Hybrid Models:
Fuzzy-Neural Applications

1

Lecture Outline

- Hybrid System
- Property Value Example
- Constructibility Analysis Example

2

Introduction

- A **hybrid intelligent system** is one that combines at least two intelligent technologies. For example, combining a neural network with a fuzzy system results in a hybrid neuro-fuzzy system.

- The combination of probabilistic reasoning, fuzzy logic, neural networks and evolutionary computation forms the core of **soft computing**, an emerging approach to building hybrid intelligent systems capable of reasoning and learning in an uncertain and imprecise environment.

- Although words are less precise than numbers, precision carries a high cost. We use words when there is a tolerance for imprecision. Soft computing exploits the tolerance for uncertainty and imprecision to achieve greater tractability and robustness, and lower the cost of solutions.

- We also use words when the available data is not precise enough to use numbers. This is often the case with complex problems, and while "hard" computing fails to produce any solution, soft computing is still capable of finding good solutions.

Lord Stafford 2009 Award: Video Application of Hybrid Neuro Fuzzy Decision Support System

- This video features an intelligent data analysis and decision support system which was developed based on a patented hybrid neuro-fuzzy based approach (Patent No: WO/2009/141631).
- The system was developed as part of a partnership between the University of Essex and an Ipswich-based business and received the prestigious Lord Stafford Achievement in Innovation award for East of England 2009.
- Video

5

- Lotfi Zadeh is reputed to have said that a good hybrid would be “British Police, German Mechanics, French Cuisine, Swiss Banking and Italian Love”. But **“British Cuisine, German Police, French Mechanics, Italian Banking and Swiss Love” would be a bad one.**

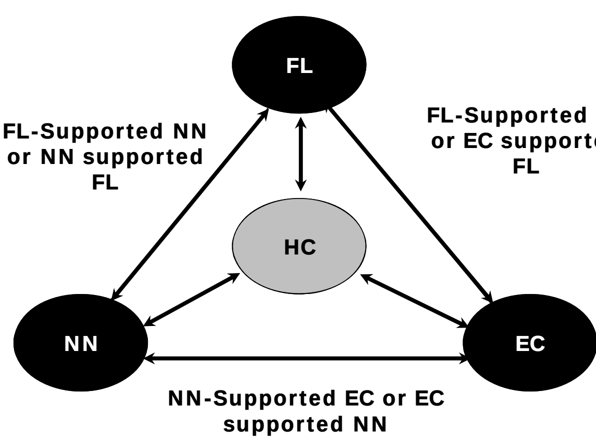
- Likewise, a hybrid intelligent system can be good or bad – it depends on which components constitute the hybrid. So our goal is to select the right components for building a good hybrid system.

Comparison of Expert Systems, Fuzzy Systems, Neural Networks and Genetic Algorithms

	ES	FS	NN	GA
Knowledge representation	rather good	good	bad	rather bad
Uncertainty tolerance	rather good	good	good	good
Imprecision tolerance	bad	good	good	good
Adaptability	bad	rather bad	good	good
Learning ability	bad	bad	good	good
Explanation ability	good	good	bad	rather bad
Knowledge discovery and data mining	bad	rather bad	good	rather good
Maintainability	bad	rather good	good	rather good

* The terms used for grading are:
- bad, - rather bad, - rather good and - good

Hybrid Systems



8

Neural expert systems

- Expert systems rely on logical inferences and decision trees and focus on modelling human reasoning. Neural networks rely on parallel data processing and focus on modelling a human brain.
- Expert systems treat the brain as a black-box. Neural networks look at its structure and functions, particularly at its ability to learn.
- Knowledge in a rule-based expert system is represented by IF-THEN production rules. Knowledge in neural networks is stored as synaptic weights between neurons.

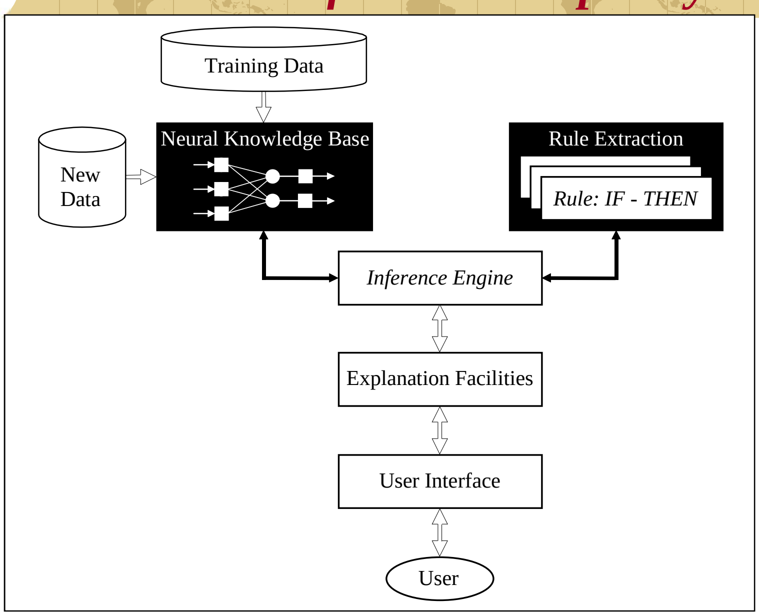
- In expert systems, knowledge can be divided into individual rules and the user can see and understand the piece of knowledge applied by the system.

