UNIVERSITÀ DEGLI STUDI
DI MILANO

Fuzzy Logic Using Matlab

Enrique Muñoz Ballester

Dipartimento di Informatica
via Bramante 65, 26013 Crema (CR), Italy
enrique.munoz@unimi.it

Material

- Download slides data and scripts:

https://homes.di.unimi.it/munoz/teaching.html

Introduction

- Fuzzy concepts first introduced by Zadeh in the 1960s and 70s
- Traditional computational logic and set theory is all about
  - true or false
  - zero or one
  - in or out (in terms of set membership)
  - black or white (no grey)
- Fuzzy logic and sets are different

Basic concepts

- Approximation (“granulation”)
  - A color can be described precisely using RGB values, or it can be approximately described as “red”, “blue”, etc.
- Degree (“graduation”)
  - Two different colors may both be described as “red”, but one is considered to be more red than the other
- Fuzzy logic attempts to reflect the human way of thinking

Terminology

- Fuzzy set
  - A set X in which each element y has a grade of membership $\mu_X(y)$ in the range 0 to 1, i.e. set membership may be partial
  - e.g. if cold is a fuzzy set, exact temperature values might be mapped to the fuzzy set as follows:
    - 15 degrees → 0.2 (slightly cold)
    - 10 degrees → 0.5 (quite cold)
    - 0 degrees → 1 (totally cold)
- Fuzzy relation
  - Relationships can also be expressed on a scale of 0 to 1
  - e.g. degree of resemblance between two people

Terminology

- Fuzzy variable
  - Variable with (labels of) fuzzy sets as its values
- Linguistic variable
  - Fuzzy variable with values that are words or sentences in a language
  - e.g. variable color with values red, blue, yellow, green...
- Linguistic hedge
  - Term used as a modifier for basic terms in linguistic values
  - e.g. words such as very, a bit, rather, somewhat, etc.

Why fuzzy logic?

- Fuzzy logic is conceptually easy to understand.
  - The mathematical concepts behind fuzzy reasoning are very simple. Fuzzy logic is a more intuitive approach without the far-reaching complexity.
- Fuzzy logic is flexible
  - With any given system, it is easy to layer on more functionality without starting again from scratch.
- Fuzzy logic is tolerant of imprecise data
  - Everything is imprecise if you look closely enough, but more than that, most things are imprecise even on careful inspection. Fuzzy reasoning builds this understanding into the process rather than tacking it onto the end.
- Fuzzy logic can model nonlinear functions of arbitrary complexity
  - You can create a fuzzy system to match any set of input-output data. This process is made particularly easy by adaptive techniques like Adaptive Neuro-Fuzzy Inference Systems (ANFIS), which are available in Fuzzy Logic Toolbox software.

Why fuzzy logic

- Fuzzy logic can be built on top of the experience of experts.
  - In direct contrast to neural networks, which take training data and generate opaque, impenetrable models, fuzzy logic lets you rely on the experience of people who already understand your system.
- Fuzzy logic can be blended with conventional control techniques.
  - Fuzzy systems don't necessarily replace conventional control methods. In many cases fuzzy systems augment them and simplify their implementation.
- Fuzzy logic is based on natural language.
  - The basis for fuzzy logic is the basis for human communication. This observation underpins many of the other statements about fuzzy logic. Because fuzzy logic is built on the structures of qualitative description used in everyday language, fuzzy logic is easy to use.

Fuzzy logic applications

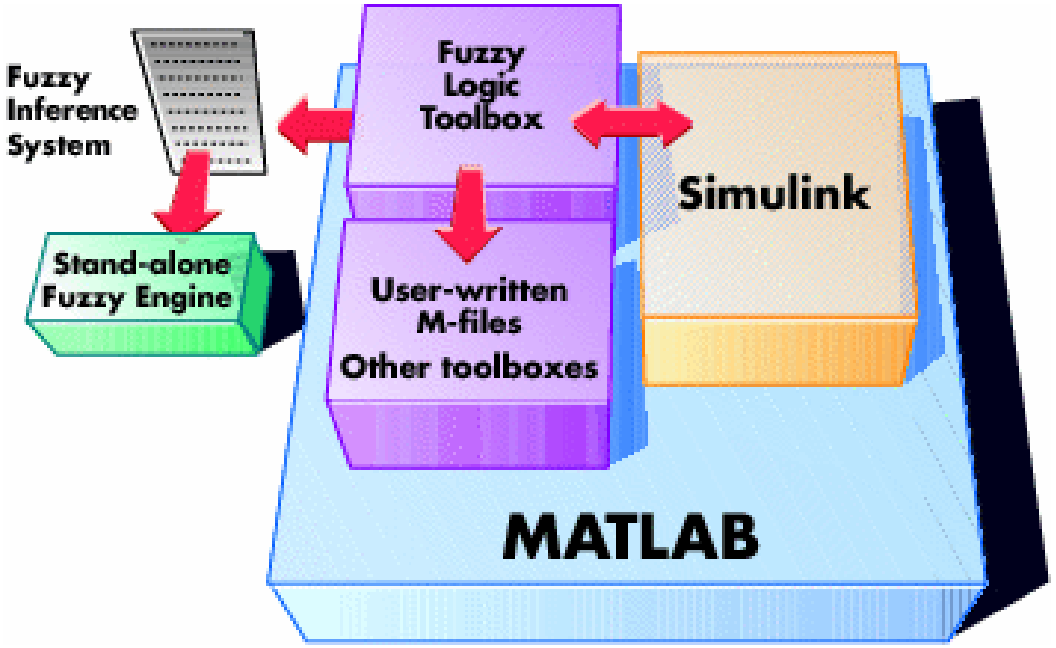
- Control Systems
  - Consumer systems
    - automatic transmissions
    - washing machines
    - camera autofocus
    - Temperature controllers
    - ABS
  - Industrial systems
    - Train driving
    - aircraft engines
    - power supply regulation
    - steam turbine start-up

Fuzzy logic applications

- Artificial Intelligence
  - Robotics
  - Image segmentation
  - Medical diagnosis systems
  - Pattern recognition

Fuzzy Logic Toolbox™ 1/2

- Impleamentation of Fuzzy Inference Systems
  - GUI
  - M-files
- Interaction with Simulink
- Standalone C



