Advanced Topics of Nature-Inspired Learning Algorithms

Dr H.K. Lam

Department of Engineering

King's College London

Office S2.14, Strand Building, Strand Campus

Email: hak-keung.lam@kcl.ac.uk

https://nms.kcl.ac.uk/hk.lam

Nature-Inspired Learning Algorithms (7CCSMBIM)

Outline

Learning Objectives

- To know the limits and advanced topics of GA.
- To extend the learning algorithms to multiple-objective optimisation problems, constrained optimisation problems and permutation problems.
- To get a basic concept of fuzzy logic system and appreciate how the system structure represents knowledge in linguistic terms and the idea how it is incorporated with learning algorithms.

Handling Expensive Cost Functions

Handling Expensive Cost Functions

- Complicated cost function $\Rightarrow$ time consuming for evaluation
- **Approaches:**
  - Identical chromosomes are not evaluated more than once.
  - Only non-identical chromosomes are allowed in the population.
  - Duplicated offspring is discarded (if searching time is less than evaluation time).
  - Does not allow identical chromosomes to mate.
  - Only evaluate the costs of offspring of mutated chromosomes.

Handling Expensive Cost Functions

Population with non-identical Chromosomes:

Example: 8 Chromosomes; 9 bits, 3 bits for each gene

First 3 bits are different

000101010
001101010
010101010
011101010
100101010
101101010
110101010
111101010

First bit of each gene is different

011000010
011000110
011100010
011100110
111000010
111000110
111100010
111100110

Multiple-Objective Optimisation

Multiple-Objective Optimisation

Definition: Given a set of cost functions, $f_i(\mathbf{x})$, $i = 1, \cdots, K$, where $\mathbf{x} \in \Re^n$ is a decision variable vector, find a vector $\mathbf{x}^*$ which minimises $\{f_1(\mathbf{x}^*), \cdots, f_K(\mathbf{x}^*)\}$.

- Objectives conflict each other.
- Optimising $\mathbf{x}$ with respect to one cost function will result in degrading others.
- It is not possible to have all cost functions being optimised.
- Find an acceptable solution or a feasible solution (when constraints are considered).

Example: Maximising the yield and minimising the costs (running costs, production costs, labour costs, production time, etc.) are conflicting to each other.

Weighted Sum Approach

- $f(\mathbf{x}) = \sum_{i=1}^{K} w_i f_i(\mathbf{x})$
- $w_i > 0$ and $\sum_{i=1}^{K} w_i = 1$

Advantages:

- Simple and straightforward.
- Single cost function.

Disadvantages:

- w_i has to be provided by the user.
- There is no rule to select w_i.

Note: All $f_i(\mathbf{x})$ must be of the same type of optimisation problem, say, all minimisation.

Gray Codes

Gray Codes

Problem: Representation of the variable values using ordinary binary number may lead to slow convergence of a GA.

$$parent_1 = \left[\cdots \underbrace{100 \uparrow 00000}_{gene} \cdots \right] = [\cdots 128 \cdots]$$

$$parent_2 = \left[\cdots \underbrace{011 \uparrow 11111}_{gene} \cdots \right] = [\cdots 127 \cdots]$$

Gray Codes

Problem: Representation of the variable values using ordinary binary number may lead to slow convergence of a GA.

$$parent_1 = \left[\cdots \underbrace{100 \uparrow 00000}_{gene} \cdots \right] = [\cdots 128 \cdots]$$

$$parent_2 = \left[\cdots \underbrace{011 \uparrow 11111}_{gene} \cdots \right] = [\cdots 127 \cdots]$$

After crossover:

$$offspring_1 = \left[\cdots \underbrace{10011111}_{gene} \cdots \right] = [\cdots 159 \cdots]$$

$$offspring_2 = \left[\cdots \underbrace{01100000}_{gene} \cdots \right] = [\cdots 96 \cdots]$$

- The offspring have diverging values from the parents.
- In some cases, two adjacent numbers may have many bit difference. For example, 3 is represented by 011 and 4 is represented by 100. All bits are different.
- The binary number encoding is nonlinear in number of bit difference, which introduce nonlinearity in the optimisation problem.

Gray Codes

A solution: Representation of the variable values using Gray code.

Property:

- Binary representations of consecutive numbers have a Hamming distance of one.

In words, the number of bit difference between two consecutive numbers is one.

- Hamming distance: the number of bits by which two binary numbers differ.

Example: The Hamming distance of 111111 and 010010 is 4.

Gray Codes

Consider the 127 and 128 are represented as 01000000 and 11000000, respectively (Hamming distance is 1).

$$parent_1 = \left[\cdots \underbrace{110 \uparrow 00000}_{gene} \cdots \right] = [\cdots 128 \cdots]$$

$$parent_2 = \left[\cdots \underbrace{010 \uparrow 00000}_{gene} \cdots \right] = [\cdots 127 \cdots]$$

Gray Codes

Consider the 127 and 128 are represented as 01000000 and 11000000, respectively (Hamming distance is 1).

$$parent_1 = \left[\cdots \underbrace{110 \uparrow 00000}_{gene} \cdots \right] = [\cdots 128 \cdots]$$

$$parent_2 = \left[\cdots \underbrace{010 \uparrow 00000}_{gene} \cdots \right] = [\cdots 127 \cdots]$$

After crossover:

$$offspring_1 = \left[\cdots \underbrace{11000000}_{gene} \cdots \right] = [\cdots 128 \cdots]$$

$$offspring_2 = \left[\cdots \underbrace{01000000}_{gene} \cdots \right] = [\cdots 127 \cdots]$$

- Parents with good solutions are more likely to produce good solutions.

Benefits of using gray codes instead of binary codes

- The hamming distance between two adjacent numbers is one, which means that the number of bit difference between two adjacent numbers is one, which reduces the nonlinearity introduced by coding.
- In the crossover process, some adjacent numbers are represented by binary codes with large number of bit difference, e.g., 3 represented by 011 to 4 represented by 100 (3 bit difference). After crossover, the offspring are very different from their parents, e.g., the parents 0|11 and 1|00 produce offspring 000 representing 0 and 111 representing 7. The big jump in candidate solutions may affect the search result especially local search is needed (when all candidates are more all less the same quality near the end of the search).
- In the mutation process, usually a small number of bits will be flipped. When binary codes are used, in some cases, a large number of bit flips are required to change a binary number to its adjacent number, e.g., 3 represented by 011 to 4 represented by 100 (3 bit difference). This will affect the search process especially when local search is needed.

Binary Code to Gray Code

Binary code to Gray code conversion: Consider a binary code of n bits, i.e., $b_n b_{n-1} b_{n-2} \cdots b_2 b_1$.

$$\begin{aligned} g_n &= b_n \\ g_{n-1} &= b_n \oplus b_{n-1} \\ g_{n-2} &= b_{n-1} \oplus b_{n-2} \\ &\vdots \\ g_2 &= b_3 \oplus b_2 \\ g_1 &= b_2 \oplus b_1 \end{aligned}$$

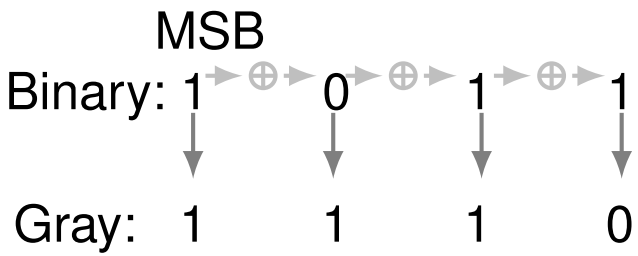
where $\oplus$ denotes the exclusive OR (XOR) operation; $A \oplus B = A\overline{B} + \overline{A}B$

$(0 \oplus 0 = 0; \quad 0 \oplus 1 = 1; \quad 1 \oplus 0 = 1; \quad 1 \oplus 1 = 0)$

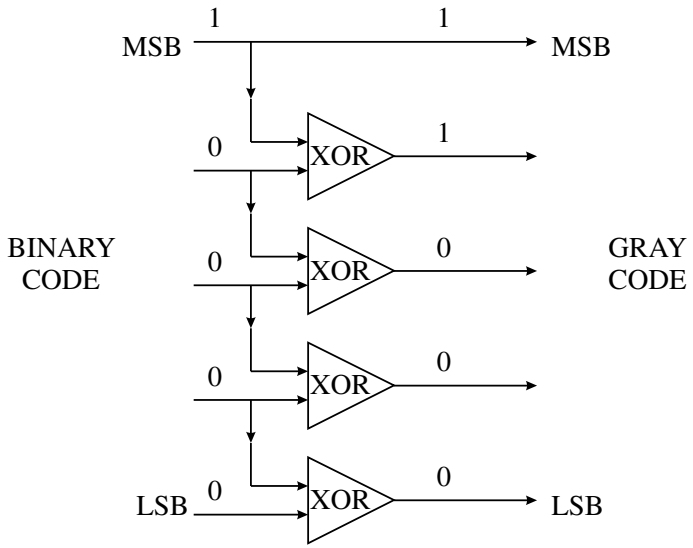
Gray code: $g_n g_{n-1} g_{n-2} \cdots g_2 g_1$

Binary Code to Gray Code

Example: Represent the 4-bit binary number 1011 in Gray code.



Example: Represent the 5-bit binary number 10000 in Gray code.



Gray Code to Binary Code

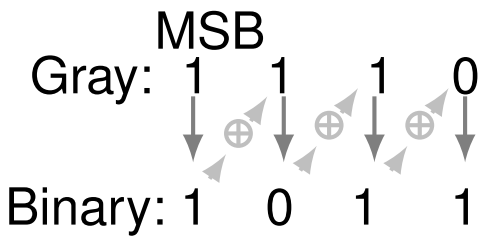
Gray code to binary code conversion: Consider a Gray code of n bits, i.e., $g_n g_{n-1} g_{n-2} \cdots g_2 g_1$.

$$
\begin{aligned}
b_n &= g_n \\
b_{n-1} &= b_n \oplus g_{n-1} \\
b_{n-2} &= b_{n-1} \oplus g_{n-2} \\
&\vdots \\
b_2 &= b_3 \oplus g_2 \\
b_1 &= b_2 \oplus g_1
\end{aligned}
$$

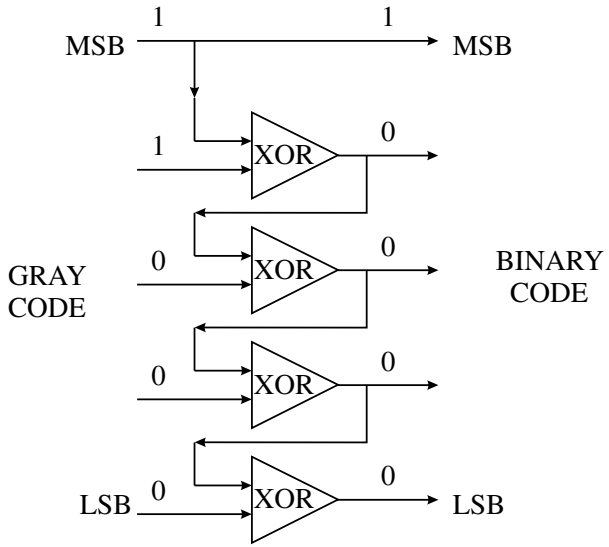
Binary code: $b_n b_{n-1} b_{n-2} \cdots b_2 b_1$

Gray Code to Binary Code

Example: Represent the 4-bit Gray code 1110 in binary code.



Example: Represent the 5-bit binary number 10000 in Gray code.



Permutation Problem

Permutation Problem

- **Permutation problem:** Optimisation problems involve sorting a list or putting things in the right order.
- The standard crossover and mutation operators are not appropriate.