- In neural networks, one cannot select a single synaptic weight as a discrete piece of knowledge. Here knowledge is embedded in the entire network; it cannot be broken into individual pieces, and any change of a synaptic weight may lead to unpredictable results. A neural network is, in fact, a **black-box** for its user.

Can we combine advantages of expert systems and neural networks to create a more powerful and effective expert system?

A hybrid system that combines a neural network and a rule-based expert system is called a **neural expert system** (or a **connectionist expert system**).

Basic structure of a neural expert system



The heart of a neural expert system is the **inference engine**. It controls the information flow in the system and initiates inference over the neural knowledge base. A neural inference engine also ensures **approximate reasoning**.

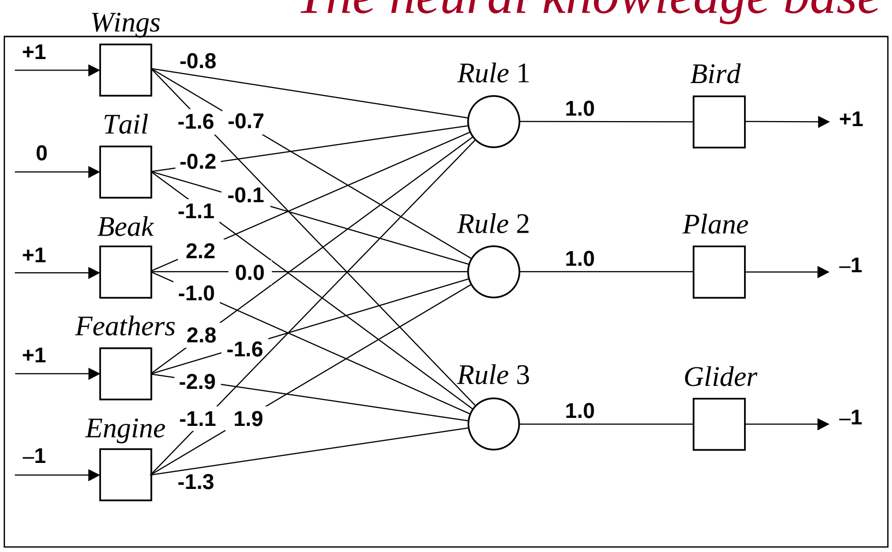
Approximate reasoning

- In a rule-based expert system, the inference engine compares the condition part of each rule with data given in the database. When the IF part of the rule matches the data in the database, the rule is fired and its THEN part is executed. The **precise matching** is required (inference engine cannot cope with noisy or incomplete data).
- Neural expert systems use a trained neural network in place of the knowledge base. The input data does not have to precisely match the data that was used in network training. This ability is called **approximate reasoning**.

Rule extraction

- Neurons in the network are connected by links, each of which has a numerical weight attached to it.
- The weights in a trained neural network determine the strength or importance of the associated neuron inputs.

The neural knowledge base



If we set each input of the input layer to either +1 (true), –1 (false), or 0 (unknown), we can give a semantic interpretation for the activation of any output neuron. For example, if the object has *Wings* (+1), *Beak* (+1) and *Feathers* (+1), but does not have *Engine* (–1), then we can conclude that this object is *Bird* (+1):

$$X_{Rule\,1} = 1\cdot(-0.8) + 0\cdot(-0.2) + 1\cdot 2.2 + 1\cdot 2.8 + (-1)\cdot(-1.1) = 5.3 > 0$$
$$Y_{Rule\,1} = Y_{Bird} = +1$$

We can similarly conclude that this object is not *Plane*:

$$X_{Rule\,2} = 1\cdot(-0.7) + 0\cdot(-0.1) + 1\cdot 0.0 + 1\cdot(-1.6) + (-1)\cdot 1.9 = -4.2 < 0$$
$$Y_{Rule\,2} = Y_{Plane} = -1$$

and not *Glider*:

$$X_{Rule\,3} = 1\cdot(-0.6) + 0\cdot(-1.1) + 1\cdot(-1.0) + 1\cdot(-2.9) + (-1)\cdot(-1.3) = -4.2 < 0$$
$$Y_{Rule\,3} = Y_{Glider} = -1$$

By attaching a corresponding question to each input neuron, we can enable the system to prompt the user for initial values of the input variables:

Neuron: *Wings*
Question: Does the object have wings?
Neuron: *Tail*
Question: Does the object have a tail?
Neuron: *Beak*
Question: Does the object have a beak?
Neuron: *Feathers*
Question: Does the object have feathers?
Neuron: *Engine*
Question: Does the object have an engine?

An inference can be made if the known net weighted input to a neuron is greater than the sum of the absolute values of the weights of the unknown inputs.

$$\sum_{i=1}^{n} x_i w_i > \sum_{j=1}^{n} \left| w_j \right|$$

where $i \in$ known, $j \notin$ known and n is the number of neuron inputs.