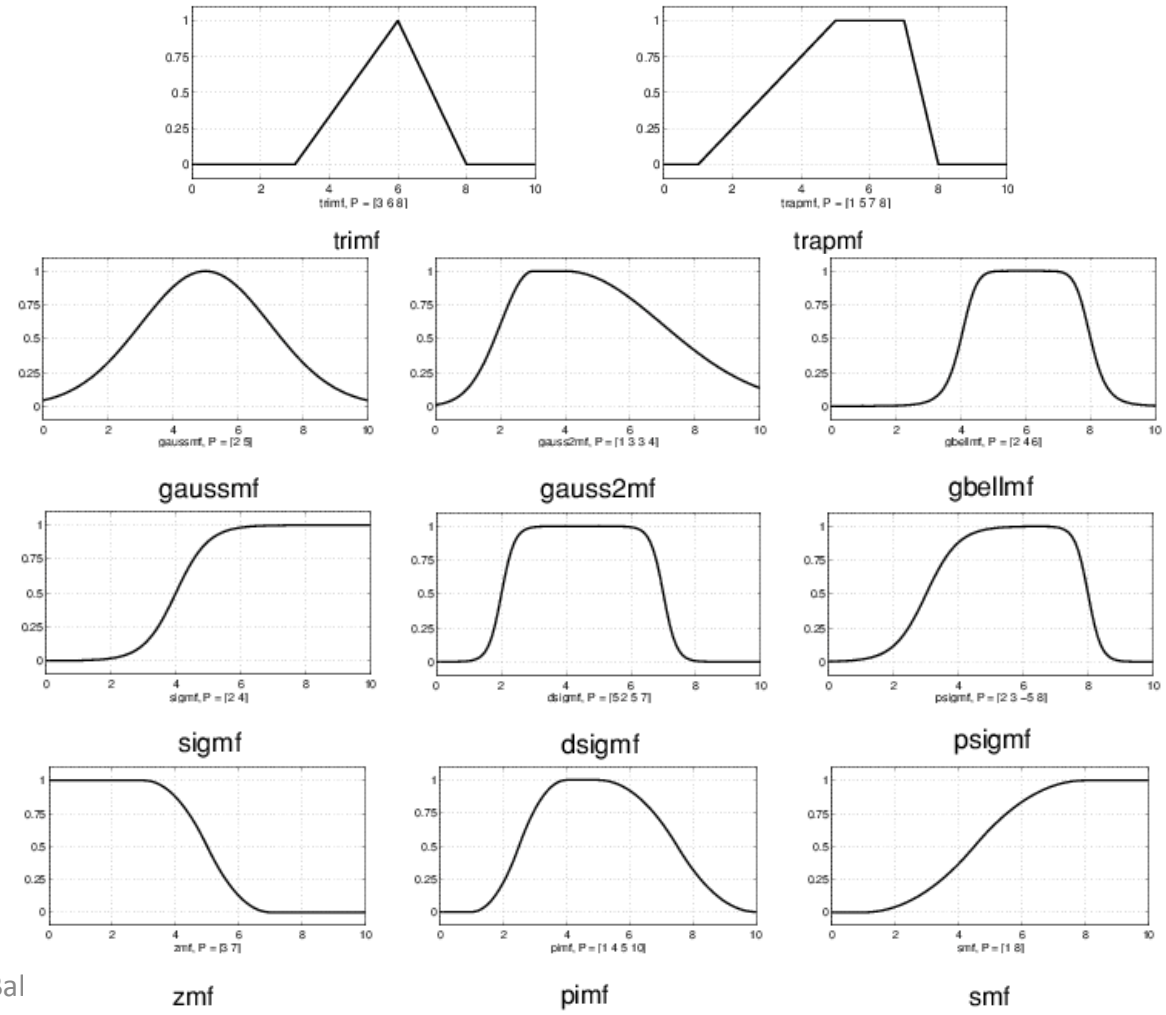
Fuzzy Logic Toolbox™ 2/2

- Important concepts:
  - Membership Functions
  - Logical Operations
  - If-Then Rules
  - Defuzzification Methods