Example: Reorder 6 numbers to minimise a function.

$parent_1 = [3\ 4 \uparrow 6\ 2\ 1\ 5]$

$parent_2 = [4\ 1 \uparrow 5\ 3\ 2\ 6]$

After single-point crossover

$offspring_1 = [3\ 4\ 5\ 3\ 2\ 6]$

$offspring_2 = [4\ 1\ 6\ 2\ 1\ 5]$

Permutation Problem

Four Methods:

1. Partially matched crossover (PMX)
2. Ordered crossover (OX)
3. Cycle crossover (CX)
4. Coding with reference list

Permutation Problem: Partially matched crossover (PMX)

Partially matched crossover (PMX) - Steps:

1. Select two crossover points.
2. Swap genes between parents according to the crossover points.
3. Swap duplicated genes between parents beyond the crossover points.

Partially matched crossover (PMX) - Example:

Step 1: Select two crossover points

$parent_1 = [3 \uparrow 4\ 6 \uparrow 2\ 1\ 5]$

$parent_2 = [4 \uparrow 1\ 5 \uparrow 3\ 2\ 6]$

Partially matched crossover (PMX) - Example:

Step 1: Select two crossover points

$parent_1 = [3 \uparrow 4\ 6 \uparrow 2\ 1\ 5]$

$parent_2 = [4 \uparrow 1\ 5 \uparrow 3\ 2\ 6]$

Step 2: Swap genes

$offspring_{1A} = [3\ 1\ 5\ 2\ \underline{1}\ \underline{5}]$

$offspring_{2A} = [\underline{4}\ 4\ 6\ 3\ 2\ \underline{6}]$

Partially matched crossover (PMX) - Example:

Step 1: Select two crossover points

$parent_1 = [3 \uparrow 4\ 6 \uparrow 2\ 1\ 5]$

$parent_2 = [4 \uparrow 1\ 5 \uparrow 3\ 2\ 6]$

Step 2: Swap genes

$offspring_{1A} = [3\ 1\ 5\ 2\ \underline{1}\ \underline{5}]$

$offspring_{2A} = [\underline{4}\ 4\ 6\ 3\ 2\ \underline{6}]$

Step 3: Swap Duplicates

$offspring_{1B} = [3\ 1\ 5\ 2\ \mathbf{4}\ \mathbf{6}]$

$offspring_{2B} = [\mathbf{1}\ 4\ 6\ 3\ 2\ \mathbf{5}]$

Permutation Problem: Ordered crossover (OX)

Ordered crossover (OX) - Steps:

1. Select two crossover points.
2. Swap genes between parents according to the crossover points.
3. Cross out the duplicated genes in the parent.
4. Copy the remaining genes from the parent to the offspring starting from the second crossover point.

Advantage: relative ordering is preserved (although the absolute position within the string is not).

Ordered crossover (OX) - Example:

Step 1: Select two crossover points

$parent_1 = [3\ 4 \uparrow 6\ 2 \uparrow 1\ 5]$

$parent_2 = [4\ 1 \uparrow 5\ 3 \uparrow 2\ 6]$

Ordered crossover (OX) - Example:

Step 1: Select two crossover points

$parent_1 = [3\ 4 \uparrow 6\ 2 \uparrow 1\ 5]$

$parent_2 = [4\ 1 \uparrow 5\ 3 \uparrow 2\ 6]$

Step 2: Swap genes

$offspring_{1A} = [3\ 4\ 5\ 3\ 1\ 5]$

$offspring_{2A} = [4\ 1\ 6\ 2\ 2\ 6]$

Ordered crossover (OX) - Example:

Step 1: Select two crossover points

$parent_1 = [3\ 4 \uparrow 6\ 2 \uparrow 1\ 5]$

$parent_2 = [4\ 1 \uparrow 5\ 3 \uparrow 2\ 6]$

Step 2: Swap genes

$offspring_{1A} = [3\ 4\ 5\ 3\ 1\ 5]$

$offspring_{2A} = [4\ 1\ 6\ 2\ 2\ 6]$

Step 3: Cross out duplicates

$offspring_{1B} = [?\ ?\ 5\ 3\ ?\ ?]$

$parent_1 = [X\ 4 \uparrow 6\ 2 \uparrow 1\ X]$

$offspring_{2B} = [?\ ?\ 6\ 2\ ?\ ?]$

$parent_2 = [4\ 1 \uparrow 5\ 3 \uparrow X\ X]$

Ordered crossover (OX) - Example:

Step 1: Select two crossover points

$parent_1 = [3\ 4 \uparrow 6\ 2 \uparrow 1\ 5]$

$parent_2 = [4\ 1 \uparrow 5\ 3 \uparrow 2\ 6]$

Step 2: Swap genes

$offspring_{1A} = [3\ 4\ 5\ 3\ 1\ 5]$

$offspring_{2A} = [4\ 1\ 6\ 2\ 2\ 6]$

Step 3: Cross out duplicates

$offspring_{1B} = [?\ ?\ 5\ 3\ ?\ ?]$

$parent_1 = [X\ 4 \uparrow 6\ 2 \uparrow 1\ X]$

$offspring_{2B} = [?\ ?\ 6\ 2\ ?\ ?]$

$parent_2 = [4\ 1 \uparrow 5\ 3 \uparrow X\ X]$

Step 4: Copy the remaining genes from the parent.

$offspring_{1C} = [6\ 2\ 5\ 3\ 1\ 4]$

$offspring_{2C} = [5\ 3\ 6\ 2\ 4\ 1]$

Permutation Problem: Cycle crossover (CX)

Cycle crossover (CX) - Steps:

1. Start from the leftmost genes of the chromosomes.
2. Swap genes between parents.
3. Move to the next duplicated gene and swap genes at that position.
4. Repeat Step 3 until all genes are unique.

Cycle crossover (CX) - Example:

Step 1: Start from the leftmost genes

$parent_1 = [\mathbf{3}\ 4\ 6\ 2\ 1\ 5]$

$parent_2 = [\mathbf{4}\ 1\ 5\ 3\ 2\ 6]$

Cycle crossover (CX) - Example:

Step 1: Start from the leftmost genes

$parent_1 = [\mathbf{3}\ 4\ 6\ 2\ 1\ 5]$

$parent_2 = [\mathbf{4}\ 1\ 5\ 3\ 2\ 6]$

Step 2: Swap genes

$offspring_{1A} = [4\ \underline{4}\ 6\ 2\ 1\ 5]$

$offspring_{2A} = [3\ 1\ 5\ 3\ 2\ 6]$

Cycle crossover (CX) - Example:

Step 1: Start from the leftmost genes

$parent_1 = [\mathbf{3}\ 4\ 6\ 2\ 1\ 5]$

$parent_2 = [\mathbf{4}\ 1\ 5\ 3\ 2\ 6]$

Step 2: Swap genes

$offspring_{1A} = [4\ \underline{4}\ 6\ 2\ 1\ 5]$

$offspring_{2A} = [3\ 1\ 5\ 3\ 2\ 6]$

Step 3a: Swap Duplicates

$offspring_{1B} = [4\ 1\ 6\ 2\ \underline{1}\ 5]$

$offspring_{2B} = [3\ 4\ 5\ 3\ 2\ 6]$

Cycle crossover (CX) - Example:

Step 1: Start from the leftmost genes

$parent_1 = [\mathbf{3}\ 4\ 6\ 2\ 1\ 5]$

$parent_2 = [\mathbf{4}\ 1\ 5\ 3\ 2\ 6]$

Step 2: Swap genes

$offspring_{1A} = [4\ \underline{4}\ 6\ 2\ 1\ 5]$

$offspring_{2A} = [3\ 1\ 5\ 3\ 2\ 6]$

Step 3a: Swap Duplicates

$offspring_{1B} = [4\ 1\ 6\ 2\ \underline{1}\ 5]$

$offspring_{2B} = [3\ 4\ 5\ 3\ 2\ 6]$

Step 3b: Swap Duplicates

$offspring_{1C} = [4\ 1\ 6\ \underline{2}\ 2\ 5]$

$offspring_{2C} = [3\ 4\ 5\ 3\ 1\ 6]$

Cycle crossover (CX) - Example:

Step 1: Start from the leftmost genes

$parent_1 = [\mathbf{3}\ 4\ 6\ 2\ 1\ 5]$

$parent_2 = [\mathbf{4}\ 1\ 5\ 3\ 2\ 6]$

Step 2: Swap genes

$offspring_{1A} = [4\ \underline{4}\ 6\ 2\ 1\ 5]$

$offspring_{2A} = [3\ 1\ 5\ 3\ 2\ 6]$

Step 3a: Swap Duplicates

$offspring_{1B} = [4\ 1\ 6\ 2\ \underline{1}\ 5]$

$offspring_{2B} = [3\ 4\ 5\ 3\ 2\ 6]$

Step 3b: Swap Duplicates

$offspring_{1C} = [4\ 1\ 6\ \underline{2}\ 2\ 5]$

$offspring_{2C} = [3\ 4\ 5\ 3\ 1\ 6]$

Step 3c: Swap Duplicates

$offspring_{1D} = [4\ 1\ 6\ 3\ 2\ 5]$

$offspring_{2D} = [3\ 4\ 5\ 2\ 1\ 6]$

Permutation Problem: Coding with reference list

Coding with reference list - Steps:

1. Start from the leftmost gene.
2. Code the gene with the index of the actual element in the reference list $\{1, \cdots, N\}$.
3. Remove the used element in the list.
4. Move to the next gene and repeat Step 2 based on the updated reference list.

- **Advantage:** Any crossover methods can be applied.
- **Disadvantage:** Coding is required $\Rightarrow$ Computational time increases.

Permutation Problem: Coding with reference list

Example (Coding):

Step 1: Start from the leftmost gene

Reference list $\{1,2,3,4,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

Permutation Problem: Coding with reference list

Example (Coding):

Step 1: Start from the leftmost gene

Reference list $\{1,2,3,4,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

Step 2: Coding

Reference list $\{1,2,③,4,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [\mathbf{3}\ *\ *\ *\ *\ *]$

Example (Coding):

Step 1: Start from the leftmost gene

Reference list $\{1,2,3,4,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

Step 2: Coding

Reference list $\{1,2,③,4,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [\mathbf{3}\ *\ *\ *\ *\ *]$

Step 3a: Update and code

Reference list $\{1,2,④,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [3\ \mathbf{3}\ *\ *\ *\ *]$

Permutation Problem: Coding with reference list

Example (Coding):

Step 1: Start from the leftmost gene

Reference list $\{1,2,3,4,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

Step 2: Coding

Reference list $\{1,2,③,4,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [\mathbf{3}\ *\ *\ *\ *\ *]$

Step 3a: Update and code

Reference list $\{1,2,④,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [3\ \mathbf{3}\ *\ *\ *\ *]$

Step 3b: Update and code

Reference list $\{1,2,5,⑥\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [3\ 3\ \mathbf{4}\ *\ *\ *]$

KING'S College LONDON
University of London

Example (Coding):

Step 1: Start from the leftmost gene

Reference list $\{1,2,3,4,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

Step 2: Coding

Reference list $\{1,2,③,4,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [\mathbf{3}\ *\ *\ *\ *\ *]$

Step 3a: Update and code

Reference list $\{1,2,④,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [3\ \mathbf{3}\ *\ *\ *\ *]$

Step 3b: Update and code

Reference list $\{1,2,5,⑥\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [3\ 3\ \mathbf{4}\ *\ *\ *]$

Step 3c: Update and code

Reference list $\{1,②,5\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [3\ 3\ 4\ \mathbf{2}\ *\ *]$

KING'S College LONDON
University of London

Example (Coding):

