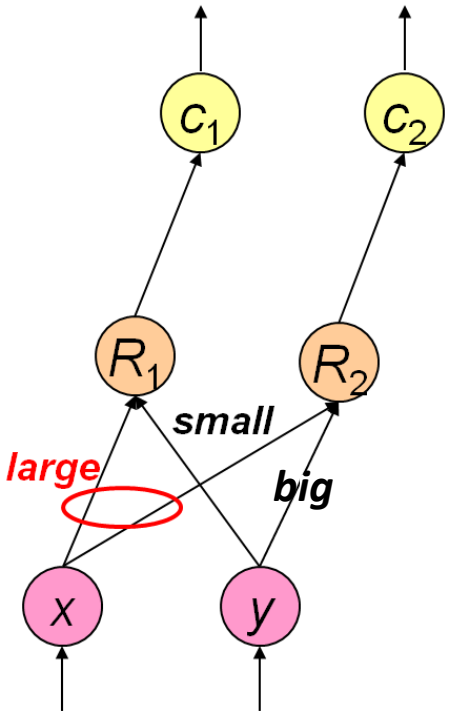
unweighted connections

fuzzy rules

fuzzy sets (antecedents)

input attributes (variables)

Representation of Fuzzy Rules



Example: 2 rules

R_1 : if x is *large* and y is *small*, then class is c_1

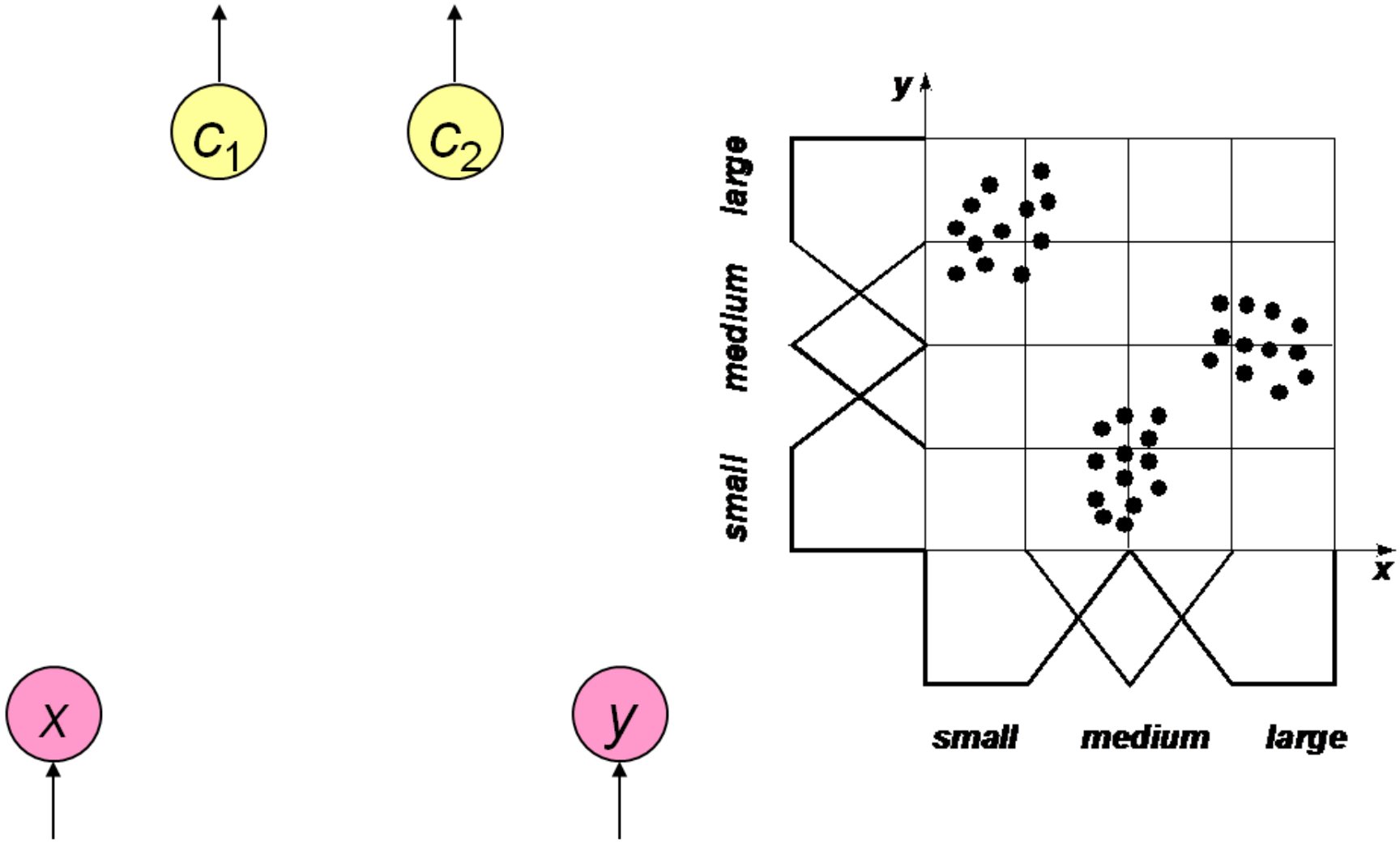
R_2 : if x is *large* and y is *big*, then class is c_2