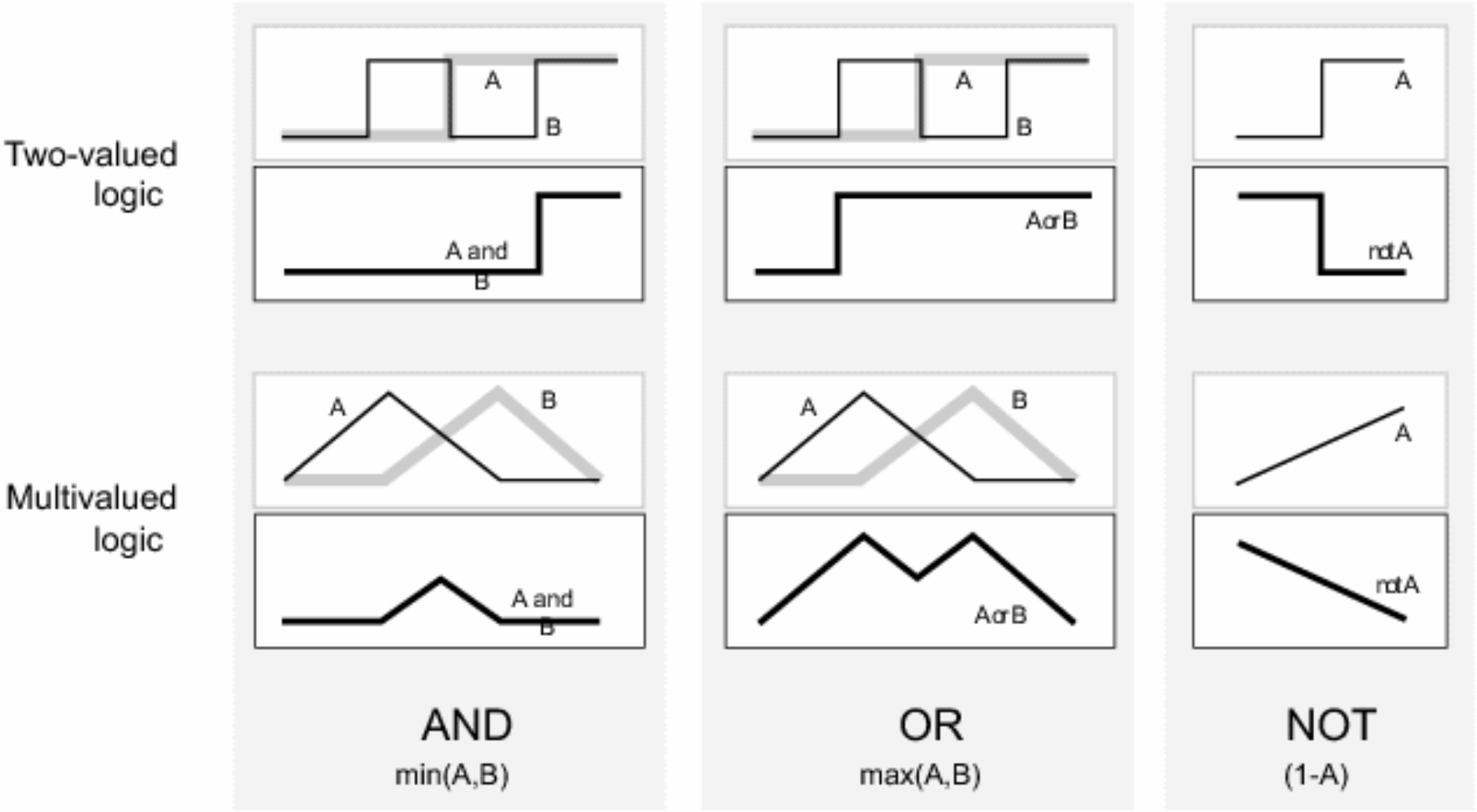
- ANFIS

Membership Functions

- 11 functions (**mfdemo**) + custom functions



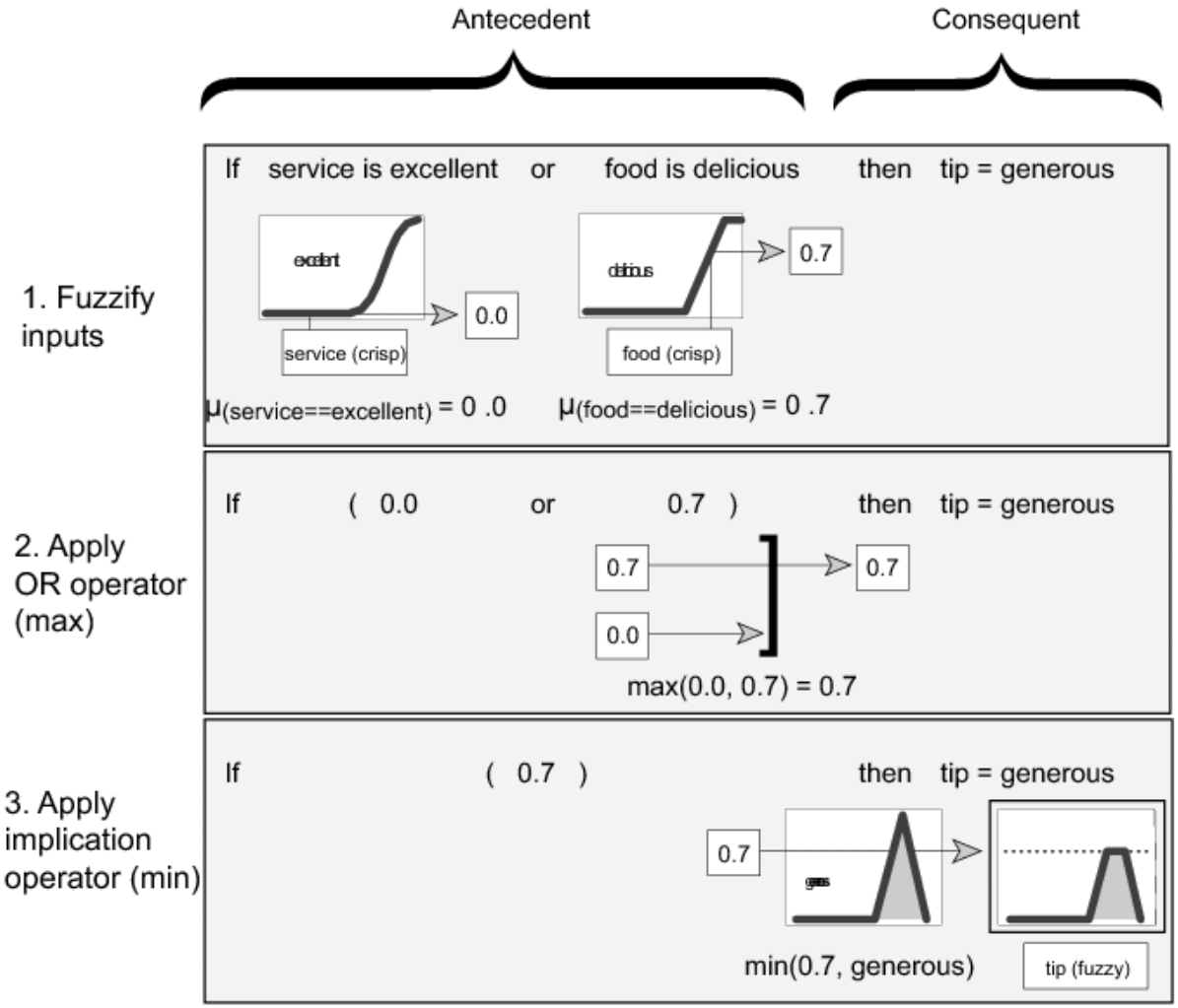
Logical Operations



If-Then Rules 1/3

- Interpreting if-then rules is a three-part process.
  - **Fuzzify inputs**: Resolve all fuzzy statements in the antecedent to a degree of membership between 0 and 1.
  - **Apply fuzzy operator to multiple part antecedents**: If there are multiple parts to the antecedent, apply fuzzy logic operators and resolve the antecedent to a single number between 0 and 1. This is the degree of support for the rule.
  - **Apply implication method**: Use the degree of support for the entire rule to shape the output fuzzy set. The consequent of a fuzzy rule assigns an entire fuzzy set to the output. This fuzzy set is represented by a membership function that is chosen to indicate the qualities of the consequent. If the antecedent is only partially true, (i.e., is assigned a value less than 1), then the output fuzzy set is truncated according to the implication method.

If-Then Rules 2/3

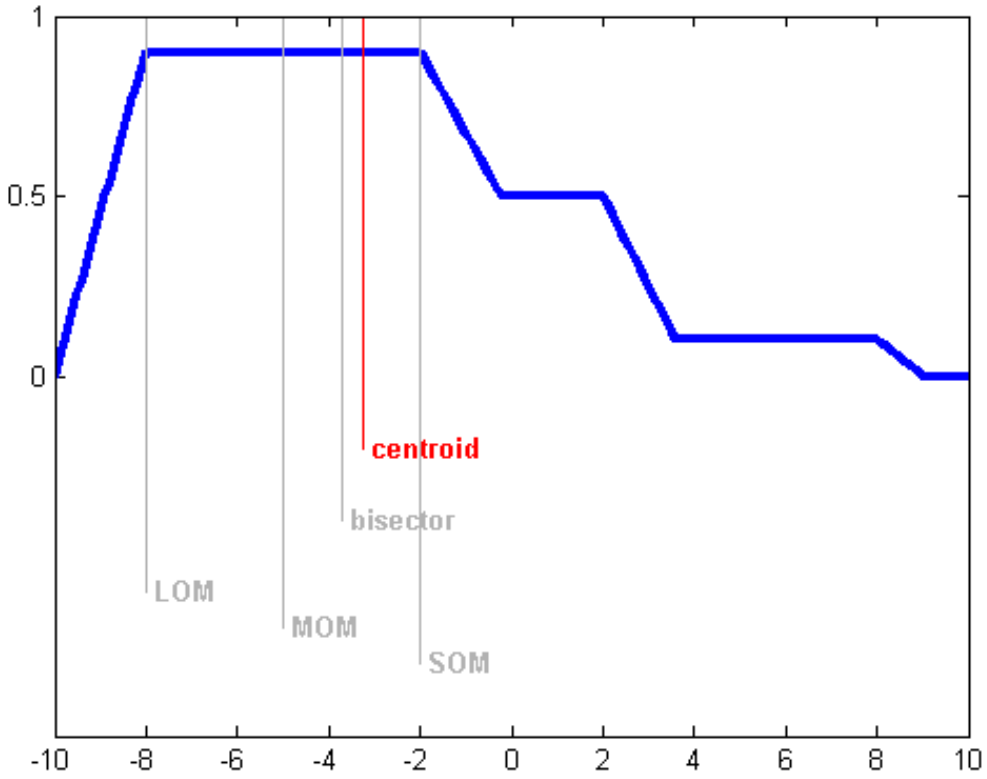


If-Then Rules 3/3

- Sintax
  - Verbose
    - If (service is poor) or (food is rancid) then (tip is cheap)
  - Symbolic
    - If (service == poor) | (food == rancid) => (tip = cheap)
  - Indexed
    - 1 1, 1