Step 1: Start from the leftmost gene

Reference list $\{1,2,3,4,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

Step 2: Coding

Reference list $\{1,2,③,4,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [\mathbf{3}\ *\ *\ *\ *\ *]$

Step 3a: Update and code

Reference list $\{1,2,④,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [3\ \mathbf{3}\ *\ *\ *\ *]$

Step 3b: Update and code

Reference list $\{1,2,5,⑥\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [3\ 3\ \mathbf{4}\ *\ *\ *]$

Step 3c: Update and code

Reference list $\{1,②,5\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [3\ 3\ 4\ \mathbf{2}\ *\ *]$

Step 3d: Update and code

Reference list $\{①,5\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ \mathbf{1}\ *]$

Permutation Problem: Coding with reference list

Example (Coding):

Step 1: Start from the leftmost gene

Reference list $\{1,2,3,4,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

Step 2: Coding

Reference list $\{1,2,③,4,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [\mathbf{3}\ *\ *\ *\ *\ *]$

Step 3a: Update and code

Reference list $\{1,2,④,5,6\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [3\ \mathbf{3}\ *\ *\ *\ *]$

Step 3b: Update and code

Reference list $\{1,2,5,⑥\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [3\ 3\ \mathbf{4}\ *\ *\ *]$

Step 3c: Update and code

Reference list $\{1,②,5\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [3\ 3\ 4\ \mathbf{2}\ *\ *]$

Step 3d: Update and code

Reference list $\{①,5\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ \mathbf{1}\ *]$

Step 3d: Update and code

Reference list $\{5\}$

$parent_1 = [3\ 4\ 6\ 2\ 1\ 5]$

$\boxed{coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ \mathbf{1}]}$

Permutation Problem: Coding with reference list

Example (Decoding):

Step 1: Start from the leftmost gene

Reference list $\{1,2,3,4,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

Example (Decoding):

Step 1: Start from the leftmost gene

Reference list $\{1,2,3,4,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

Step 2: Decode

Reference list $\{1,2,③,4,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [\mathbf{3}\ *\ *\ *\ *\ *]$

Example (Decoding):

Step 1: Start from the leftmost gene

Reference list $\{1,2,3,4,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

Step 2: Decode

Reference list $\{1,2,③,4,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [\mathbf{3}\ *\ *\ *\ *\ *]$

Step 3a: Update and decode

Reference list $\{1,2,④,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [3\ \mathbf{4}\ *\ *\ *\ *]$

Example (Decoding):

Step 1: Start from the leftmost gene

Reference list $\{1,2,3,4,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

Step 2: Decode

Reference list $\{1,2,③,4,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [\mathbf{3}\ *\ *\ *\ *\ *]$

Step 3a: Update and decode

Reference list $\{1,2,④,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [3\ \mathbf{4}\ *\ *\ *\ *]$

Step 3b: Update and decode

Reference list $\{1,2,5,⑥\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [3\ 4\ \mathbf{6}\ *\ *\ *]$

Example (Decoding):

Step 1: Start from the leftmost gene

Reference list $\{1,2,3,4,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

Step 2: Decode

Reference list $\{1,2,③,4,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [\mathbf{3}\ *\ *\ *\ *\ *]$

Step 3a: Update and decode

Reference list $\{1,2,④,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [3\ \mathbf{4}\ *\ *\ *\ *]$

Step 3b: Update and decode

Reference list $\{1,2,5,⑥\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [3\ 4\ \mathbf{6}\ *\ *\ *]$

Step 3c: Update and decode

Reference list $\{1,②,5\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [3\ 4\ 6\ \mathbf{2}\ *\ *]$

Example (Decoding):

Step 1: Start from the leftmost gene

Reference list $\{1,2,3,4,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

Step 2: Decode

Reference list $\{1,2,③,4,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [\mathbf{3}\ *\ *\ *\ *\ *]$

Step 3a: Update and decode

Reference list $\{1,2,④,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [3\ \mathbf{4}\ *\ *\ *\ *]$

Step 3b: Update and decode

Reference list $\{1,2,5,⑥\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [3\ 4\ \mathbf{6}\ *\ *\ *]$

Step 3c: Update and decode

Reference list $\{1,②,5\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [3\ 4\ 6\ \mathbf{2}\ *\ *]$

Step 3d: Update and decode

Reference list $\{①,5\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [3\ 4\ 6\ 2\ \mathbf{1}\ *]$

KING'S College LONDON
University of London

Example (Decoding):

Step 1: Start from the leftmost gene

Reference list $\{1,2,3,4,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

Step 2: Decode

Reference list $\{1,2,③,4,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [\mathbf{3}\ *\ *\ *\ *\ *]$

Step 3a: Update and decode

Reference list $\{1,2,④,5,6\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [3\ \mathbf{4}\ *\ *\ *\ *]$

Step 3b: Update and decode

Reference list $\{1,2,5,⑥\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [3\ 4\ \mathbf{6}\ *\ *\ *]$

Step 3c: Update and decode

Reference list $\{1,②,5\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [3\ 4\ 6\ \mathbf{2}\ *\ *]$

Step 3d: Update and decode

Reference list $\{①,5\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$Decoded\ parent_1 = [3\ 4\ 6\ 2\ \mathbf{1}\ *]$

Step 3d: Update and decode

Reference list $\{5\}$

$Coded\ parent_1 = [3\ 3\ 4\ 2\ 1\ 1]$

$\boxed{Decoded\ parent_1 = [3\ 4\ 6\ 2\ 1\ \mathbf{5}]}$

Permutation Problem: Mutation

Mutation:

a. **Inversion** – reverse the substring between two points.
Example: $[6 \uparrow 1\ 5\ 3\ 2 \uparrow 4]$, After mutation: $[6 \uparrow 2\ 3\ 5\ 1 \uparrow 4]$.

b. **Insertion and Displacement** – select a substring and insert it in a random place.
Example: $[6\ 1\ 5\ 3\ 2\ 4]$, After mutation: $[6\ 5\ 3\ 2\ 1\ 4]$

c. **Reciprocal exchange** – randomly choose two positions within the chromosome to swap. (Decoding is required when reference-list coding method is applied)
Example: $[6\ \mathbf{1}\ 5\ 3\ \mathbf{2}\ 4]$, After mutation: $[6\ \mathbf{2}\ 5\ 3\ \mathbf{1}\ 4]$

Penalty Functions

Penalty Function

- A method of handling constraints (equality and/or inequality constraints).

$$f_p(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{m} C_i(\mathbf{x})$$

- $f_p(\mathbf{x})$: penalised cost function,
- $f(\mathbf{x})$: unpenalised cost function,
- m: number of constraints,
- $C_i(\mathbf{x})$: a penalty function imposed for violation of constraints i.

Genetic Programming

Genetic Programming

- The (original) aim of genetic programming (GP) is to evolve executable computer programs (A GP is a computer program to write another computer program).
- The chromosome representation uses a **tree structure** (while GA uses **string**) with terminal set and function set.
- The terminal set specifies all variables and constants.
- The function sets contains all the functions that can be applied to the elements of the terminal set.
- The evolved program is executed to measure its performance within the problem domain to quantify the fitness of that program.

Genetic Programming

Tree-Based Representation:

- Each chromosome represents a program.
- **Adaptive Chromosome**: The chromosomes are of variable length structure in terms of size, shape and complexity.
  - **Size**: tree depth
  - **Shape**: branching factor of nodes in the tree
- **Domain-Specific Grammar:** A grammar needs to be defined to solve a specific problem.
  - **Terminal set** (variables and constants). *Elements of the terminal set form the leaf nodes of the evolved tree*.
  - **Function set** (all the functions applied to the elements of the terminal set, e.g., mathematical, arithmetic and/Boolean functions, even decision structures such as *if-then-else* and loops). *Element of the function set from the non-leaf nodes*.
  - **Semantic rules** to ensure the construction of semantically correct trees.

Example 1 (Boolean expression evolution - XOR):

(x_1 AND NOT x_2) OR (NOT x_1 AND x_2)

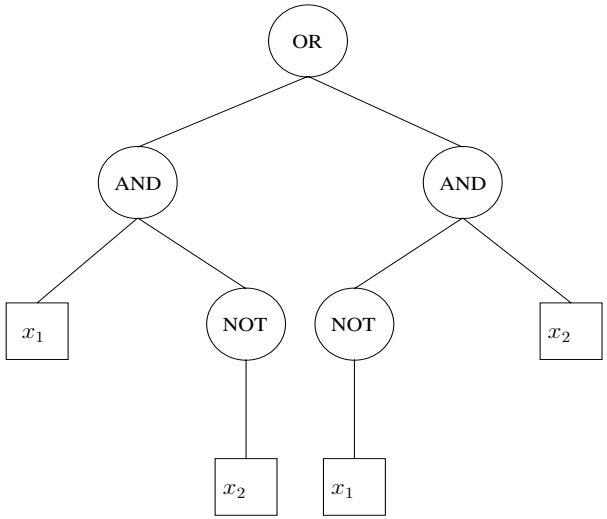


Figure 3: Tree representation of XOR.

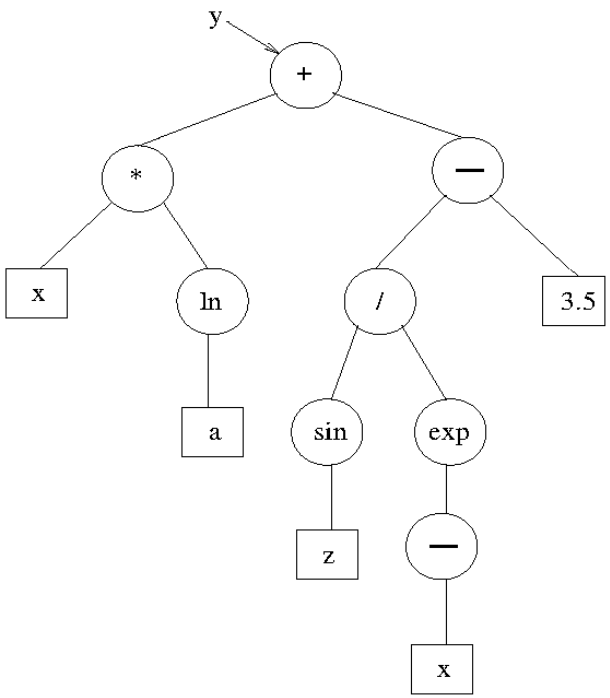
x_1	x_2	Target output
0	0	0
0	1	1
1	0	1
1	1	0

Table 1: XOR truth table

- Terminal set: { x_1, x_2 }
- Function set: { AND, OR, NOT }

Example 2 (mathematical expression):

$$y = x * \ln(a) + \sin(z) / \exp(-x) - 3.5$$



- Terminal set: { a, x, z, 3.5 }
- Function set: { —, +, *, /, sin, exp, ln }

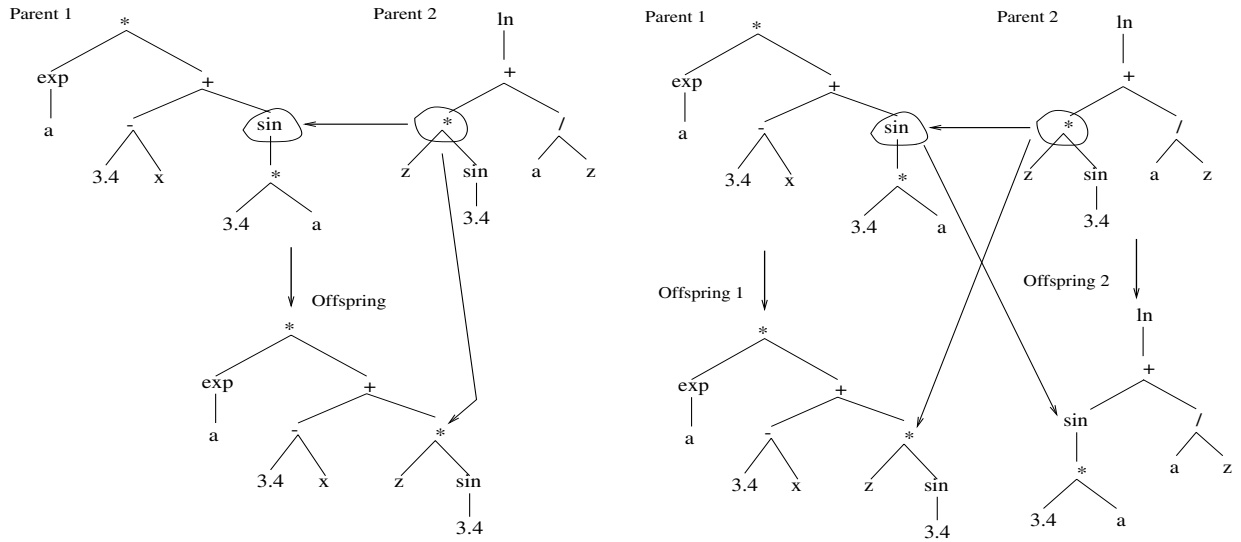
Figure 4: Tree representation of a mathematical expression.