Example:

Enter initial value for the input Feathers:

$>$ +1

KNOWN = 1·2.8 = 2.8

UNKNOWN = $|-0.8| + |-0.2| + |2.2| + |-1.1| = 4.3$

KNOWN < UNKNOWN

Enter initial value for the input Beak:

$>$ +1

KNOWN = 1·2.8 + 1·2.2 = 5.0

UNKNOWN = $|-0.8| + |-0.2| + |-1.1| = 2.1$

KNOWN > UNKNOWN

CONCLUDE: Bird is TRUE

An example of a multi-layer knowledge base

Rule 1:
IF *a*1 AND *a*3 THEN *b*1 (0.8)

Rule 2:
IF *a*1 AND *a*4 THEN *b*1 (0.2)

Rule 3:
IF *a*2 AND *a*5 THEN *b*2 (-0.1)

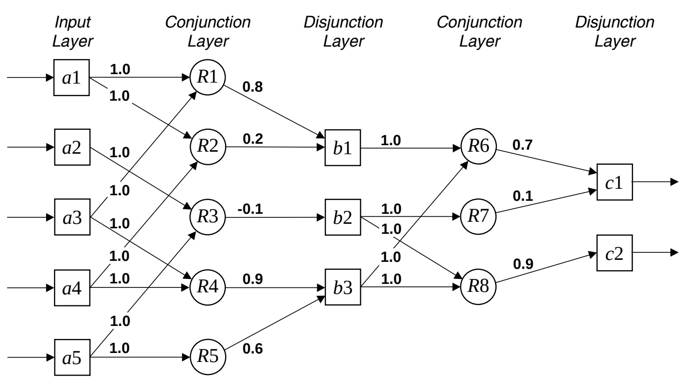
Rule 4:
IF *a*3 AND *a*4 THEN *b*3 (0.9)

Rule 5:
IF *a*5 THEN *b*3 (0.6)

Rule 6:
IF *b*1 AND *b*3 THEN *c*1 (0.7)

Rule 7:
IF *b*2 THEN *c*1 (0.1)

Rule 8:
IF *b*2 AND *b*3 THEN *c*2 (0.9)



Neuro-fuzzy systems

- Fuzzy logic and neural networks are natural complementary tools in building intelligent systems.
- While neural networks are low-level computational structures that perform well when dealing with raw data, fuzzy logic deals with reasoning on a higher level, using linguistic information acquired from domain experts.
- However, fuzzy systems lack the ability to learn and cannot adjust themselves to a new environment. On the other hand, although neural networks can learn, they are opaque to the user.

- Integrated neuro-fuzzy systems can 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.

- A neuro-fuzzy system is a neural network which is functionally equivalent to a fuzzy inference model.
- It can be trained to develop IF-THEN fuzzy rules and determine membership functions for input and output variables of the system.
- Expert knowledge can be incorporated into the structure of the neuro-fuzzy system. At the same time, the connectionist structure avoids fuzzy inference, which entails a substantial computational burden.

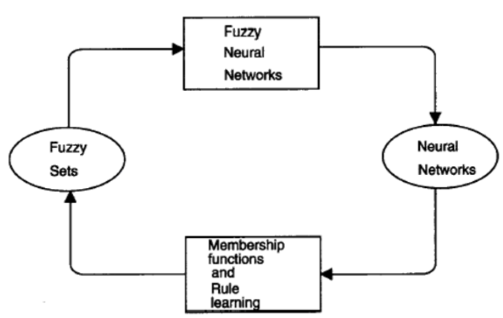
- The structure of a neuro-fuzzy system is similar to a multi-layer neural network.
- In general, a neuro-fuzzy system has input and output layers, and three hidden layers that represent membership functions and fuzzy rules.

Neuro-Fuzzy Systems

- Neuro-fuzzy systems
  - Soft computing methods that combine in various ways neural networks and fuzzy concepts
- ANN – nervous system – low level perceptive and signal integration
- Fuzzy part – represents the emergent "higher level" reasoning aspects

27

Neuro-Fuzzy Systems



- "Fuzzification" of neural networks
- Endowing of fuzzy system with neural learning features

28

Neuro-Fuzzy Systems

- Co-operative-neural algorithm adapt fuzzy systems
  - Off-line – adaptation
  - On-line – algorithms are used to adapt as the system operates
- Concurrent – where the two techniques are applied after one another as pre- or post-processing
- Hybrid – fuzzy system being represented as a network structure, making it possible to take advantage of learning algorithm inherited from ANNs

29

Neuro-Fuzzy Systems

- Neural networks are good at recognizing patterns, they are not good at explaining how they reach their decisions.
- Fuzzy logic systems, which can reason with imprecise information, are good at explaining their decisions but they cannot automatically acquire the rules they use to make those decisions.
- These limitations have been a central driving force behind the creation of intelligent hybrid systems where two or more techniques are combined in a manner that overcomes individual techniques