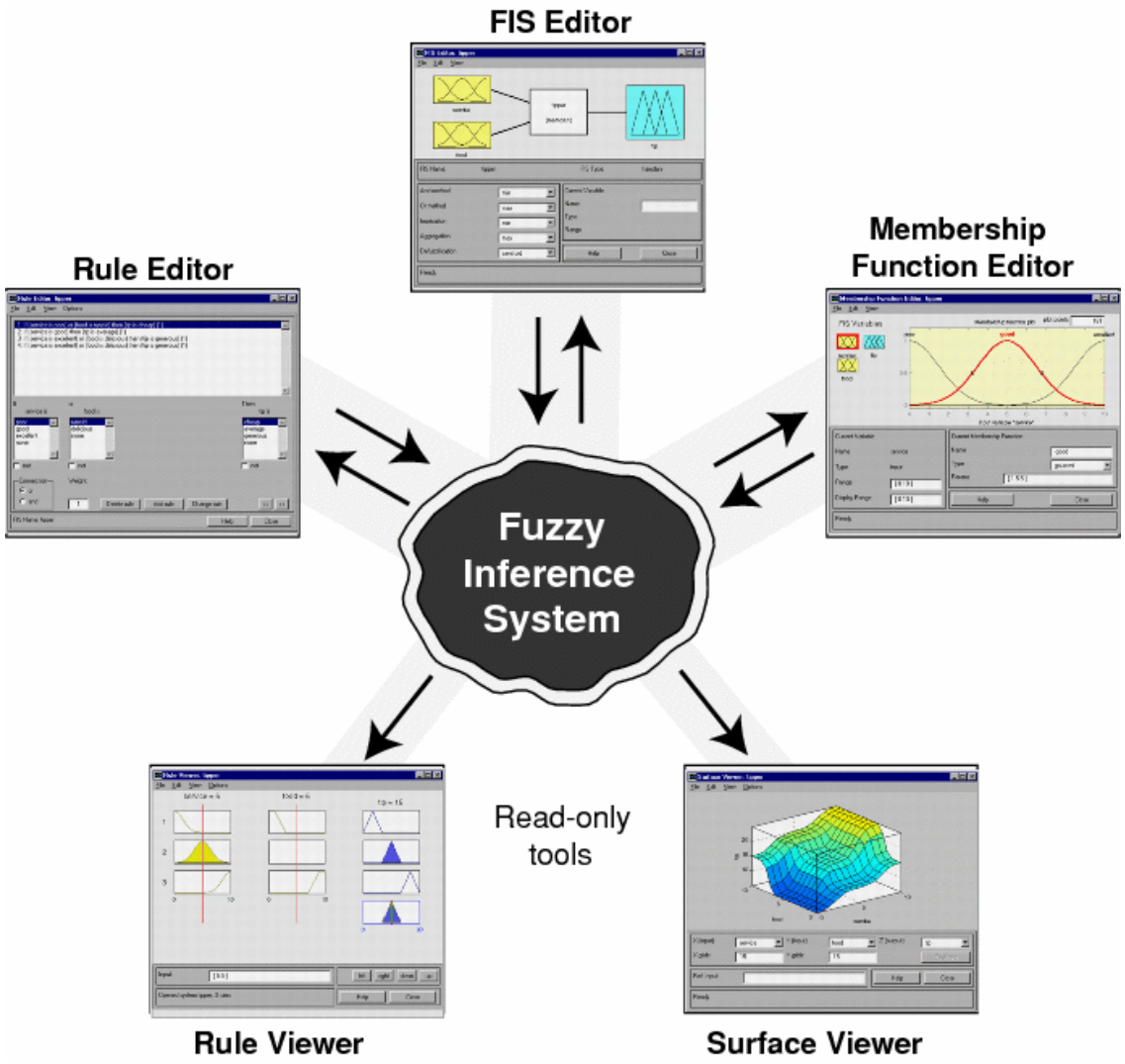
Defuzzification Methods

- 5 methods

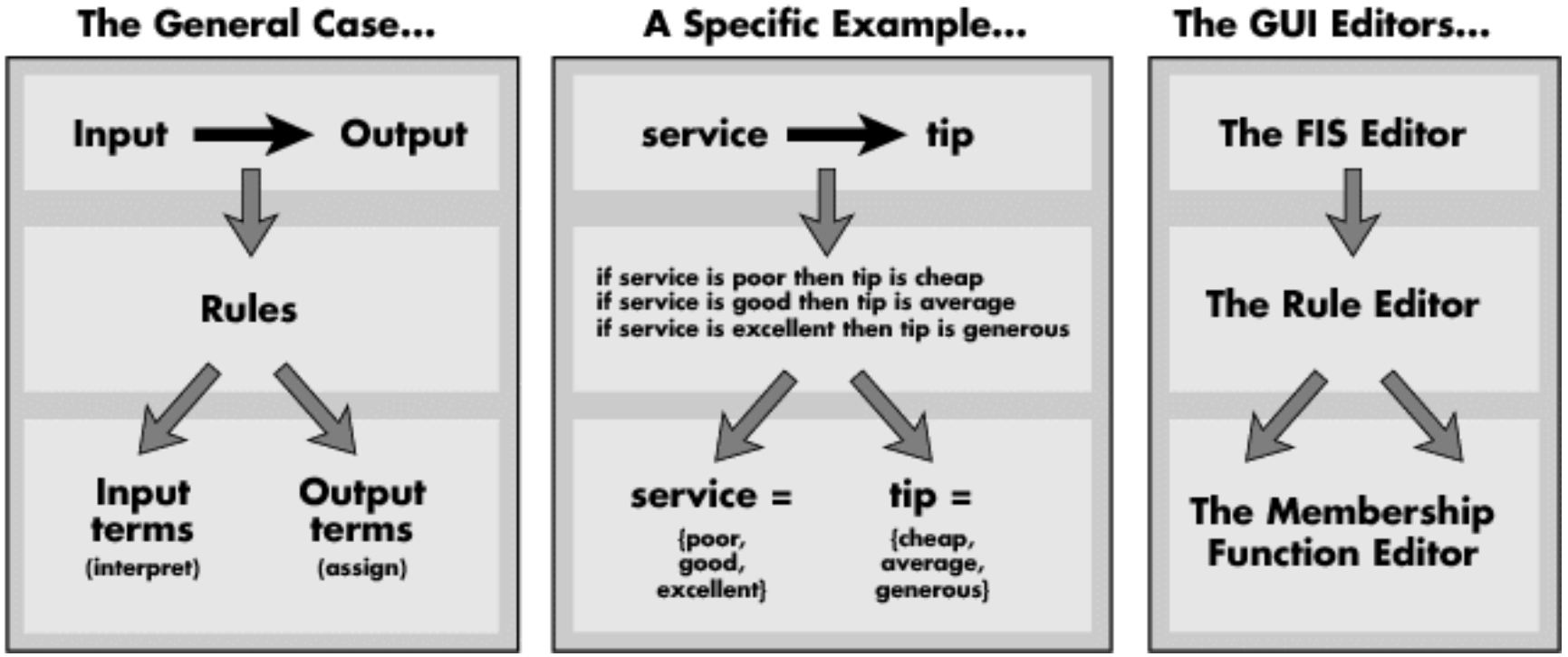


- Custom methods

Fuzzy Logic Toolbox™ GUI Tools

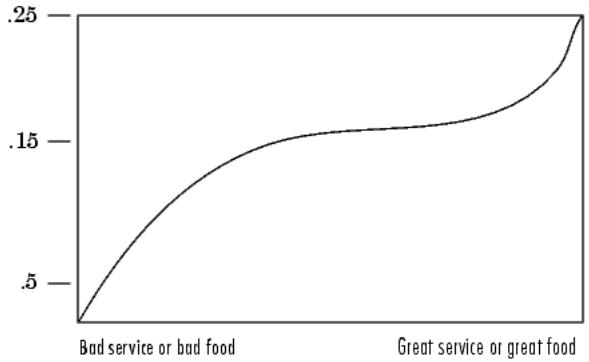


Implementation of Fuzzy Inference Systems



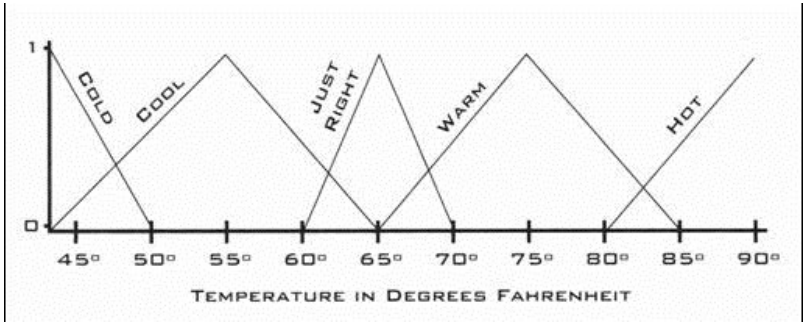
Example 1

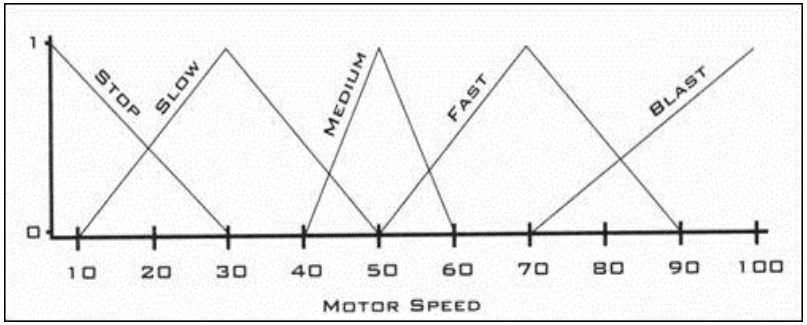
- Create an automatic tipper (Mamdani)
  - Inputs:
    - Quality of service: between 0 and 10 with fuzzy sets poor, good or excellent
    - Quality of food: between 0 and 10 with fuzzy sets rancid and delicious
  - Output: Percentage of tip between 5 and 30 with fuzzy sets cheap, average and generous
  - Rules:
    - If the service is poor or the food is rancid, then tip is cheap (5%)
    - If the service is good, then tip is average (15%)
    - If the service is excellent or the food is delicious, then tip is generous (25%)
- Function: fuzzyLogicDesigner or fuzzy



Exercise

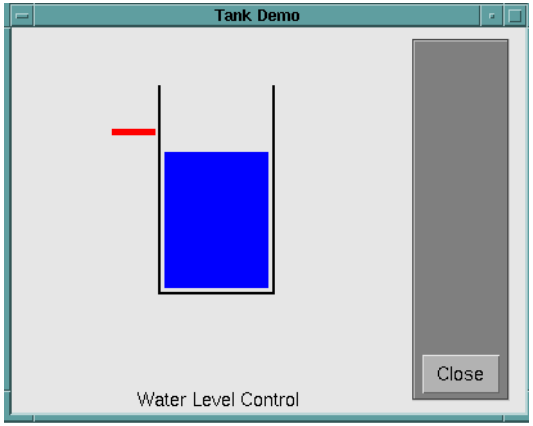
1. Design a motor speed controller for air conditioner
   - Input: Temperature (in Farenheit)
   - Output: Motor speed of the air conditioner
   - Rules:
     - If temperature is *cold* then motor speed is *stop*
     - If temperature is *cool* then motor speed is *slow*
     - If temperature is *just right* then motor speed is *medium*
     - If temperature is *warm* then motor speed is *fast*
     - If temperature is *hot* then motor speed is *blast*