Process

- **Initial population** (randomly generated)
- **Cost (fitness) function**: measure the performance of the chromosomes
- **Crossover**: exchange information between parents
- **Mutation**: explore new information

Crossover

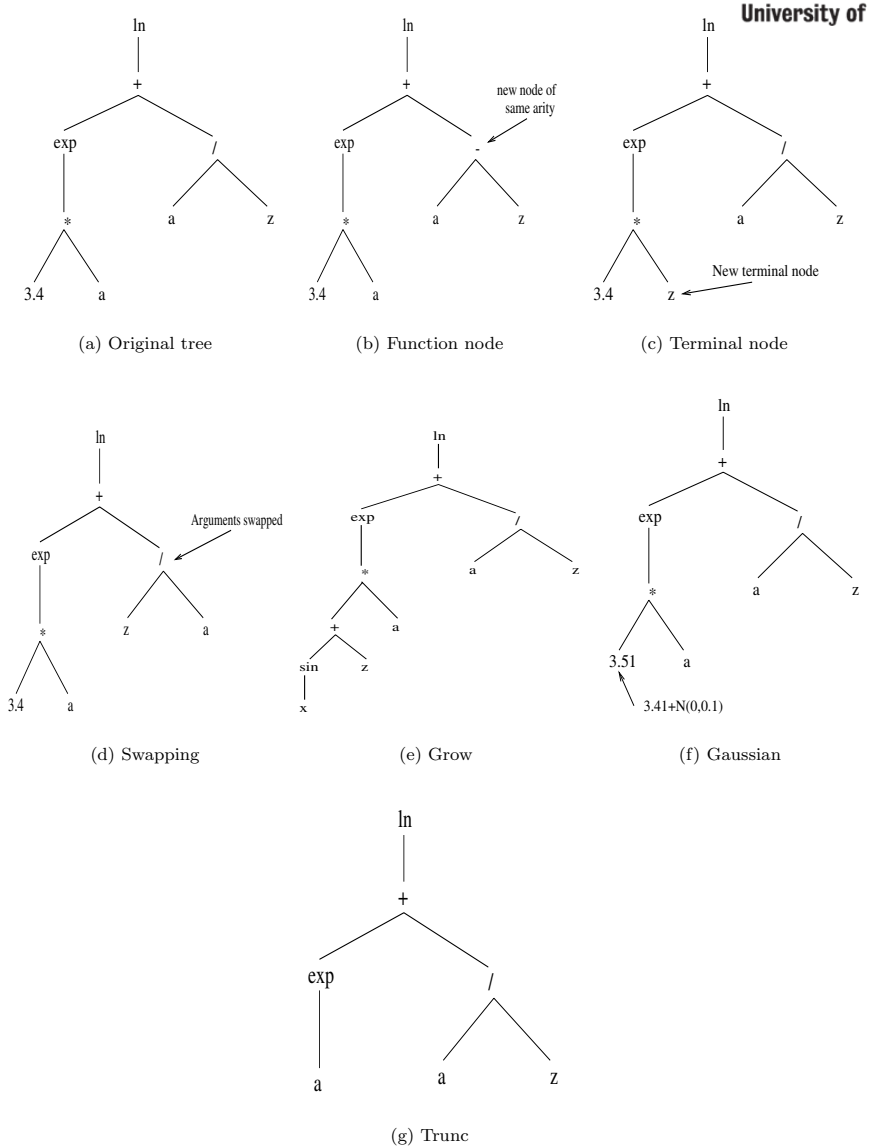
- **Generating one offspring**
  - Select a random node and replace the corresponding sub-tree of one parent by that of other.
- **Generating two offspring**
  - Select a random node and swap the sub-trees between parents



(a) One offspring (b) Two offspring

Mutation

- Function node mutation
- Terminal node mutation
- Swap mutation
- Grow mutation
- Gaussian mutation
- Trunc mutation



(a) Original tree (b) Function node (c) Terminal node

(d) Swapping (e) Grow (f) Gaussian

(g) Trunc

Figure 5: Different mutation methods.

Examples

Examples

Example 3 (coefficient identification): Identify the coefficients of $(8+1)\sin(5/4)+6\cos(3)+9-7/2=8.1009$, where the coefficients are unique integers in the range of 1 to 9.

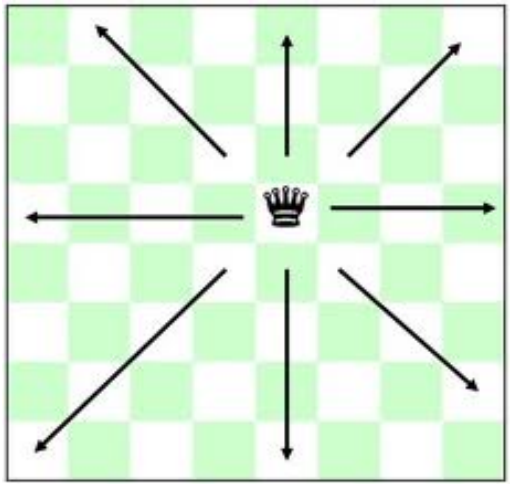
- Chromosome: $\mathbf{c}=[c_1,\cdots,c_9]$
- Cost function:

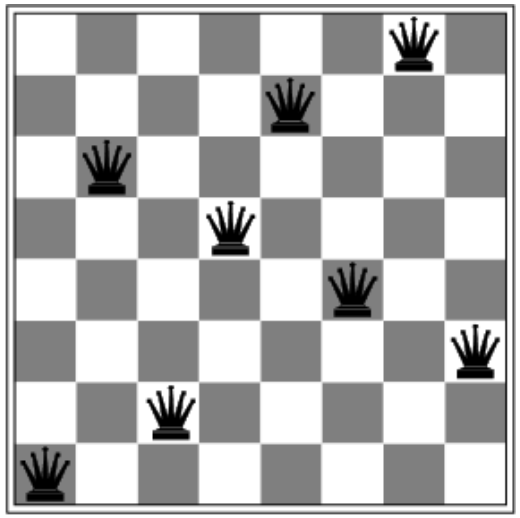
$$f(\mathbf{x})=|(c_1+c_2)\sin(c_3/c_4)+c_5\cos(c_6)+c_7-c_8/c_9-8.1009|.$$

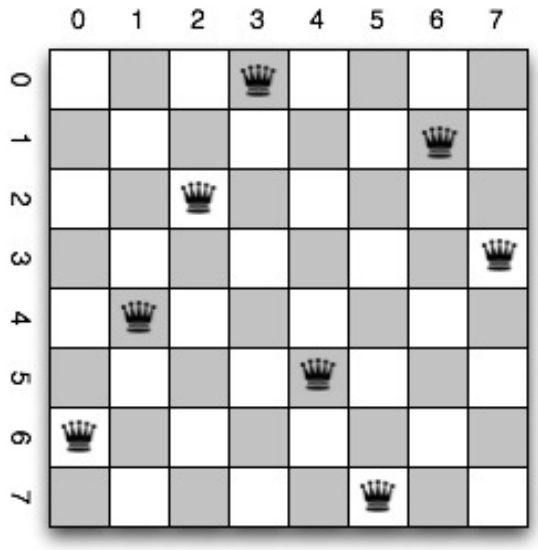
- Minimisation: $\min_{\mathbf{c}} f(\mathbf{c})$.
- $\mathbf{c}^*=[8,1,5,4,6,3,9,7,2]$; $\min_{\mathbf{c}^*} f(\mathbf{c}^*)=0$.

Examples

Example 4 (eight queens problem): Find a solution to the eight queens problem.

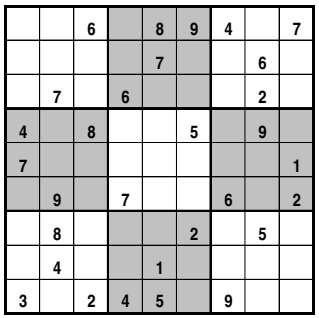




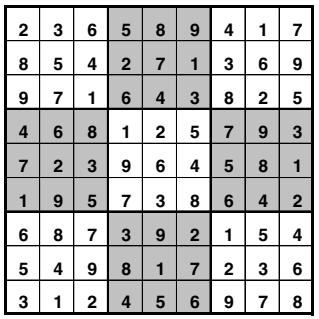


- 8 queens; 28 pairs
- Chromosome: $[x_1, \cdots, x_8]$, e.g., [7 2 6 3 1 4 0 5] (row index)
- Cost function: $f(\mathbf{x}) = 28 - h(\mathbf{x})$, $h(\mathbf{x})$: number of non-attacking pairs.
- Minimisation: $\min_{\mathbf{x}} f(\mathbf{x})$.

Example 5 (Sudoku puzzle): Solve Sudoku puzzle using GA.



		6		8	9	4		7
				7			6	
	7		6				2	
4		8			5		9	
7								1
	9		7			6		2
	8				2		5	
	4			1				
3		2	4	5		9		



2	3	6	5	8	9	4	1	7
8	5	4	2	7	1	3	6	9
9	7	1	6	4	3	8	2	5
4	6	8	1	2	5	7	9	3
7	2	3	9	6	4	5	8	1
1	9	5	7	3	8	6	4	2
6	8	7	3	9	2	1	5	4
5	4	9	8	1	7	2	3	6
3	1	2	4	5	6	9	7	8

- $x_{i,j}$: ij^{th} element of the Sudoku Puzzle;
- Chromosome: $[x_{1,1}, \cdots, x_{9,9}]$.
- Each row consists of the missing numbers in the range of 1 to 9.
- Cost function: $f(\mathbf{x}) = \sum_{i=1}^{9} |45 - \sum_{j=1}^{9} x_{i,j}| + \sum_{i=1}^{9} |9! - \prod_{j=1}^{9} x_{i,j}| + \sum_{j=1}^{9} |45 - \sum_{i=1}^{9} x_{i,j}| + \sum_{j=1}^{9} |9! - \prod_{i=1}^{9} x_{i,j}| + \sum_{i=1}^{3}\sum_{j=1}^{3} M_{i,j}$
- $M_{i,j}$: number of missing elements ($\{1, \cdots, 9\}$) in the ij^{th} block.
- Minimisation: $\min_{\mathbf{x}} f(\mathbf{x})$.