30

Industrial Production Problem

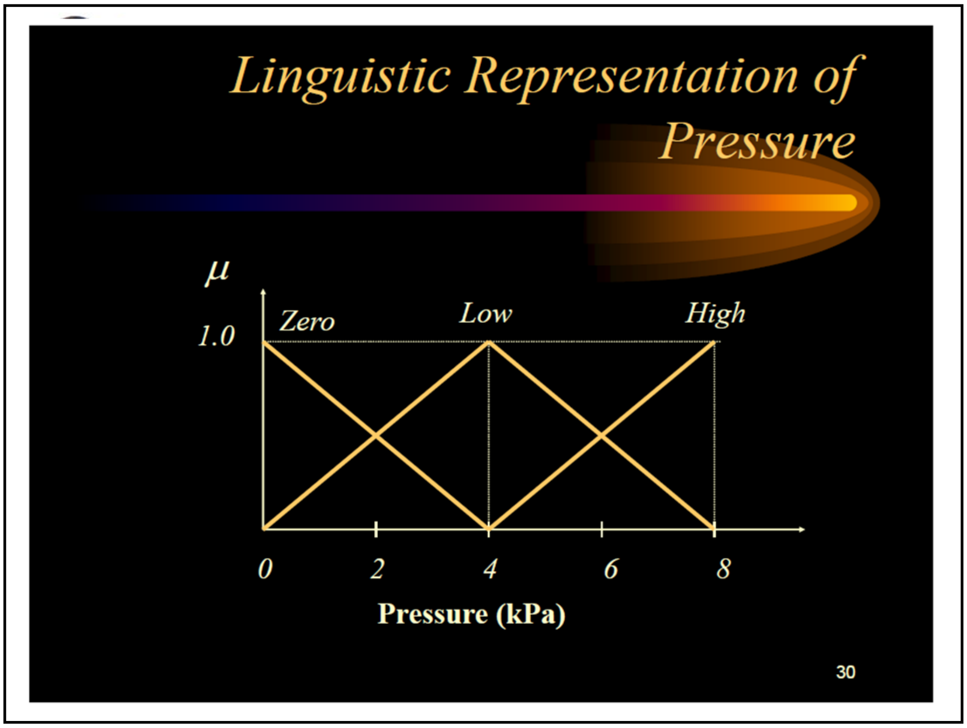
28

Industrial Production Problem

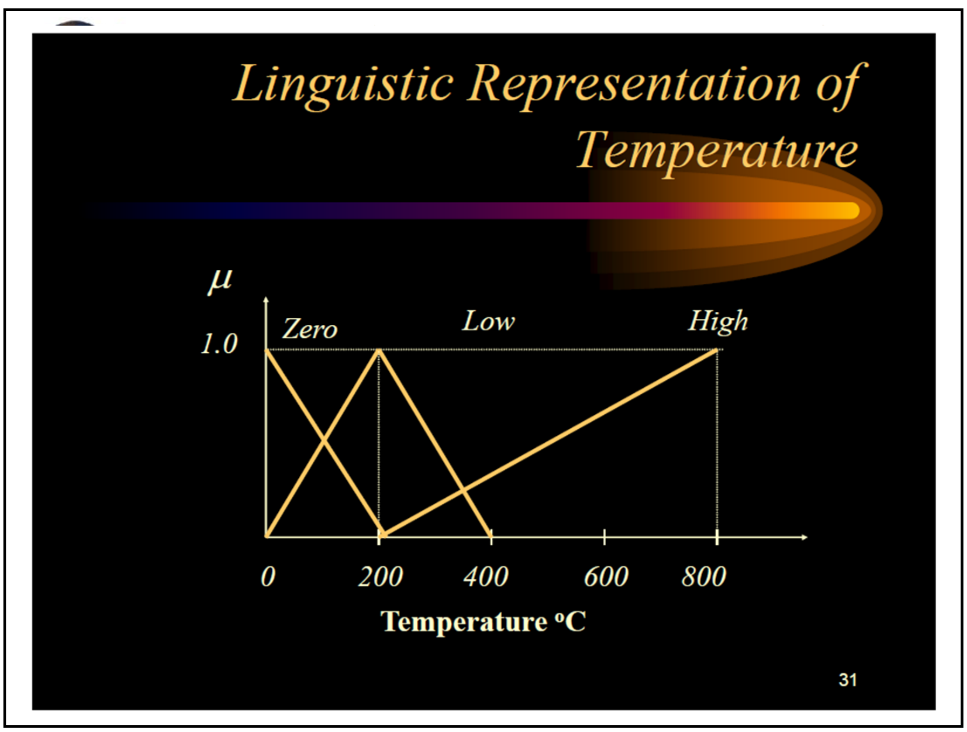
- Production Process – Involves three features
  - ***Pressure***
  - ***Temperature***
  - ***Flow Rate***

Pattern Recognition task – The system reads sensor indicators of each features as crisp read-out values – Determine the current mode of operation