Example 2

- Create a controller to regulate water level in a tank (using simulink)
  - A tank with a pipe flowing in and a pipe flowing out
  - Adjust the valve controlling the water that flows in
  - The outflow rate depends on the diameter of the outflow pipe (constant) and the pressure in the tank (varies with the water level)
- The controller
  - Knows current water level
  - Sets the valve aperture
  - Input: water level error
  - Output: rate at which the valve closes or opens
  - Rules:
    1. *If (level is okay) then (valve is no_change)*
    2. *If (level is low) then (valve is open_fast)*
    3. *If (level is high) then (valve is close_fast)*



Exercise

2. Correct the problems of Example 2 by adjusting the controller:
   - Use the input ‘rate’ (not used in the example), which describes the water level's rate of change
   - ‘rate’ is in the range [-0.1, 0.1], with negative values indicating a decrease in water level and positive values an increase
   - Create and tune two fuzzy sets to describe ‘rate’
   - Add two rules that use ‘rate’ to slow down the valve movement when we get close to the right level
   - Use the fuzzy sets of the output variable ‘valve’ close_slow and open_slow

Important functions

- Main GUI: **fuzzy** eventualFileName
- Instanciate the FIS: a = readfis('tipper.fis')
- Execute the FIS: evalfis([input1 input2 inputN], a)
- Show the FIS schema: showfis(a)
- Edit the source code: open tipper.fis
- GUI:
  - fuzzy(a) displays the FIS Editor
  - mfedit(a) displays the Membership Function Editor
  - ruleedit(a) displays the Rule Editor
  - ruleview(a) displays the Rule Viewer
  - surfview(a) displays the Surface Viewer
- Plots:
  - plotfis(a)
  - plotmf(a,'input',1)
  - gensurf(a)
  - showrule(a)

Textual mode

- Example

```
a=newfis('tipper');
a=addvar(a,'input','service',[0 10]);
a=addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a=addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a=addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
a=addvar(a,'input','food',[0 10]);
a=addmf(a,'input',2,'rancid','trapmf',[-2 0 1 3]);
a=addmf(a,'input',2,'delicious','trapmf',[7 9 10 12]);
a=addvar(a,'output','tip',[0 30]);
a=addmf(a,'output',1,'cheap','trimf',[0 5 10]);
a=addmf(a,'output',1,'average','trimf',[10 15 20]);
a=addmf(a,'output',1,'generous','trimf',[20 25 30]);
ruleList=[ ...
1 1 1 1 2
2 0 2 1 1
3 2 3 1 2 ];
a=addrule(a,ruleList);
```

Sugeno-Type Fuzzy Systems

- Takagi-Sugeno or simply Sugeno-type FIS has a different way of computing the consequence and defuzzification
- A general Sugeno rule has a form
  IF x_1 is A^k_1 AND x_2 is A^l_2 AND …AND x_n is A^p_n THEN $z_i = f_i(\cdot)$
- Here $z = f(\cdot)$ may be any function (even another mapping, like neural network, or another FIS)
- Usually $z_i = f_i(x_1; x_2; \ldots; x_n)$ is used. If this function is a first order polynomial, i.e.
  $z_i = a_n x_1 + a_{n-1} x_2 + \cdot \cdot \cdot + a_1 x_n + a_0$;
  the inference system is called a **first-order** Sugeno FIS.
- When f_i is a constant, the system is called a **zero-order** Sugeno FIS.

Sugeno-Type Fuzzy Inference 1/2

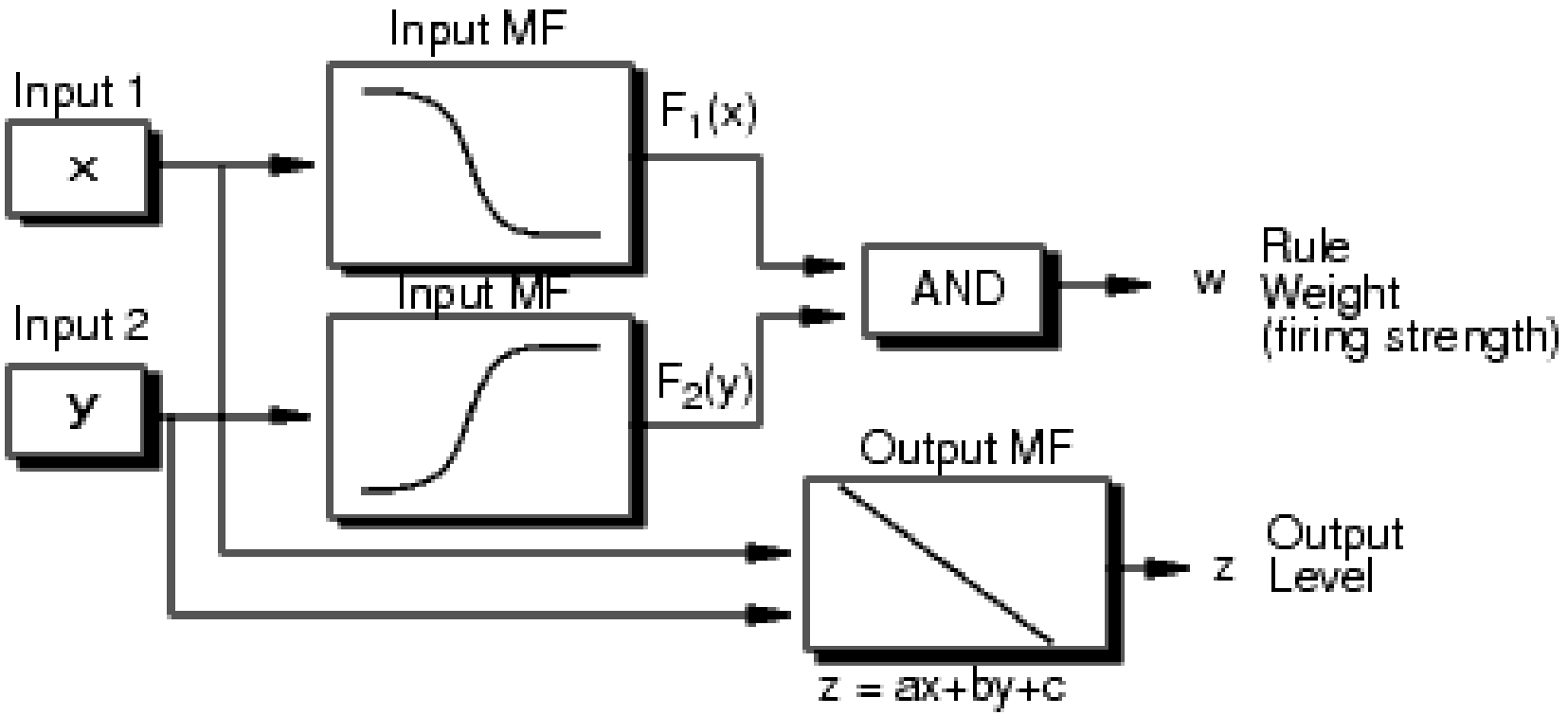
- The output level z_i of each rule is weighted by the firing strength w_i of the rule. For example, for an AND rule with Input 1 = x and Input 2 = y, the firing strength is