Connections $x \rightarrow R_1$ and $x \rightarrow R_2$ are linked.

Fuzzy set *large* is a shared weight,
i.e. the term *large* for x has always the same meaning in both rules.

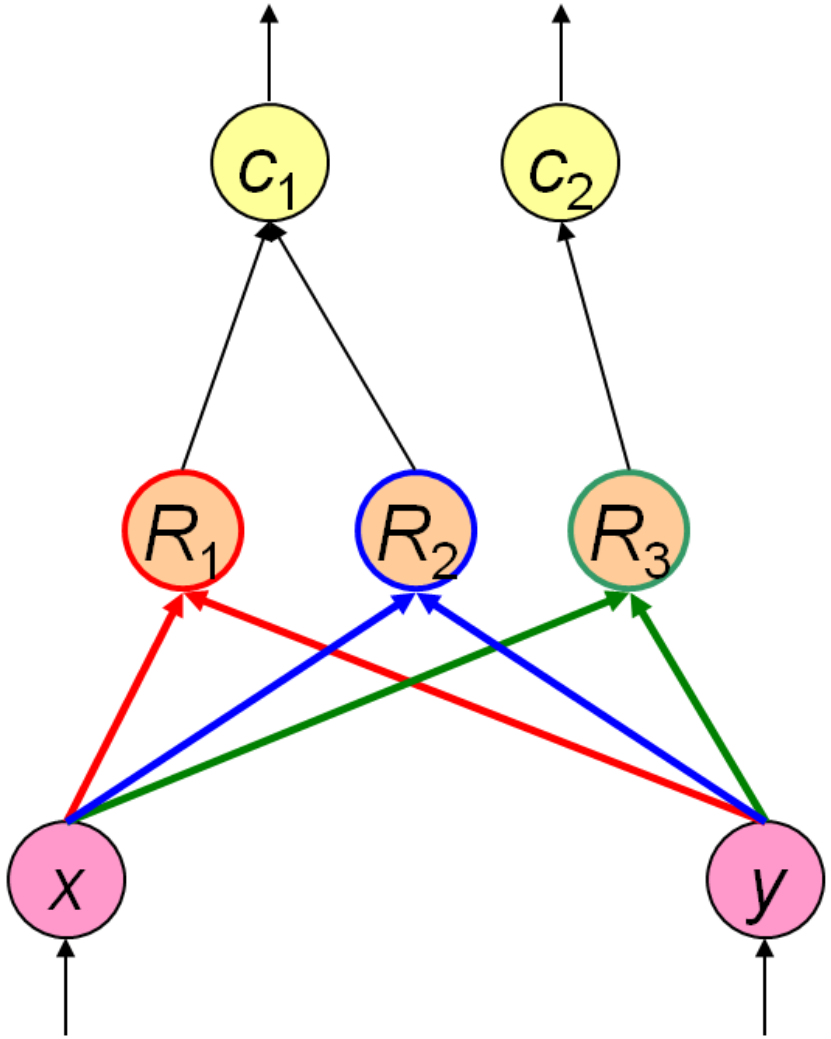