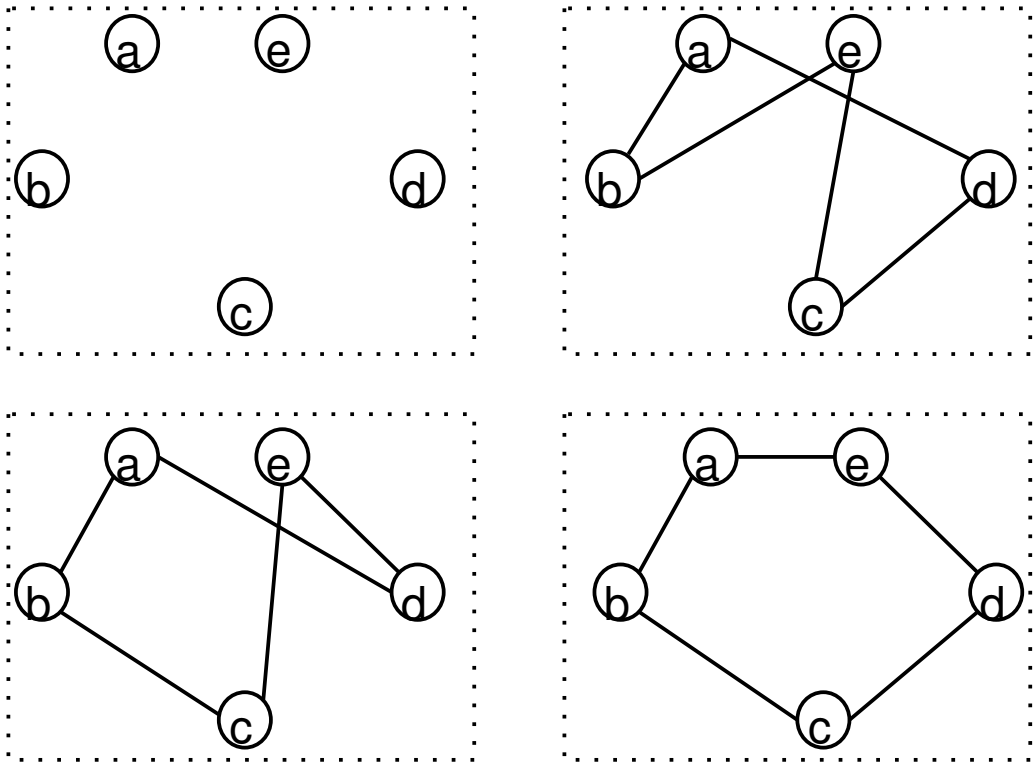
Examples

Example 6 (magic square): An $N \times N$ matrix with elements of 1 to N^2. Each element must be of a unique integer. Find the matrix that the sum of the i^{th} row, i^{th} column, the primary and secondary diagonals are all equal.

$$\begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \vdots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{bmatrix}$$

- Chromosome: $\mathbf{a} = [a_{11} \cdots a_{NN}]$
- $r_i = \sum_{j=1}^{N} a_{ij}; c_j = \sum_{i=1}^{N} a_{ij}$
- $d_1 = \sum_{i=1}^{N} a_{ii}; d_2 = \sum_{i=1}^{N} a_{i,N-i+1}$
- $s = \dfrac{\sum_{i=1}^{N} r_i + \sum_{j=1}^{N} c_j + d_1 + d_2}{2N+2}$
- $f(\mathbf{a}) = \sum_{i=1}^{N} |(r_i - s)| + \sum_{j=1}^{N} |(c_j - s)| + |d_1 - s| + |d_2 - s|$
- $\min_{\mathbf{a}} f(\mathbf{a})$

Example 7 (travelling salesman problem (TSP): Given a set of N cities, a salesman requires to find the shortest route visiting all cities once and returning to the starting city.

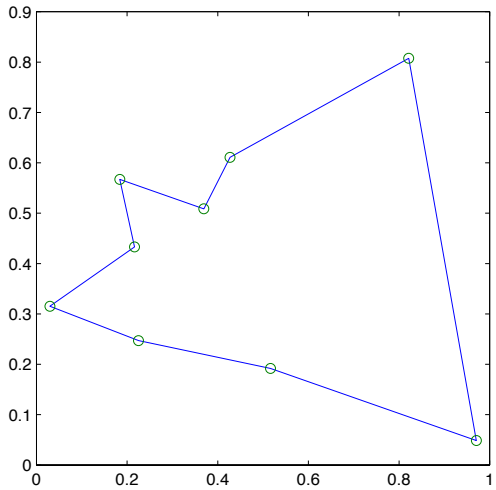


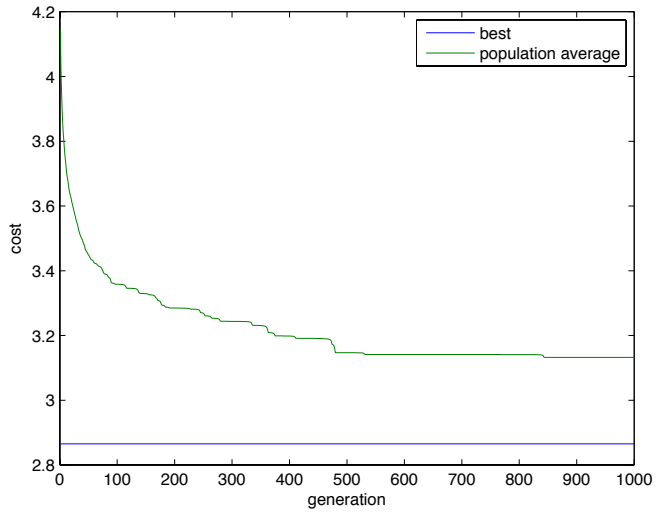
Example 7 (travelling salesman problem (TSP): Given a set of N cities, a salesman requires to find the shortest route visiting all cities once and returning to the starting city.

- City coordinates: (x_i, y_i), $i = 1, \cdots, N$
- Chromosome: $\mathbf{c} = [c_1, \cdots, c_N]$ (order of cities being visited)
- $f(\mathbf{c}) = \sum_{i=1}^{N-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} + \sqrt{(x_N - x_1)^2 + (y_N - y_1)^2}$
- Minimisation: $\min_{\mathbf{x}} f(\mathbf{x})$.

Examples

Example 7 (travelling salesman problem (TSP): Given a set of N cities, a salesman requires to find the shortest route visiting all cities once and returning to the starting city.





- 9 cities
- $N_{pop} = 10000$; $\mu = 0.2$; iterations: 1000; $X_{rate} = 50\%$
- $\mathbf{c}^* = [7\ 6\ 1\ 2\ 9\ 4\ 3\ 8\ 5]; f(\mathbf{c}^*) = 2.8654$

Examples

Example 8 (shipping application): Four items with different weights and values as shown in the following table are to form a shipment with a total weight of not more than 9 tons. Determine the number of each item maximizing the profit.

Item	Weight (Tons)	Net profit (£)
A	2	50
B	4	120
C	5	170
D	3	80

Table 2: Weights and values of Items

a. Formulate as minimisation problem by defining the cost functions and constraints.

b. Determine the penalised cost function for the minimisation problem.

c. Determine the representation of chromosome.

Examples

Let $\mathbf{x} = [x_1, x_2, x_3, x_4]$ be the number of items A, B, C and D, respectively.

a. $\min_{\mathbf{x}} -(50x_1 + 120x_2 + 170x_3 + 80x_4)$ subject to

- $2x_1 + 4x_2 + 5x_3 + 3x_4 \leq 9$

b. $f(\mathbf{x}) = -(50x_1 + 120x_2 + 170x_3 + 80x_4) + \lambda_1 c_1$

- $c_1 = \begin{cases} |2x_1 + 4x_2 + 5x_3 + 3x_4 - 9|, & \text{if } 2x_1 + 4x_2 + 5x_3 + 3x_4 > 9 \\ c_1 = 0, & \text{otherwise} \end{cases}$
- $\lambda_1 > 0$

c. Chromosome: $[x_1, x_2, x_3, x_4]$

Examples

Example 9 (roots): Given $x^3 - 11x^2 - 20x + 300 = 0$, find the three roots. r_1, r_2 and r_3 (within -10 and 10) using GA.

a. Formulate as minimisation problem by defining the cost functions and constraints.

b. Determine the penalised cost function for the minimisation problem.

c. Determine the representation of chromosome.

a. $\min_x |x^3 - 11x^2 - 20x + 300|$ subject to $x \neq r_1$ and $x \neq r_2$

b. $f(x) = |x^3 - 11x^2 - 20x + 300| + \lambda_1 \frac{1}{(x-r_1)^2} + \lambda_2 \frac{1}{(x-r_2)^2}$, $\lambda_1, \lambda_2 > 0$

c. Chromosome: $[x]$

Examples

Example 10 (Production): Two products X and Y are produced using two machines A and B where the processing times for production of each unit of X and Y are shown in the following table. Each product has to go through both machines for production. Given 1) 30 units of X and 90 units of Y are currently in stock, 2) the number of units of products X and Y must be at least 75 and 95, respectively, 3) available processing times on machines A and B are 40 hours and 35 hours, respectively, at the start of the current week, maximise the combined sum of units of X and Y in stock at the end of the week.

Product	Machine A	Machine B
X	50	30
Y	24	33

Table 3: Processing times in minutes

a. Formulate as minimisation problem by defining the cost function and constraints.
b. Determine the penalised cost function for the minimisation problem.
c. Determine the representation of chromosome.

Example 10 (Production):

Denote x and y (integers) as the number of units of X and Y, respectively.

a. $\min_{x,y} -(x+y)$ subject to

- $50x + 24y \leq 40 \times 60$
- $30x + 33y \leq 35 \times 60$
- $x \geq 75 - 30 = 45$
- $y \geq 95 - 90 = 5$

Example 10 (Production):

b. $f(x,y) = -(x+y) + \lambda_1 c_1 + \lambda_2 c_2 + \lambda_3 c_3 + \lambda_4 c_4$

subject to

- $c_1 = \begin{cases} |50x+24y-2400|, & \text{if } 50x+24y > 2400 \\ c_1 = 0, & \text{otherwise} \end{cases}$
- $c_2 = \begin{cases} |30x+33y-2100|, & \text{if } 30x+33y > 2100 \\ c_2 = 0, & \text{otherwise} \end{cases}$
- $c_3 = |x-45|$ if $x < 45$, otherwise $c_3 = 0$
- $c_4 = |y-5|$ if $y < 5$, otherwise $c_4 = 0$
- $\lambda_1, \lambda_2, \lambda_3, \lambda_4 > 0$

c. chromosome: $[x,y]$

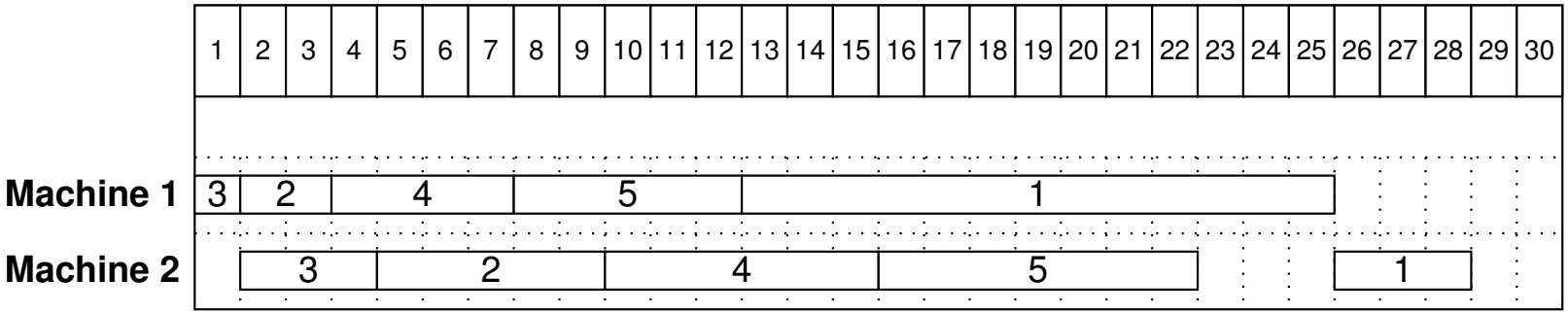
Examples

Example 11 (Job shop scheduling): We have 5 jobs denoted as job 1 to job 5. Each job must be processed by machine 1 first and then machine 2. The units of processing time for each job on each machine is given in the following table. Machines 1 and 2 are shared by all jobs. Find the job sequence such that makespan [1] is minimum.