$$w_i = AndMethod(F_1(x), F_2(y))$$

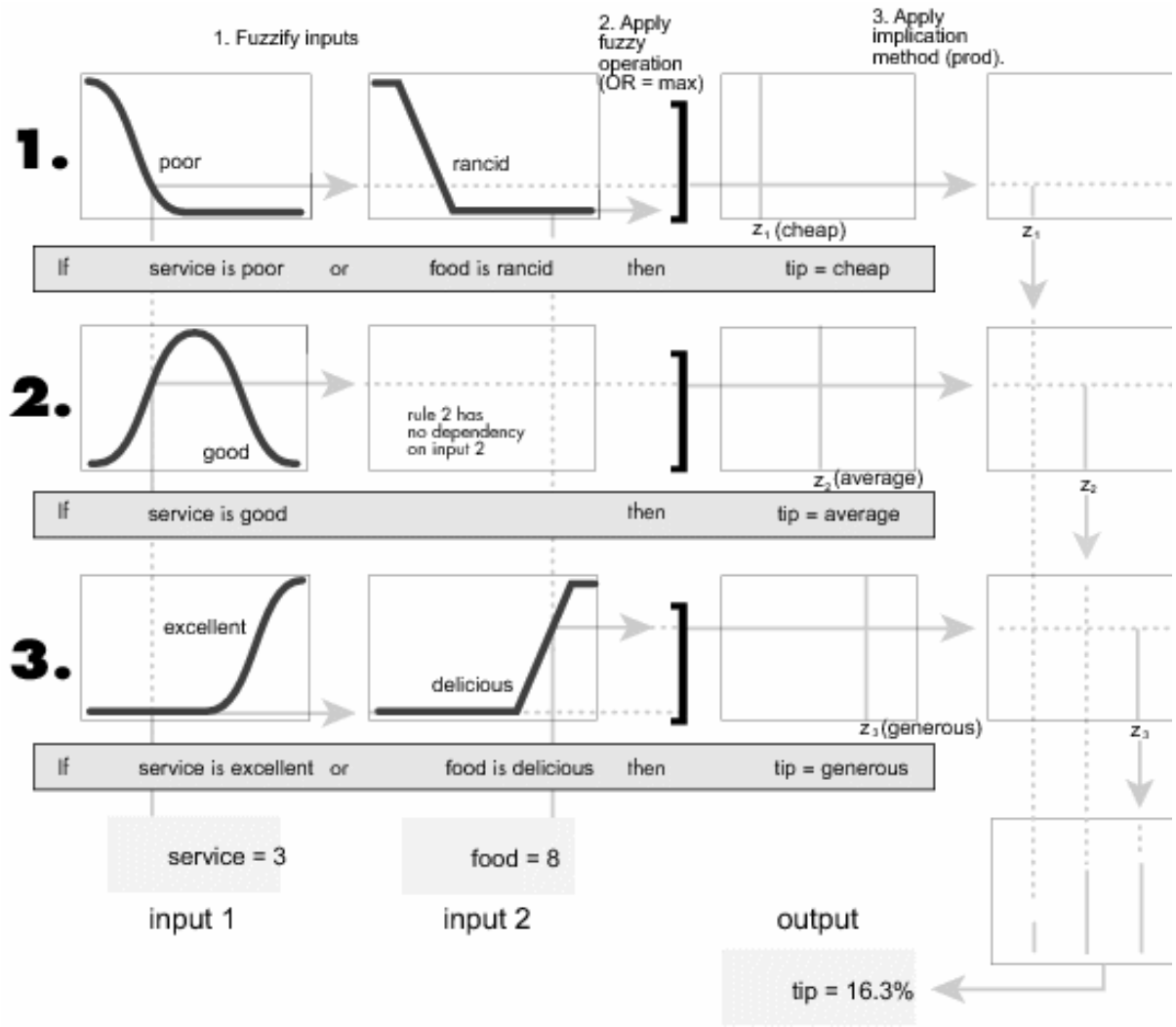
  where $F_{1,2}$ *(.)* are the membership functions for Inputs 1 and 2
- The final output of the system is the weighted average of all rule outputs, computed as

$$\text{Final Output} = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i}$$

Sugeno-Type Fuzzy Inference 2/2

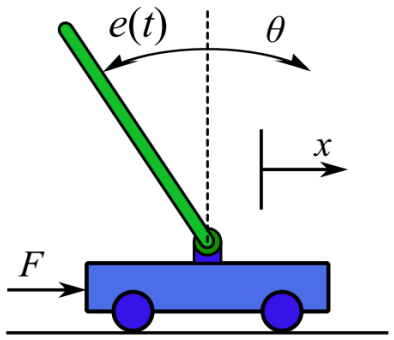


Sugeno-Type Fuzzy Inference: example



Example 3

- Create a controller for an inverted pendulum
  - A rigid pole hinged to a cart through a free joint with one degree of freedom
  - The cart can be moved to its right or left depending on the force exerted on it
  - Controller:
    - generates appropriate force on the cart such that we can move the cart to a desired position while keeping the pole balanced
    - Inputs:
      - Input 1: angular error
      - Input 2: angular error gain
      - Input 3: distance to desired position
      - Input 4: distance to desired position gain



Sugeno-Type Fuzzy Inference: advantages

- Can be more expressive than Mamdani systems
- Compact representation
- Computationally efficient
- Can be used with adaptive techniques, such as neuro-fuzzy systems

Adaptive Neuro-Fuzzy Inference System (ANFIS)

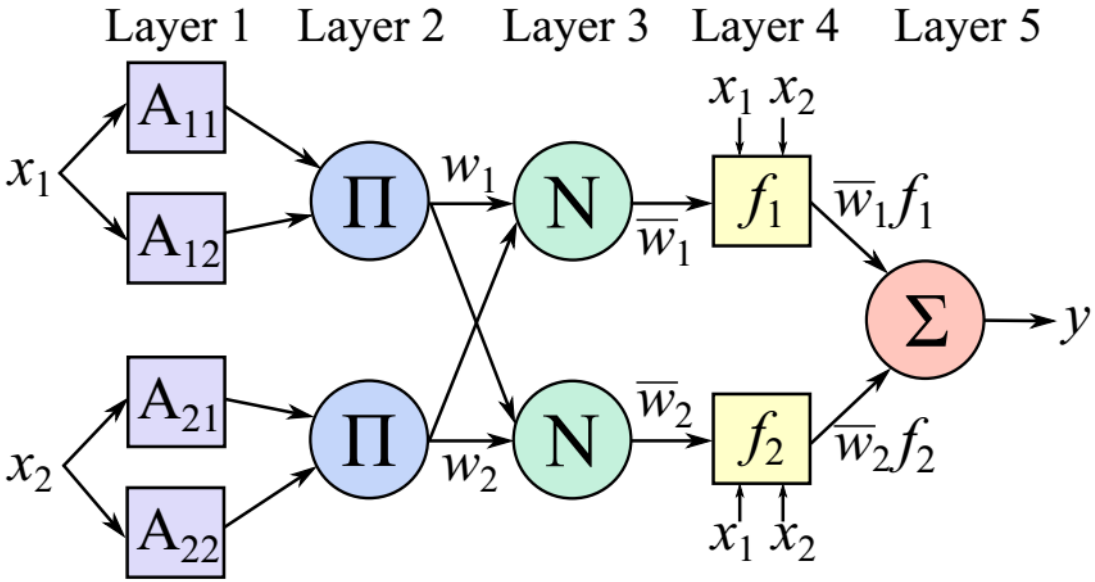
- Suppose you want to apply fuzzy inference to a system for which you already have a collection of input/output data that you would like to use for modeling, model-following, or some similar scenario. You do not necessarily have a predetermined model structure based on characteristics of variables in your system.

- In some modeling situations, you cannot discern what the membership functions should look like simply from looking at data. 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 in order to account for these types of variations in the data values. In such cases, you can use the Fuzzy Logic Toolbox neuro-adaptive learning techniques incorporated in the anfis command.