29

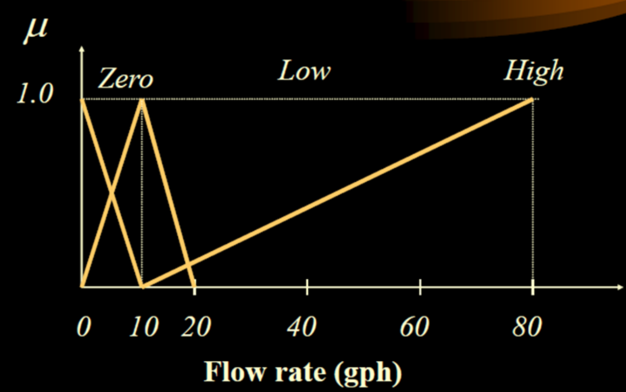


Linguistic Representation of Pressure
μ
1.0
Zero
Low
High
0
2
4
6
8
Pressure (kPa)
30



Linguistic Representation of Temperature
μ
1.0
Zero
Low
High
0
200
400
600
800
Temperature °C
31

Linguistic Representation of Flow rate



32

Relationship between operation mode and feature values

Mode (Pattern)	Pressure	Temperature	Flow Rate
Autoclaving	High	High	Zero
Annealing	High	Low	Zero
Sintering	Low	Zero	Low
Transport	Zero	Zero	High

33

Problem Objective

- System reads from a set of sensors a set of Crisp readings
- Pressure = 5 kPa
- Temperature = 150 °C
- Flow rate = 5 gph (gallon per hour)

Find Mode of Operation

34

Assumption

- Let us decide to apply weights to each features
- Feature - Pressure can be more hazardous in comparison to other features, hence provide higher weights to it.
  - $W_{pressure} = 0.5$
  - $W_{temperature} = 0.25$
  - $W_{flow} = 0.25$

35

We have

- X = { 5 kPa, 150 ^oC, 5 gph}
- W = {0.5, 0.25, 0.25}

Computation Membership Values for Each Operations

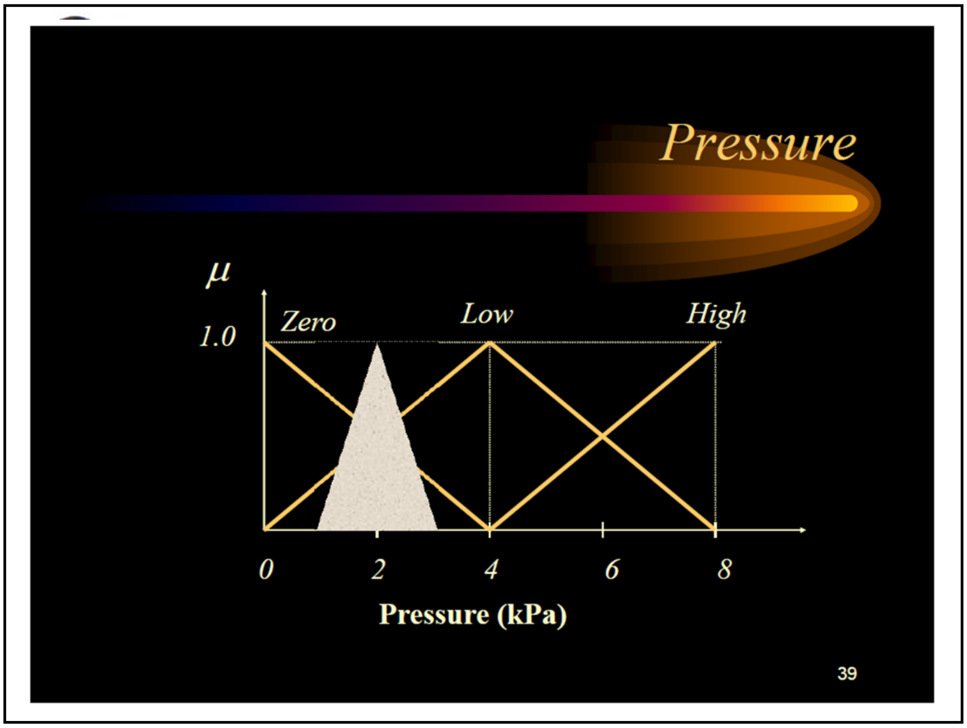
$\mu_{autoclaving}$ = (0.5).(0.25) + (0.25).(0) + (0.25).(0.5)/1= 0.25