1. Initialization

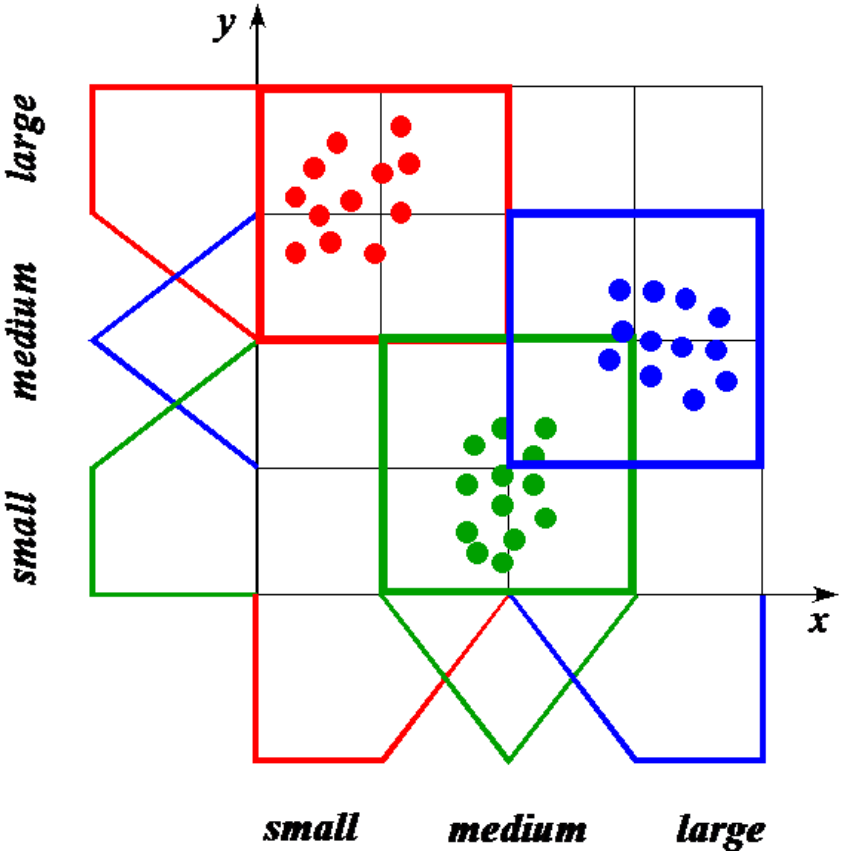
Specify initial fuzzy partitions for all input variables.



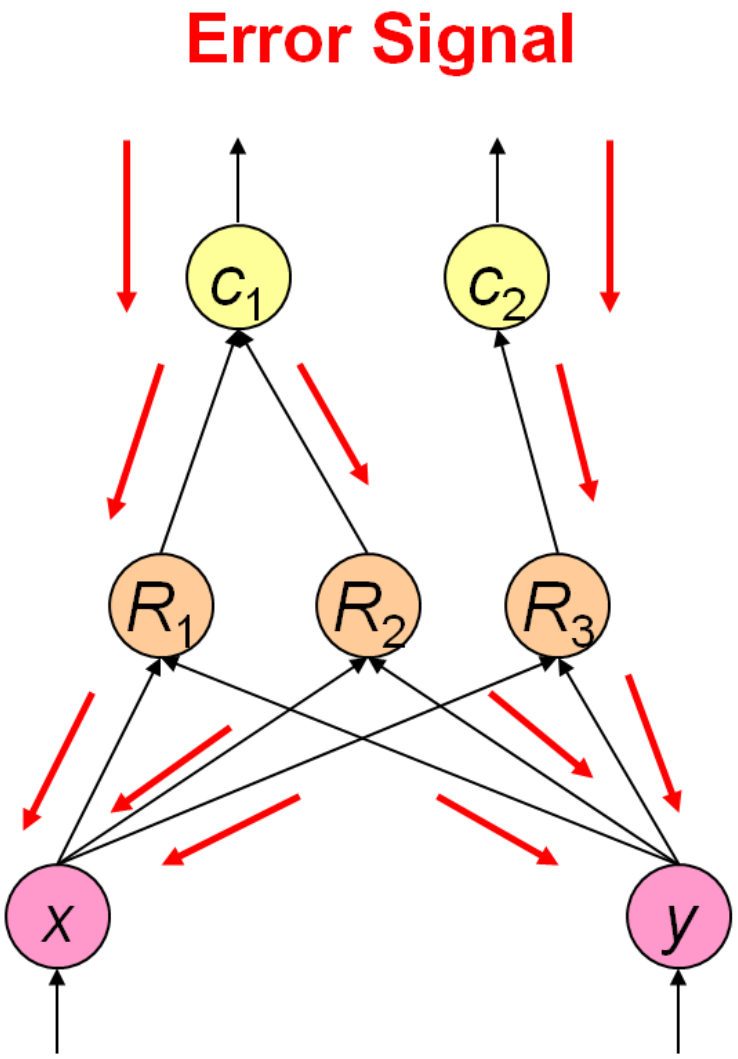
2. Rule Base Induction

NEFCLASS uses modified Wang-Mendel procedure.