ANFIS

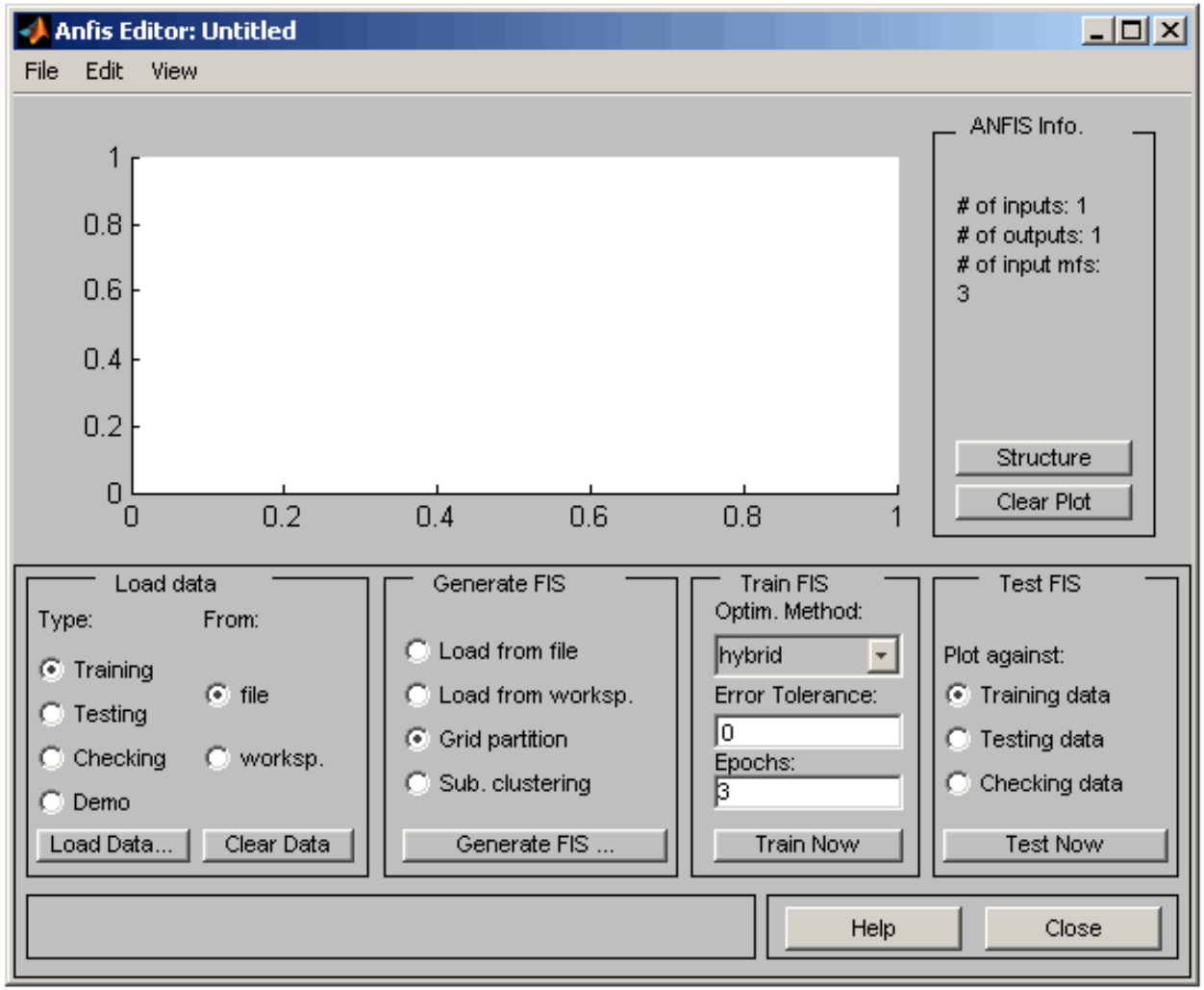
- Representation of the Sugeno FIS in a form of a feed-forward neural network

$$\text{Rule1}: \text{IF } x_1 \text{ is } A_1^1 \text{ AND } x_2 \text{ is } A_2^1 \text{ THEN } f_1 = p_{2,1}x_1 + p_{1,1}x_2 + p_{0,1}$$
$$\text{Rule2}: \text{IF } x_1 \text{ is } A_2^1 \text{ AND } x_2 \text{ is } A_2^2 \text{ THEN } f_2 = p_{2,2}x_1 + p_{1,2}x_2 + p_{0,2}$$



ANFIS GUI

- anfisedit



Example 4

- Approximate humps function with an ANFIS

  y = 1 ./ ((x-.3).^2 + .01) + 1 ./ ((x-.9).^2 + .04) - 6;

  – Generate a train set (200 random numbers in a range [0 to 2])
  – Generate a test set (200 random numbers in a range [0 to 2])
  – Optimize an ANFIS to approximate the function
  – Plot results

Exercises

3. Obtain a suitable ANFIS approximation of the signal

   x = rand(200,1)*5;

   y = erf(x);

4. Obtain a suitable ANFIS approximation of a sinusoidal signal

   x = rand(200,1)*0.05

   y = sin(100*pi*x - 2*pi*0.75)

5. Obtain a suitable ANFIS approximation of the signal

   x = rand(100,3)*6;

   $1+x(:,1)^{0.5}+x(:,2)^{-1}+x(:,3)^{-1.5}$

Exercises

6. System identification with ANFIS
   - Open folder invpen_identify (inside examples), set the MATLAB path to the folder, open invpen_sugeno.mdl (identical to the system from example 3, except for the To Workspace block)
   - Gather training data
     - Change the To Workspace block variable name to train_data
     - Choose a target position function type in the animation window and run the simulation for 20 steps
   - Train an appropriate ANFIS using the data generated
   - Save the trained FIS in a file with a different name, e.g. anfis.fis
   - Change the FIS name in the Fuzzy Logic Controller and perform the simulation with different target position functions
   - Repeat the training and test process with data obtained using the different target position functions (sine, square and sawtooth) and analyze the results
   - Repeat the training and test process with a dataset that will train the FIS for all (Sinusoid, Square, Saw) kinds of target position function types. And analyze the results