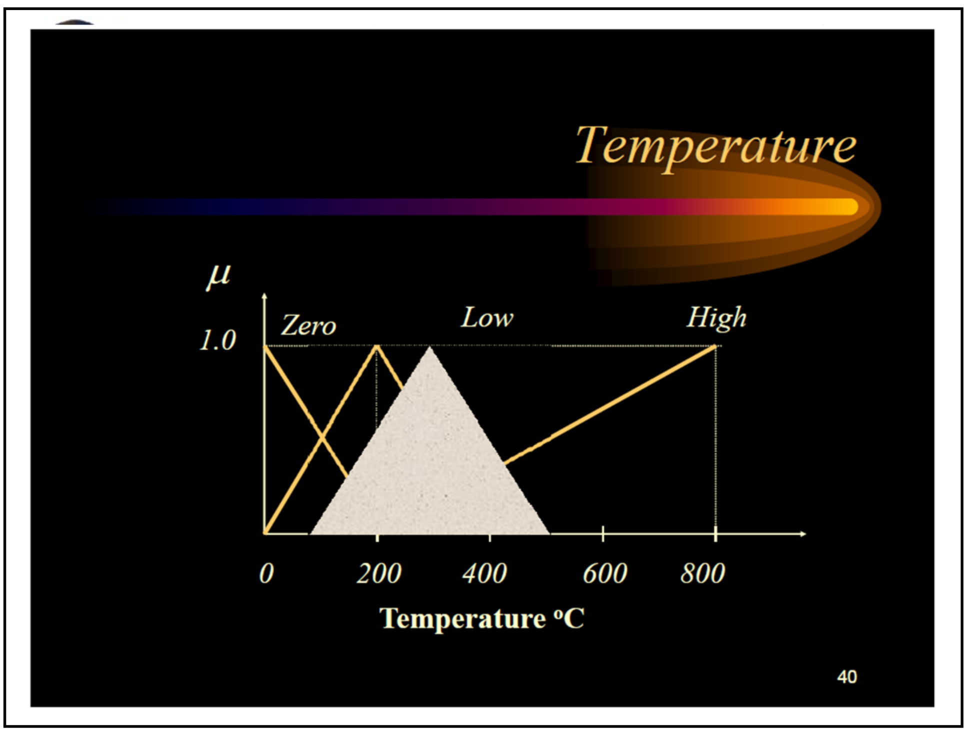
$\mu_{annealing}$ = (0.5).(0.25) + (0.25).(0.75) + (0.25).(0.5)/1 = 0.4375

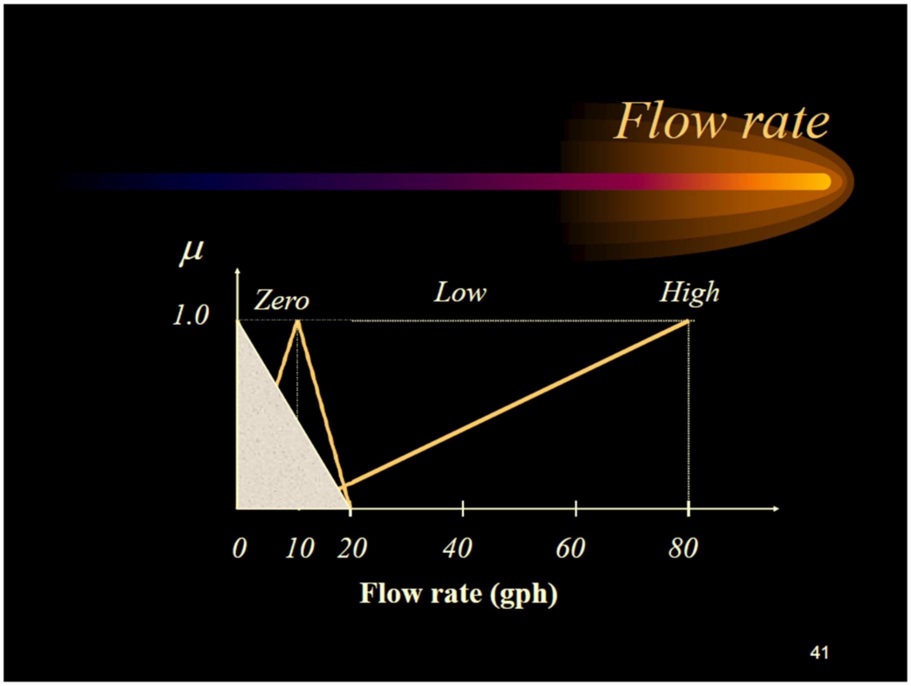
$\mu_{sintering}$ = (0.5).(0.75) + (0.25).(0.25) + (0.25).(0.5)/1 = ***0.5625***

$\mu_{transport}$ = (0.5).(0) + (0.25).(0.25) + (0.25).(0)/1 = 0.0625

Therefore, X = { 5 kPa, 150 ^oC, 5 gph} matches most closely with ***Sintering***


Pressure
μ
1.0
Zero
Low
High
0
2
4
6
8
Pressure (kPa)
39


Temperature
μ
1.0
Zero
Low
High
0
200
400
600
800
Temperature °C
40



Equal Weights

- Inputs : Fuzzy Membership functions for Pressure, Temperature and Flow Rate
- Output: 4 Outputs – 1 for the actual process and 0 for other processes
- Architecture: 9 – Hidden Layers – 4

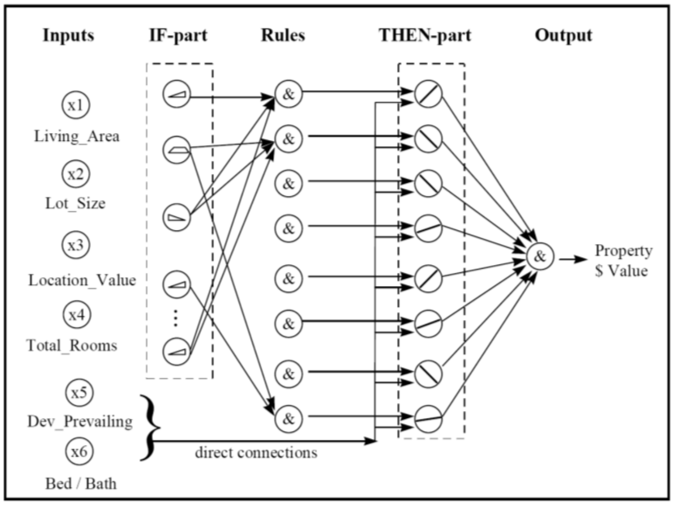
44

Different Weights

- Inputs : Fuzzy Membership functions for Pressure, Temperature and Flow Rate
- Output: Fuzzy Membership function of the Process
- Architecture: 9 – Hidden Layers – 3
- Map the output membership function on the final template and apply the criteria.