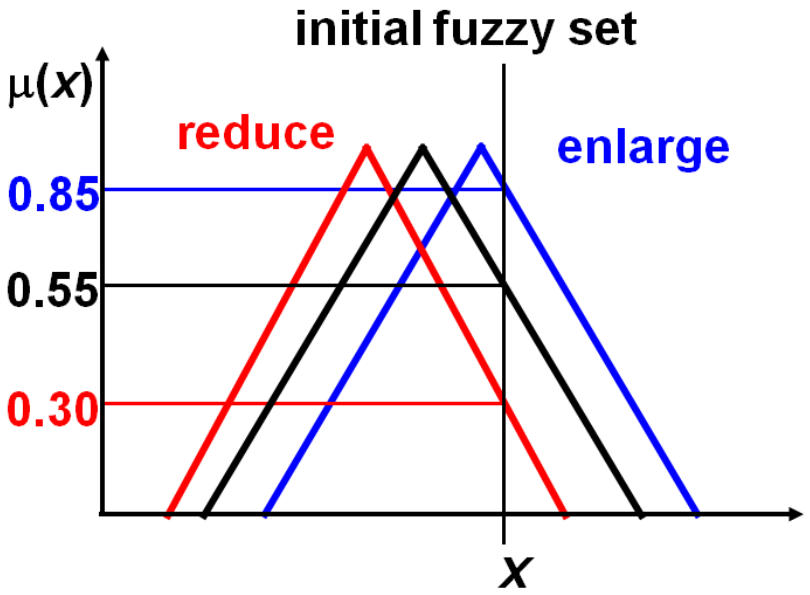


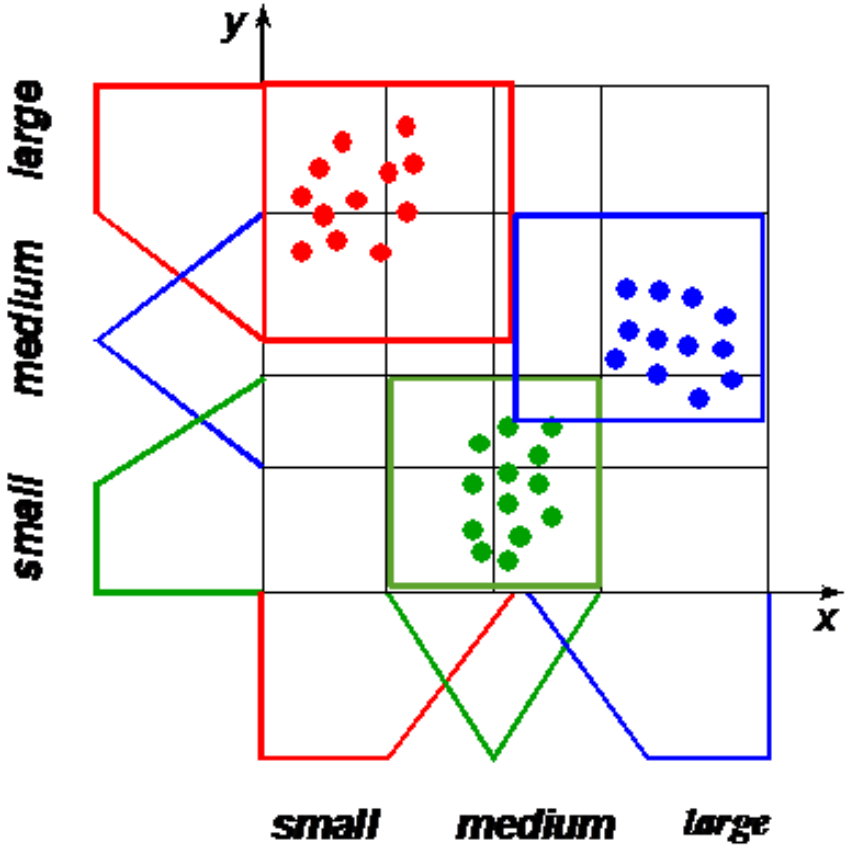


3. Improving Rules by Backpropagation of the Fuzzy Error Signal



4. Improving Fuzzy Sets by Heuristics

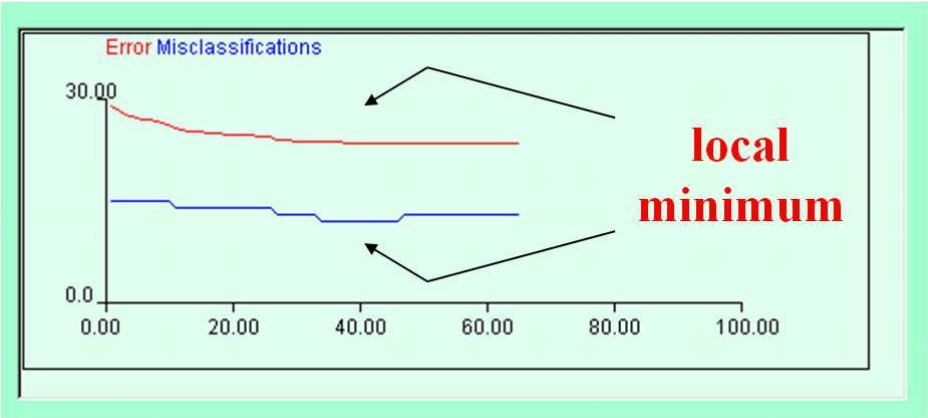




Heuristics: A fuzzy set is moved away from *x* (towards *x*) and its support is reduced (enlarged) in