Job	Machine 1	Machine 2
1	13	3
2	2	5
3	1	3
4	4	6
5	5	7

a. Formulate as minimisation problem by defining the cost function and representation of chromosome.

Example 11 (Job shop scheduling) cont'd:



a. Chromosome: $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5]$ where $x_i \in \{1,2,3,4,5\}$ is a unique job number. This is a permutation problem. $\mathbf{x}$ represents the job sequence, e.g., $\mathbf{x} = \{3,2,4,5,1\}$ as shown in the above figure.

Cost: $\sum_{i=1}^{5}(T(x_i) + I(x_i))$ where $T(x_i)$ is the time units required to process job x_i by machine 2 and $I(x_i)$ is the idle time unit machine 2 has to wait before processing job x_i.

In this example, cost = $\underbrace{\overbrace{3}^{T} + \overbrace{1}^{I}}_{\text{Job 3}} + \underbrace{\overbrace{5}^{T} + \overbrace{0}^{I}}_{\text{Job 2}} + \underbrace{\overbrace{6}^{T} + \overbrace{0}^{I}}_{\text{Job 4}} + \underbrace{\overbrace{7}^{T} + \overbrace{0}^{I}}_{\text{Job 5}} + \underbrace{\overbrace{3}^{T} + \overbrace{3}^{I}}_{\text{Job 1}} = 28$

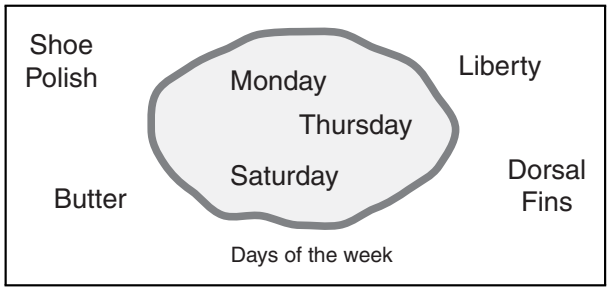
Introduction to Fuzzy Inference System

Fuzzy Logic:

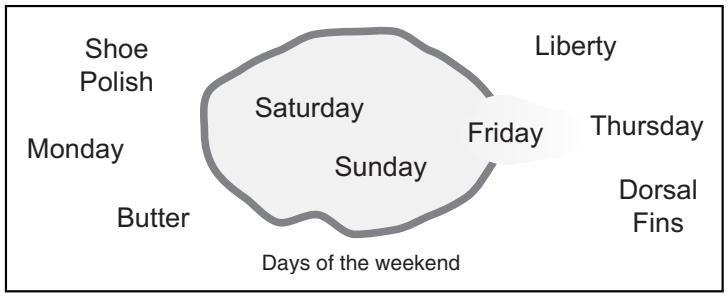
- Can a computer compute with words?
- Expert can solve a lot of complex problems using imprecision such as common sense and expert knowledge.
- Common sense and expert knowledge can be represented by linguistic rules, say, in *If-Then* format.
- Fuzzy logic is the theory of fuzzy sets, which is used to handle fuzziness/imprecision/ambiguity/vagueness.
- Fuzzy set theory which can mimic the human spirit for approximation reasoning based on imprecise information.
- By using fuzzy logic, "human spirit" can be computed/represented mathematically.
- Serves as a structure for represent knowledge and learning using natural inspired learning algorithms

Classical Sets and Fuzzy Sets:

- How do we represent imprecision and vagueness?
- How do you understand the phrase “Today is Weekend”?



(h) Classical set.

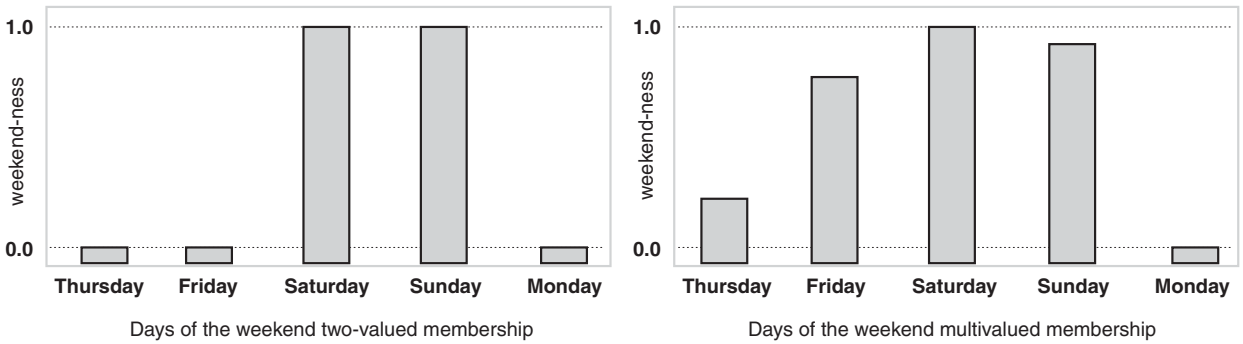


(i) Fuzzy set.

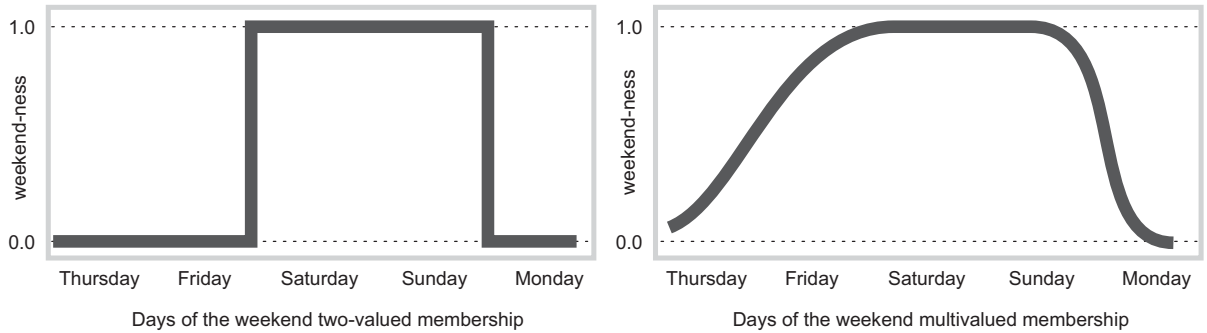
Figure 6: Classical and fuzzy sets for weekend (diagram from Matlab).

Classical Sets and Fuzzy Sets:

- Use *membership functions* to measure the degree (*membership grade*).
- How do you understand the phrase “The current season is Summer”?



(a) Discrete membership function.



(b) Continuous membership functions.

Figure 7: Discrete and continuous membership functions (diagram from Matlab).

Classical Sets and Fuzzy Sets:

- Multiple *fuzzy sets* where *membership functions* are with different labels (*fuzzy terms* or *linguistic terms*).

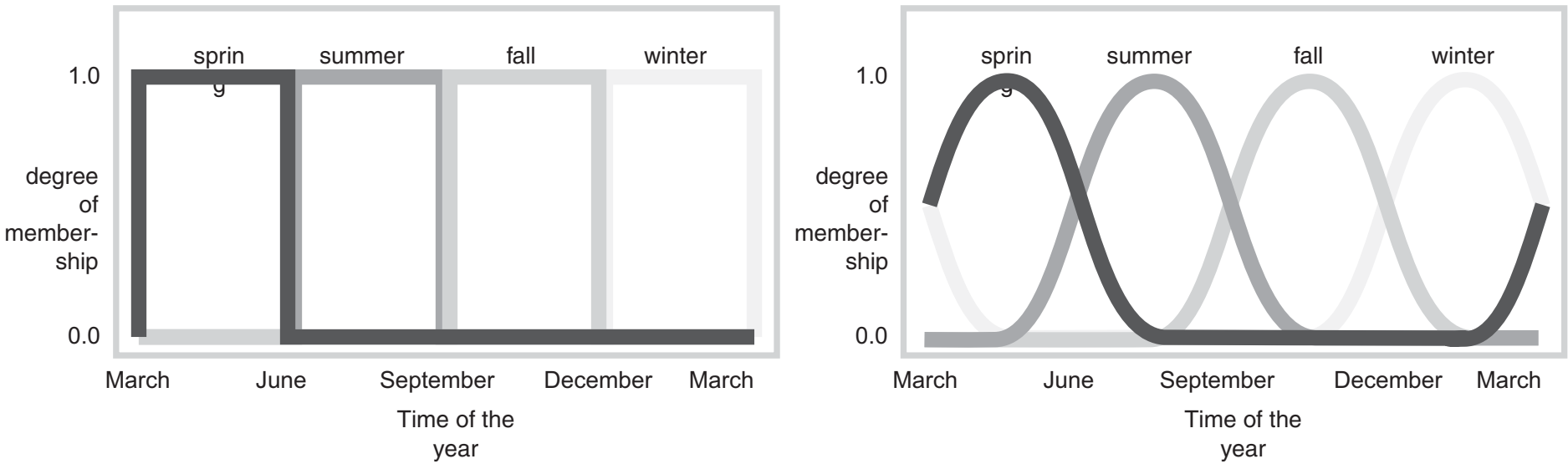


Figure 8: Fuzzy sets with different labels (diagram from Matlab).

Driving problem: I am driving and want to keep a safety distance between cars.

When the distance from the front car is x, what speed should I keep?

Linguistic Rules:

Rule 1: **If** distance is *small* **Then** speed is *low*

Rule 2: **If** distance is *medium* **Then** speed is *steady*

Rule 3: **If** distance is *large* **Then** speed is *high*

***More specific question**: When the distance from the front car is 3.5 m or so, what speed should I keep?*

Remark: Different people have different meaning of *small*, *medium*, *large*, *high*, *steady* and *low*. How do you define them? How do you explain these terms to computers?

Linguistic Rules:

Rule 1: **If** distance is *small* **Then** speed is *low*

Rule 2: **If** distance is *medium* **Then** speed is *steady*

Rule 3: **If** distance is *large* **Then** speed is *high*

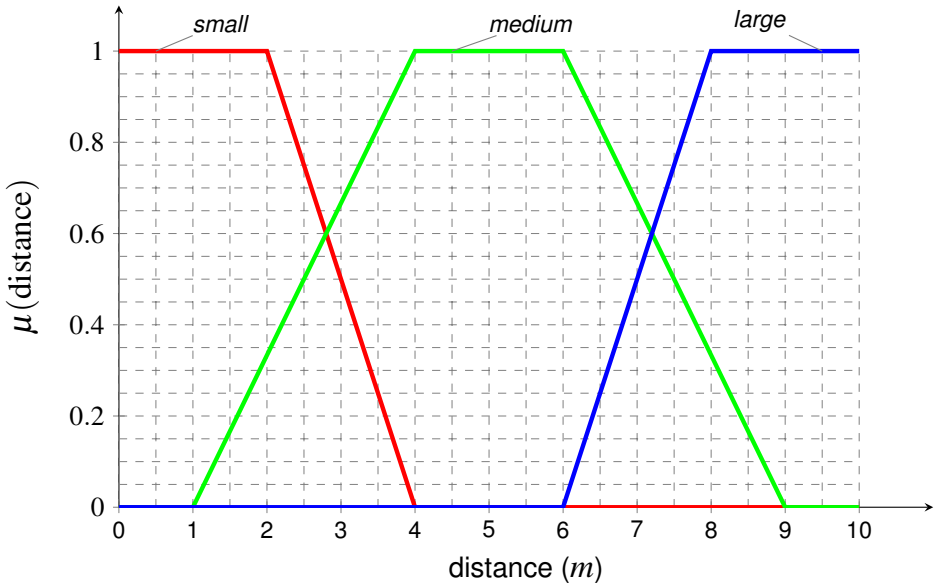


Figure 9: Fuzzification Process.