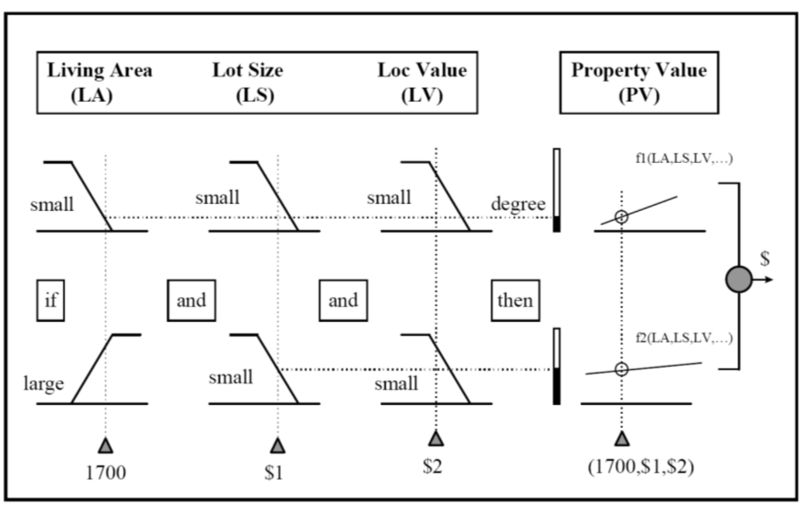
An Example: Property Value

Property Value



47

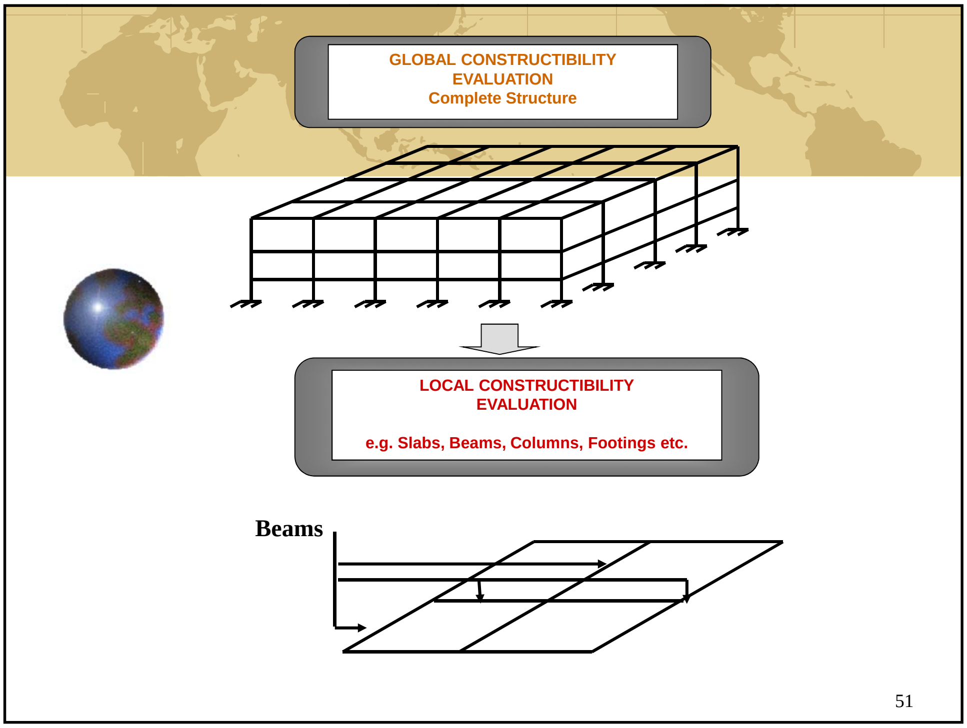
Property Value



An Example: Constructability Analysis

Problem Definition

- Decision Support Systems
- Constructibility Analysis
- Global and Local Constructibility



Objectives

- Qualitative Data Modeling – NeuroFuzzy Model
- Comparative Study
- Performance Reliability
- Missing Data Handling

52

Data Collection

- Dataset describes the constructibility of a beam in 12-storied building
- *Source:* Skibniewski, M., Arciszewski, T., Lueprasert, K., "Constructibility analysis: Machine learning approach." *J. of Computing in Civil Engrg* , ASCE, Vol. 10, No. 3, 1997, pp. 6-16.
- 31 Data Examples
- Total Number of Input Variables = 7
- Total Number of Output Variable = 1