order to reduce (enlarge) the degree of membership of *x*.

4. Improving Fuzzy Sets by Heuristics

do {
  for each pattern {
    accumulate parameter updates
    accumulate error
  }
  modify parameters
} **while** change in error



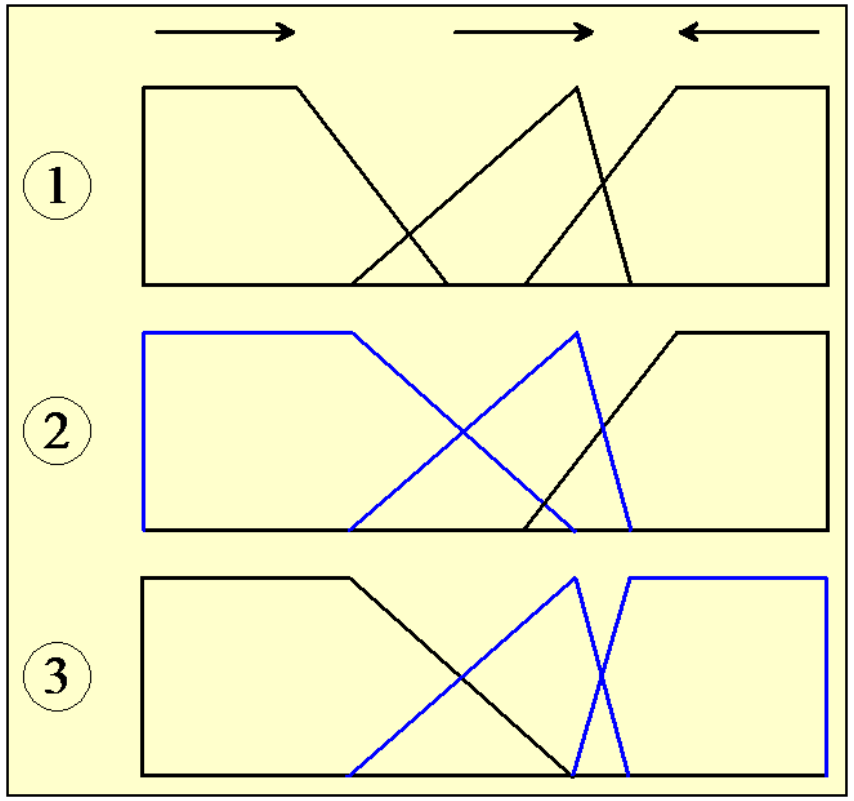
observing the error on validation set

variations:

- adaptive learning rate
- online/batch learning
- optimistic learning (*n* step look ahead)

Constraints for Training Fuzzy Sets

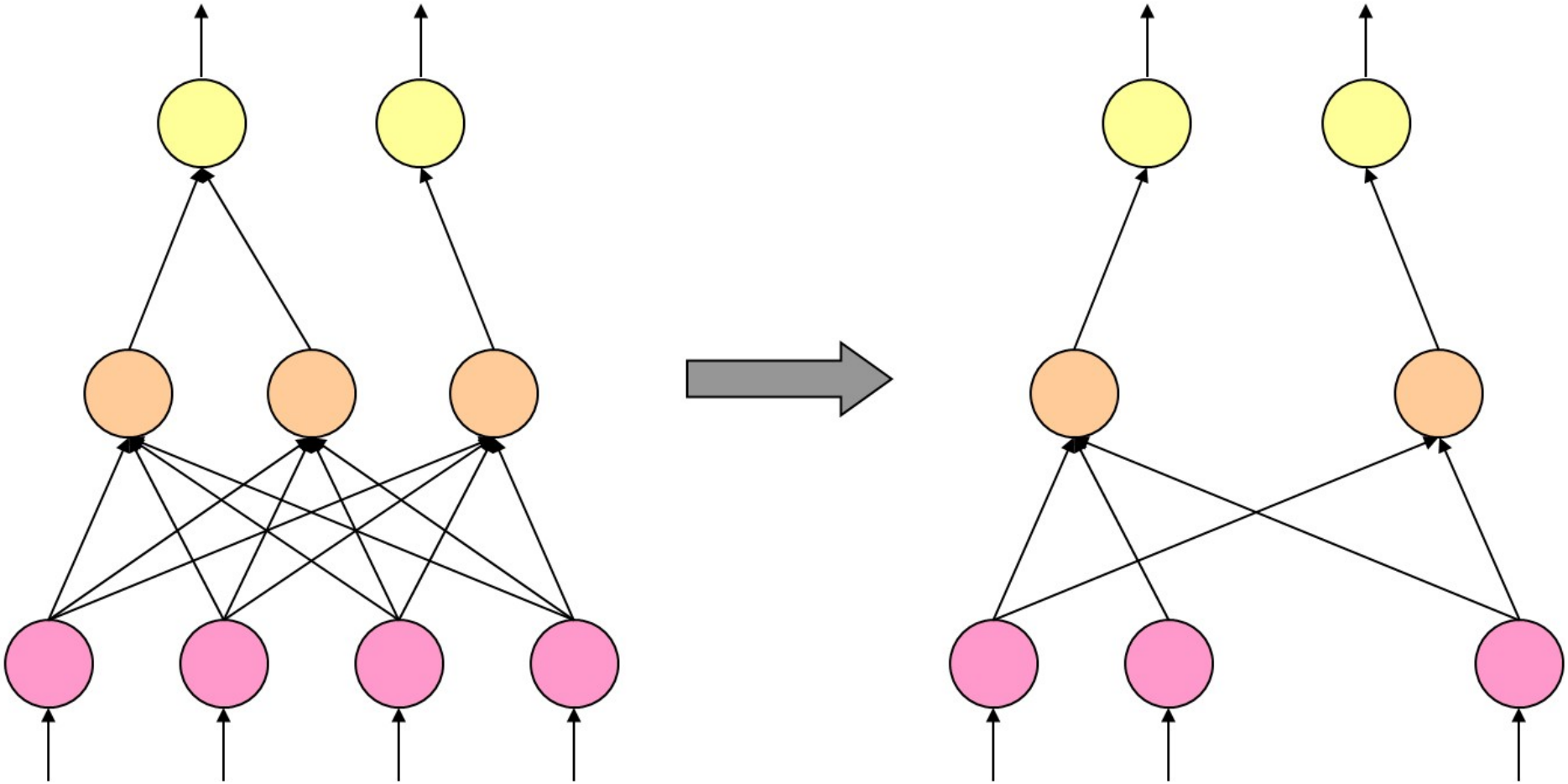
- valid parameter values
- non-empty intersection of adjacent fuzzy sets
- keep relative positions
- maintain symmetry
- complete coverage (degrees of membership add up to 1 for each element)



Correcting a partition after modifying the parameters

5. Pruning

Goal: Remove variables, rules, and fuzzy sets in order to improve the interpretability and generalization.