More specific question: *When the distance from the front car is 3.5 m or so, what speed should I keep?* ***My Answer***: *The speed should be not very "low", more toward "steady" but definitely not "high".*

Linguistic Rules:

Rule 1: **If** distance is *small* **Then** speed is *low*

Rule 2: **If** distance is *medium* **Then** speed is *steady*

Rule 3: **If** distance is *large* **Then** speed is *high*

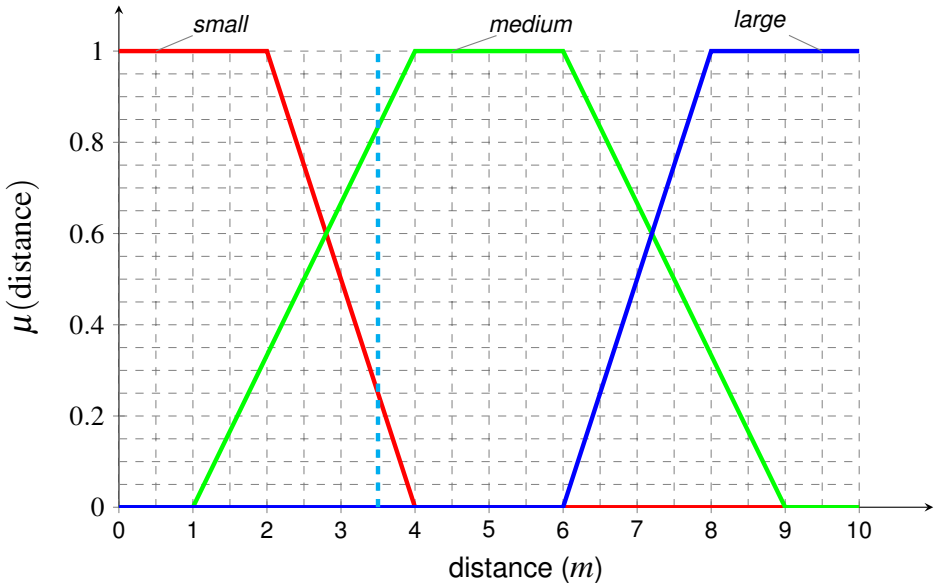


Figure 9: Fuzzification Process.

More specific question: *When the distance from the front car is 3.5 m or so, what speed should I keep?* ***My Answer***: *The speed should be not very "low", more toward "steady" but definitely not "high".*

Linguistic Rules:

Rule 1: **If** distance is *small* **Then** speed is *low*

Rule 2: **If** distance is *medium* **Then** speed is *steady*

Rule 3: **If** distance is *large* **Then** speed is *high*

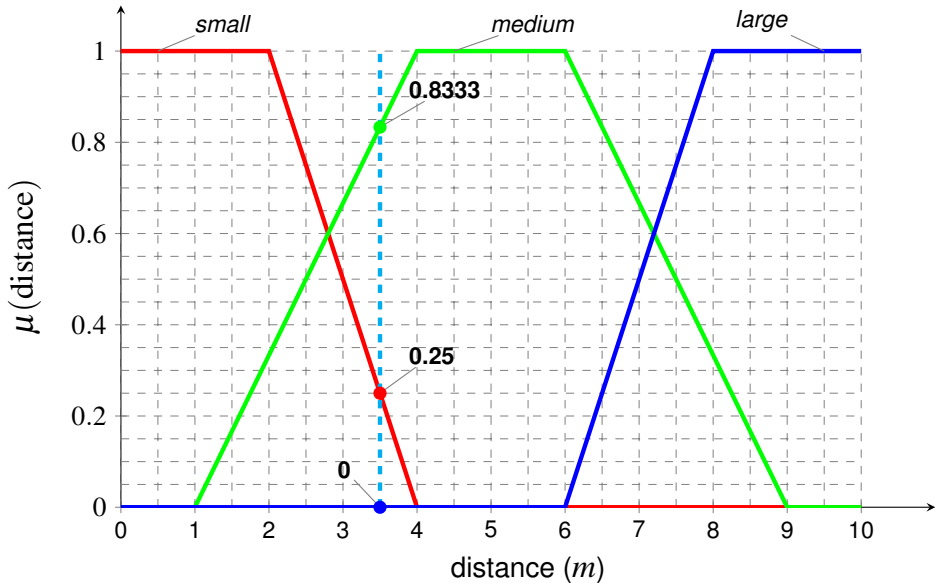


Figure 9: Fuzzification Process.

More specific question: *When the distance from the front car is 3.5 m or so, what speed should I keep?* ***My Answer***: *The speed should be not very "low", more toward "steady" but definitely not "high".*

Fuzzy Inference System

Fuzzy Inference System

- A **fuzzy inference system** (FIS) is also known as *fuzzy-rule-based system*, *fuzzy expert system*, *fuzzy logic system*, *fuzzy model*, *fuzzy associative memory* (FAM) and *fuzzy logic controller* and *fuzzy system*.
- An FIS is a computing framework based on the concepts of fuzzy set theory, fuzzy (If-Then) rules and fuzzy reasoning.
- An FIS consists of **4 components**: *fuzzifier*, *Knowledge base* (rule base or database), *fuzzy inference engine* and *defuzzifier*.

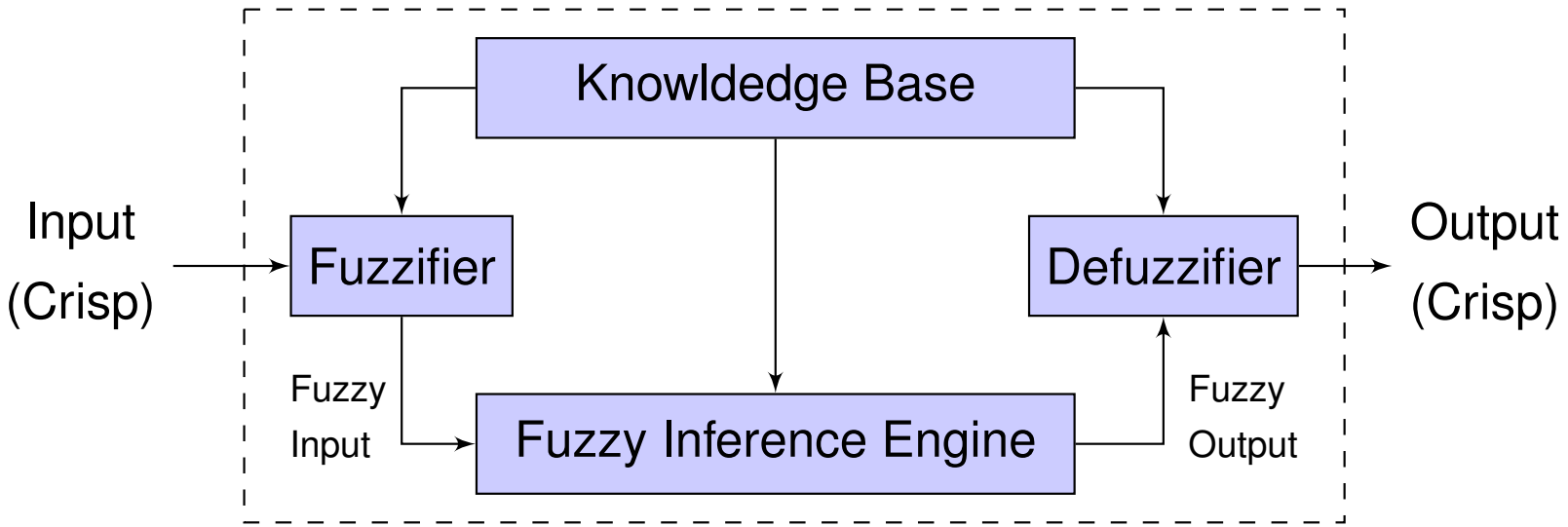
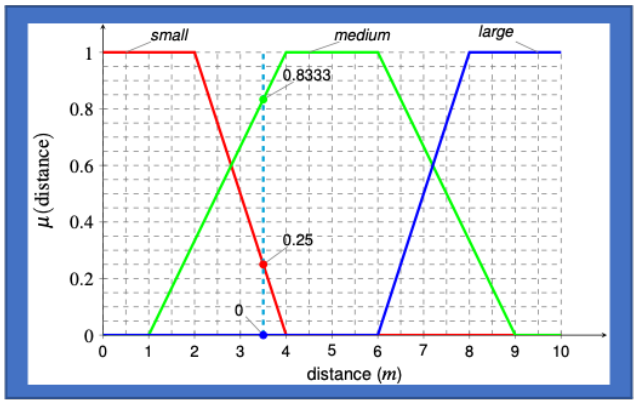


Figure 10: A diagram of fuzzy inference system.

Fuzzy Inference System

- **Fuzzifiers:** It maps the crisp (real-valued) input into a fuzzy set defined in the universe of discourse (the domain of the fuzzy set) $\mathbf{X}$ characterised by membership functions. This process is called *fuzzification*. Note: The input can also be a fuzzy set.
- **Knowledge Base:** It is a database consisting of linguistic rules in If-Then format.
- **Fuzzy Inference Engine:** Using the If-Then rules in *Knowledge base*, it performs reasoning by producing a fuzzy output according to the fuzzy input given by the *fuzzifier*.
- **Defuzzifiers:** It converts the fuzzy output given by the *fuzzy inference engine* to produce a crisp (real-valued) output. This process is called *defuzzifiaction*.