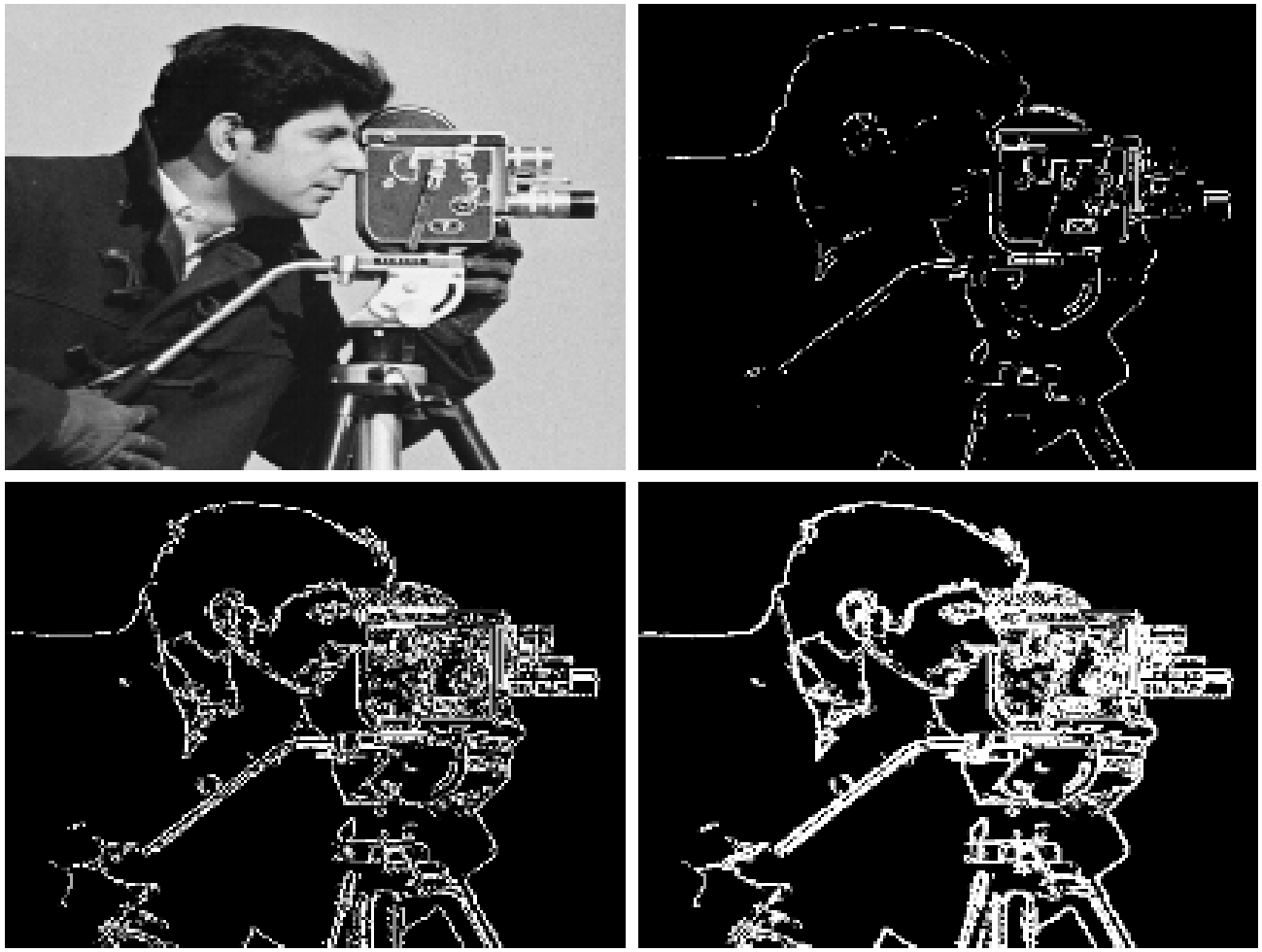
Fuzzy Robotics

- The Big Dog
  - http://www.youtube.com/watch?v=cHJJQ0zNNOM
- Football
  - http://www.youtube.com/watch?v=s5y-0SPdnVQ
- Rescue robot
  - http://www.youtube.com/watch?v=v2WRIuVI1W8
- Collaborative robots
  - http://www.youtube.com/watch?v=exEjV-Q7OKw

Fuzzy image processing

- Different applications:
  - fuzzy edge detection;
  - fuzzy contrast adjustment;
  - fuzzy image segmentation;
  - fuzzy image enhancement;
  - …
- An interesting approach consists in performing the analysis of every pixel with a FIS.

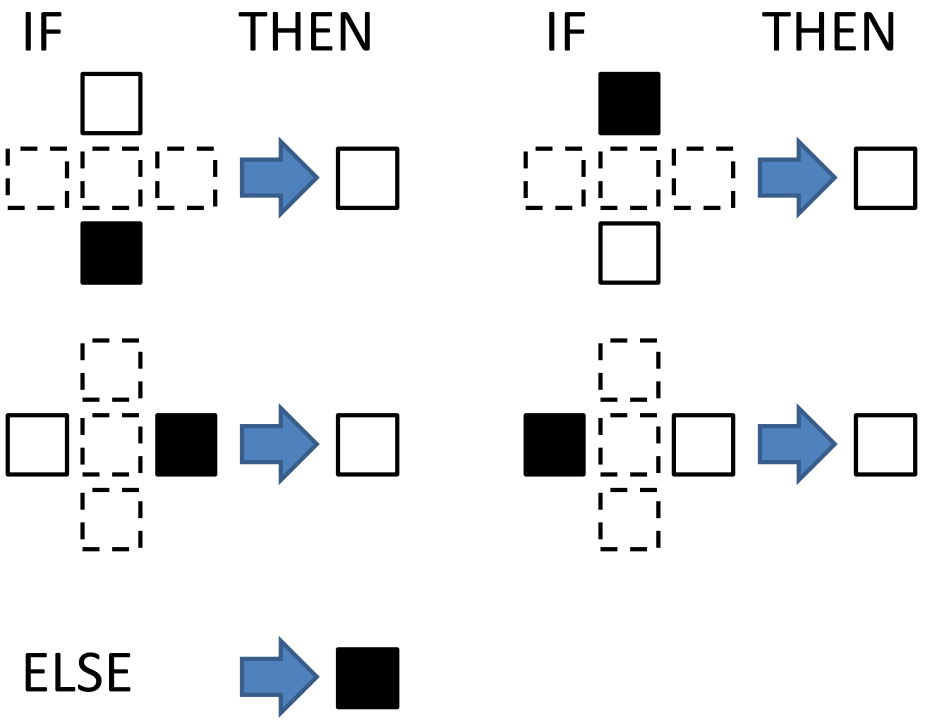
Edge detectors



Example 5
Fuzzy edge detector 4 pixel 1/2

Matlab
matrix(:)

1	4	7
2	5	8
3	6	9



Example 5
Fuzzy edge detector 4 pixel 2/2

- Membership functions should be designed according to the details that are more important in the considered applicative context.
- Evaluation of three membership functions.

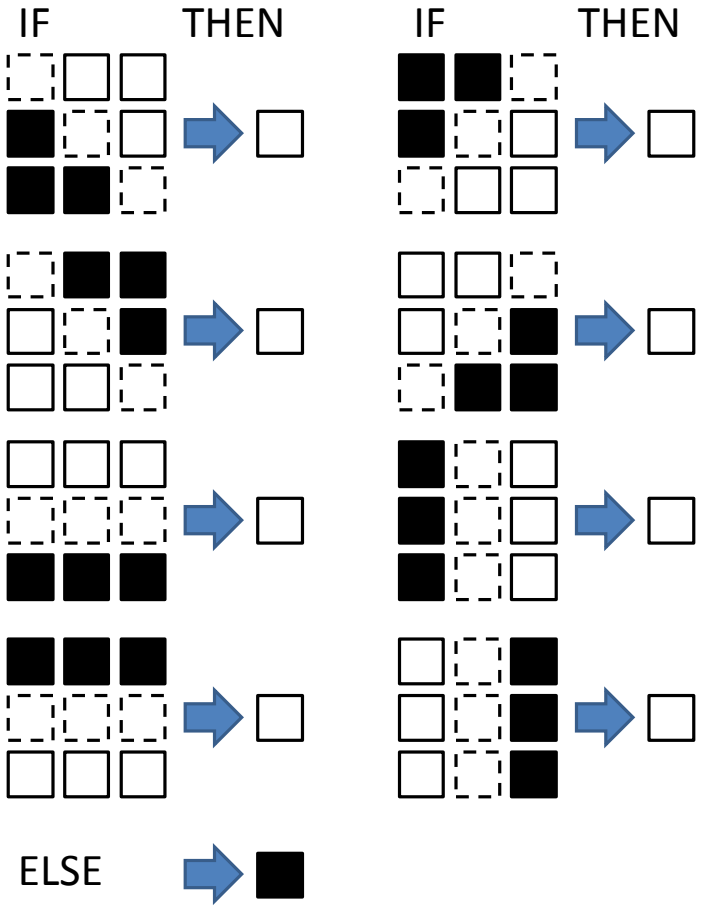
Exercise

- Create a fuzzy edge detector 8 pixel
  - Define membership functions for the 9 pixels of the neighborhood of current pixel
  - Define rules according to the following schema

Matlab

matrix(:)

1	4	7
2	5	8
3	6	9



Example 6
Fuzzy background removal 1/2

- If we interpret the image features as linguistic variables, then we can use fuzzy if-then rules to segment the image into different regions. A simple fuzzy segmentation rule may seem as follows:
  - **IF** the pixel is *dark*
    AND its neighbourhood is also *dark*
    AND *homogeneous*
    THEN it *belongs* to the background.

Example 6
Fuzzy background removal 2/2

- Three variables:
  - intensity = intensity of the pixel $I(i,j)$
  - meanI = mean intensity of the local region centered in $I(i,j)$
  - stdI = standard deviation of the local region centered in $I(i,j)$

- The parameters of the membership functions should be tuned according to the considered image

Suggested Lectures

- S. N. Sivanandam, S. Sumathi, S. N. Deepa, “Introduction to Fuzzy Logic using MatLab,” Springer, 2007
- Section “Fuzzy Logic Toolbox” of the Matlab Help
- http://a-lab.ee/edu/system/files/eduard.petlenkov/courses/ISS0023/2017_Autumn/materials/Sergei_Astapov_Fuzzy_Control_lecture_slides.pdf
- http://a-lab.ee/edu/system/files/eduard.petlenkov/courses/ISS0023/2017_Autumn/materials/Sergei_Astapov_Fuzzy_Control_Lab.pdf
- Passino, Kevin M., Stephen Yurkovich, and Michael Reinfrank. *Fuzzy control*. Vol. 20. Reading, MA: Addison-wesley, 1998.