53

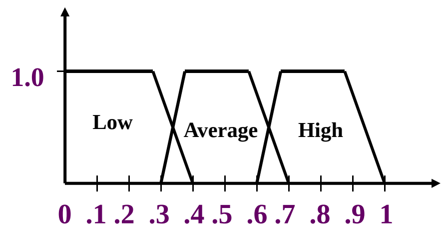
Input Data Variables

Variable Name	Description	Fuzzy Variables/ Binary Variable
ReRa	Reinforcement Ratio	Low, Average, High
CoBeRe1	First Beam to Column Connection	-do-
CoBeRe2	Second Beam to Column Connection	-do-
BeCha1	Change in Steel Reinforcements and Size of the Beam on the Left or First Side of the Beam	None, SliChgeReinf, AllChgeReinf, WDchange, AllChange
BeCha2	Change in Steel Reinforcement and Size of the Beam on the Right or Second Side of the Beam	-do-
NoSla	Number of Slabs Attached to Beam	None, One, SameTwo, DiffTwo
NoWall	Number of Walls Attached to Beam	-do-

54

Input Data Variables

Variable Name	Description	Fuzzy Variables
ReRa	Reinforcement Ratio	Low, Average, High
CoBeRe1	First Beam to Column Connection	-do-
CoBeRe2	Second Beam to Column Connection	-do-

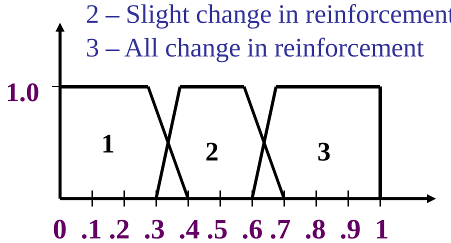


55

Input Data Variables

Variable Name	Description	Fuzzy Variables/ Binary Variable
BeCha1	Change in Steel Reinforcements and Size of the Beam on the Left or First Side of the Beam	None, SliChgeReinf, AllChgeReinf, WDchange, AllChange
BeCha2	Change in Steel Reinforcement and Size of the Beam on the Right or Second Side of the Beam	-do-

1 – No change in reinforcement
2 – Slight change in reinforcement
3 – All change in reinforcement



+

No width depth change	Width depth change
0 0	1 1

56

Input Data Variables

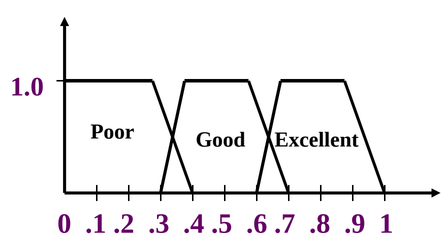
Variable Name	Description	Binary Variable
NoSla	Number of Slabs Attached to Beam	None, One, SameTwo, DiffTwo
NoWall	Number of Walls Attached to Beam	-do-

None	One	Same Two	DiffTwo
0 0	0 1	1 0	1 1

57

Output Data Variables

Variable Name	Description	Fuzzy Variables
ConEva	Constructibility Evaluation	Poor, Good, Excellent



58

Model Implementation

- Backpropagation based neural networks implemented in 'C' programming environment
- Number of Hidden Layers = 2
- Number of Hidden Units per Hidden Layer = 48
  Momentum rate = 0.9
- Learning rate = 0.7
- Maximum Total Error = 0.001
- Maximum Individual Error = 0.0001

59

Model Evaluation

✓ R - Resubstitution

✓ Cross-validation

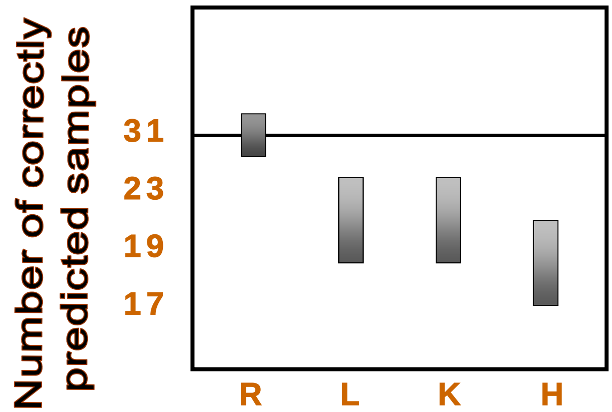
  - L - Leave-one-out and
  - K- 10-fold

✓ H - Hold-out

X TTV- Training-Testing-Validation

60

Results



Model Efficiency

- Degree of response to the Missing data
- Machine Learning generated rules = 22
- NeuroFuzzy model's correct prediction = 18
- Degree of response = 0.82
- Approximation in qualitative model

Summary

- Successful BP based NeuroFuzzy Model
- Cross-validation gives level of confidence
- Proper Modeling > Better Performance > Better data quality
- NeuroFuzzy captures ML rules
- Easy to handle missing data
- Validation for larger size of data necessary
- NeuroFuzzy – Good for qualitative data handling of Constructibility Analysis

63

The brain is a wonderful organ.
It starts working the moment
you get up in the morning and
does not stop until you get
into the Class.

64