Example: WBC Data Classification

Automatic Generation of (only) two Fuzzy Rules:

R_1: if uniformity of cell size is *small* and bare nuclei is fuzzy0 then *benign*

R_2: if uniformity of cell size is *large* then *malignant*

The main strength of NEFCLASS is its interpretability and simplicity.

Example: WBC Classification Performance

	predicted class					
	malign		benign		$\sum$	
malign	228	(32.62%)	13	(1.86%)	241	(34.99%)
benign	15	(2.15%)	443	(63.38%)	458	(65.01%)
$\sum$	243	(34.76)	456	(65.24)	699	(100.00%)

Estimated performance on unseen data (cross validation):

NEFCLASS:	95.42%
Discriminant Analysis:	96.05%
C 4.5:	95.10%

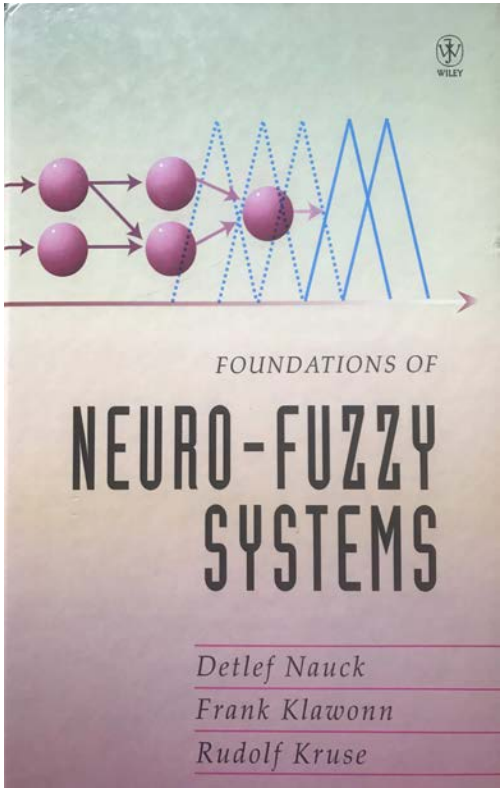
Summary

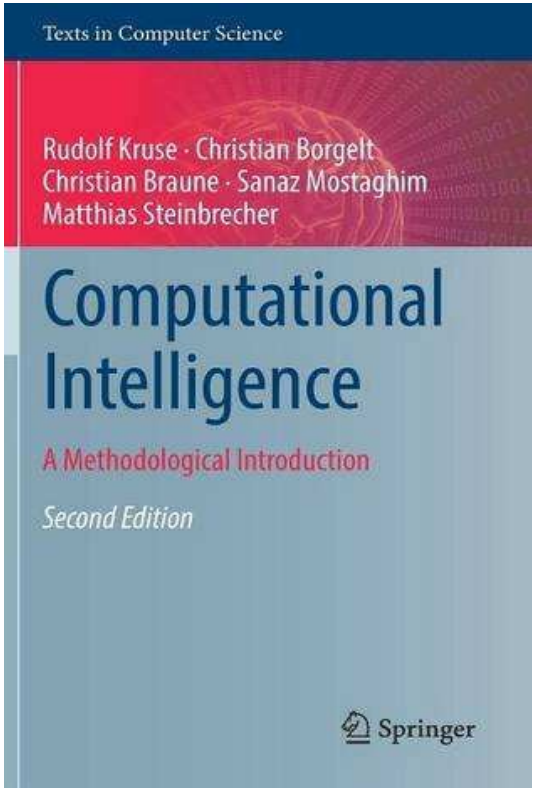
Neuro-fuzzy systems can be very useful for knowledge discovery.

The interpretability enables plausibility checks and improves the acceptance.

(Neuro-)fuzzy systems exploit the tolerance for sub-optimal solutions.

There is no automatic model creator! The user must **work** with the tool!


WILEY
FOUNDATIONS OF
NEURO-FUZZY SYSTEMS
Detlef Nauck
Frank Klawonn
Rudolf Kruse


Texts in Computer Science
Rudolf Kruse · Christian Borgelt
Christian Braune · Sanaz Mostaghim
Matthias Steinbrecher
Computational Intelligence
A Methodological Introduction
Second Edition
Springer