Fuzzy Inference System

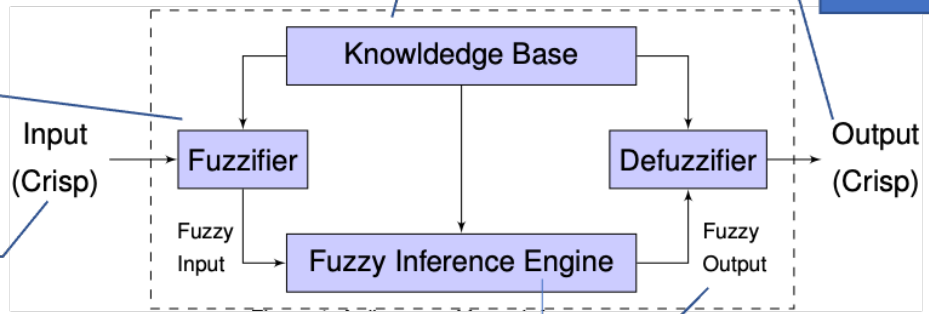


Rule 1: **If** distance is *small* **Then** speed is *low*

Rule 2: **If** distance is *medium* **Then** speed is *steady*

Rule 3: **If** distance is *large* **Then** speed is *high*

Output:
20 miles/hour

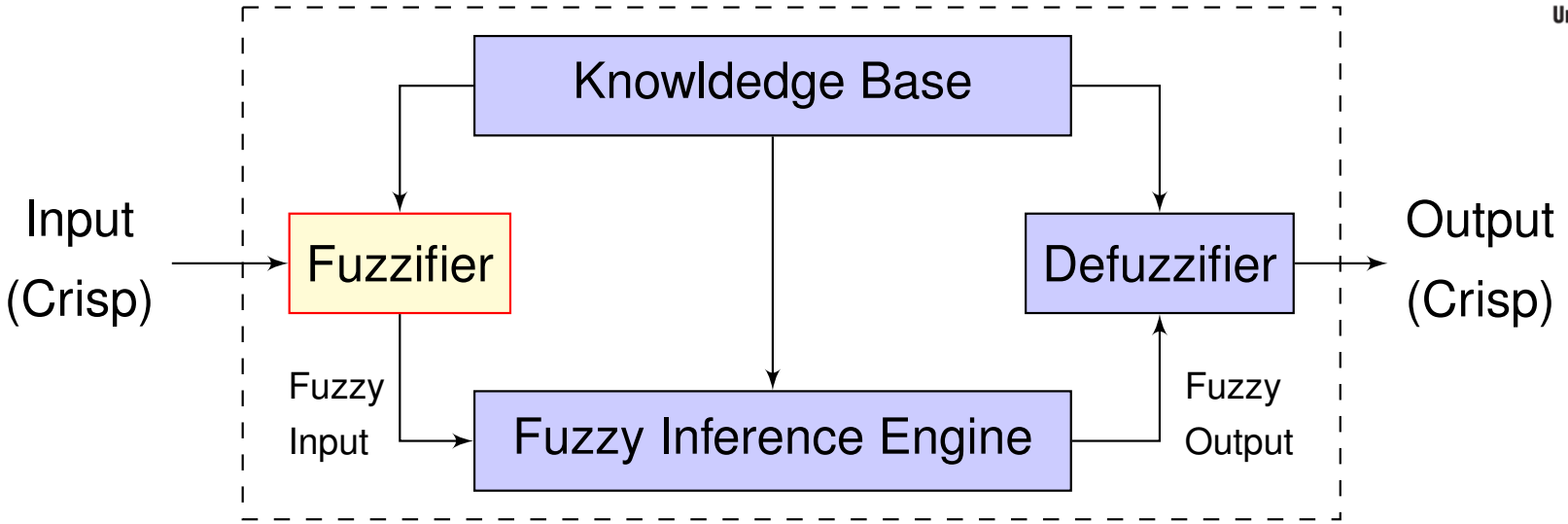


Input: 3.5m

Level of Distance	Speed
0.25 of Small	Not very Low
0.8333 of Medium	More Steady
0 of Large	Definitely not High

Speed should not be very Low, more towards Steady but definitely not High

Fuzzifiers


Knowldedge Base
Input
(Crisp)
Fuzzifier
Defuzzifier
Output
(Crisp)
Fuzzy
Input
Fuzzy Inference Engine
Fuzzy
Output

Fuzzifiers

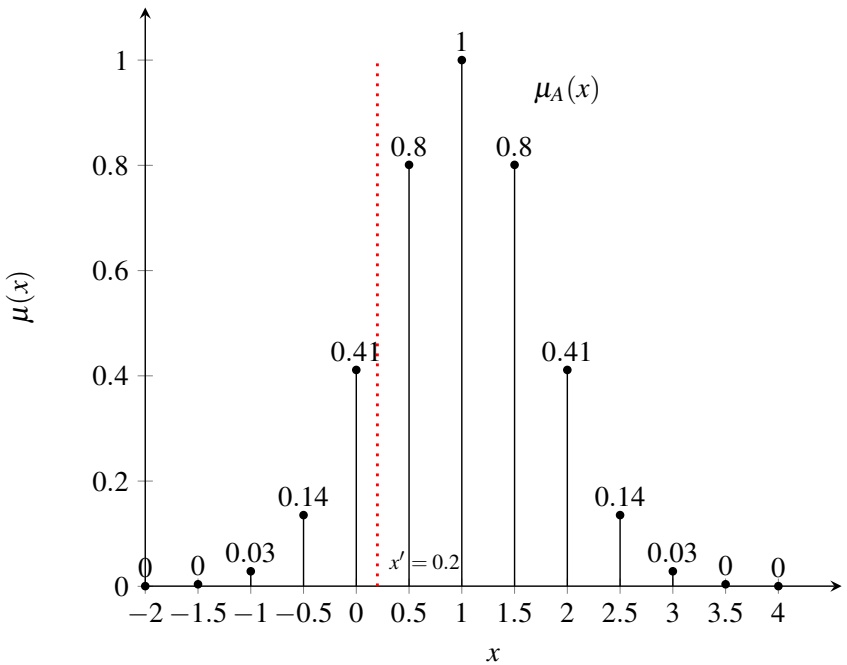
- The input is turned to fuzzy sets thought membership functions.
- A fuzzy set is represented by a membership functions (associated with a label called *fuzzy term* or *linguistic term*).

Property of membership functions:

- A membership function can be *discrete* or *continuous*.
- A membership function denoted by $\mu_A(x)$ corresponding to fuzzy set A (A is the *fuzzy term* or *linguistic term*) is characterised by a linear/nonlinear function of premise variable x.
  - E.g., "**If** x is *Positive* **Then** y is *Fast*"; "**If** distance is *small* **Then** speed is *low*"
- It is in the range of $0 \leq \mu_A(x) \leq 1$.
- Considering a particular reading, say, x', $0 \leq \mu_A(x') \leq 1$ is called the *membership degree/grade* or *degree/grade of membership*.
  - E.g., $\mu_A(x') = 0.5$ when $x' = 2$; $\mu_{small}(\text{distance}) = 0.8333$ when $\text{distance} = 3.5$

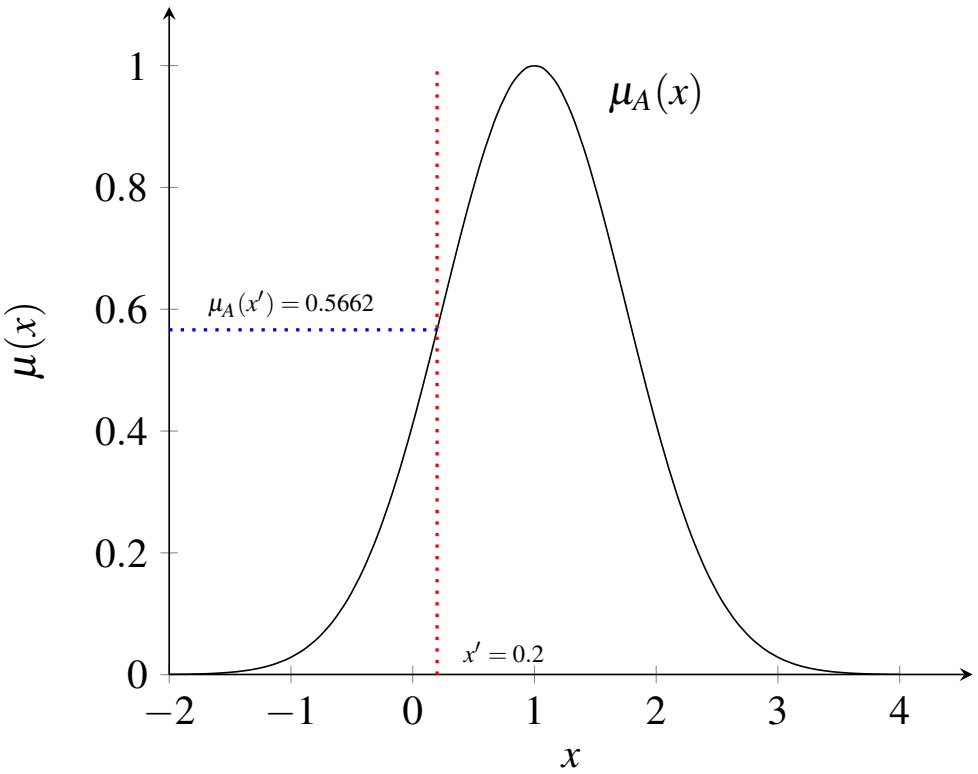
Fuzzifiers

- Notation for discrete fuzzy set: $A = \left\{ \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \cdots \right\} = \left\{ \sum_{x_i \in X} \frac{\mu_A(x_i)}{x_i} \right\}$.
- The horizontal bar is not a quotient but rather a delimiter.
- The summation symbol is not for algebraic summation, but denotes the collection or aggregation of each element. The "+" signs are not the algebraic "add" but are an aggregation or collection operator.
- Example: $A = \left\{ \frac{0}{-2} + \frac{0}{-1.5} + \frac{0.03}{-1} + \frac{0.14}{-0.5} + \frac{0.41}{0} + \frac{0.8}{0.5} + \frac{1}{1} + \frac{0.8}{1.5} + \frac{0.41}{2} + \frac{0.14}{2.5} + \frac{0.03}{3} + \frac{0}{3.5} + \frac{0}{4} \right\}$
- $\mu_A(0.2) = 0.41$



Fuzzifiers

- Notation for continuous fuzzy set: $A = \left\{ \int_X \frac{\mu_A(x)}{x} \right\}$.
- The horizontal bar is not a quotient but rather a delimiter.
- The integral sign is not an algebraic integral but a continuous function-theoretic aggregation operator for continuous variables.



More properties of membership functions:

- *Core:* The core of a membership function for a fuzzy set $\tilde{A}$ is the region of the universe of which $\mu_{\tilde{A}}(x) = 1$.
- *Support:* It is defined as the region of the universe of which $\mu_{\tilde{A}}(x) > 0$.
- *Boundaries:* It is defined as the region of the universe of which $0 < \mu_{\tilde{A}}(x) < 1$.

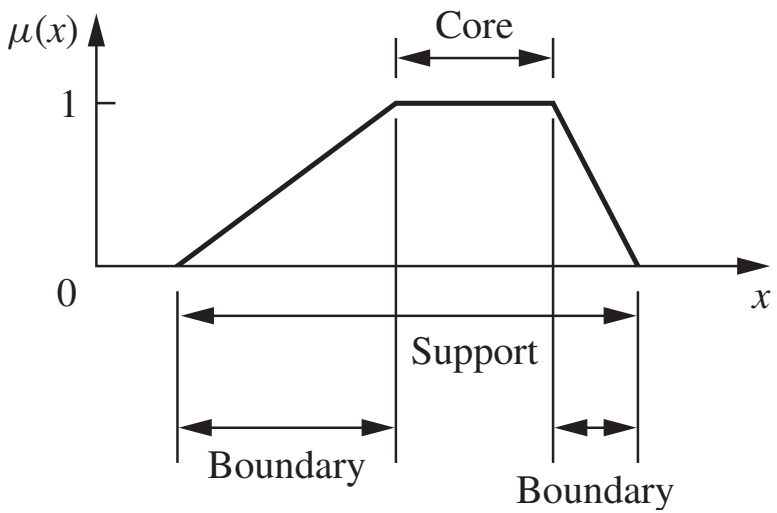


Figure 11: Core, support and boundaries of a fuzzy set.

More properties of membership functions:

- *Normal/subnormal fuzzy set:* A fuzzy set is said to be normal if its membership function has at least one element of x whose membership grade is 1, i.e., $\mu_{\tilde{A}} = 1$, otherwise, a subnormal fuzzy set.
- *height* of a fuzzy set hgt$(\tilde{A})$: $\tilde{A} = \max\{\mu_{\tilde{A}}\}$ which is the maximum membership degree of a membership function. So, hgt$(\tilde{A}) = 1 \Rightarrow$ normal membership function; hgt$(\tilde{A}) < 1 \Rightarrow$ subnormal membership function.

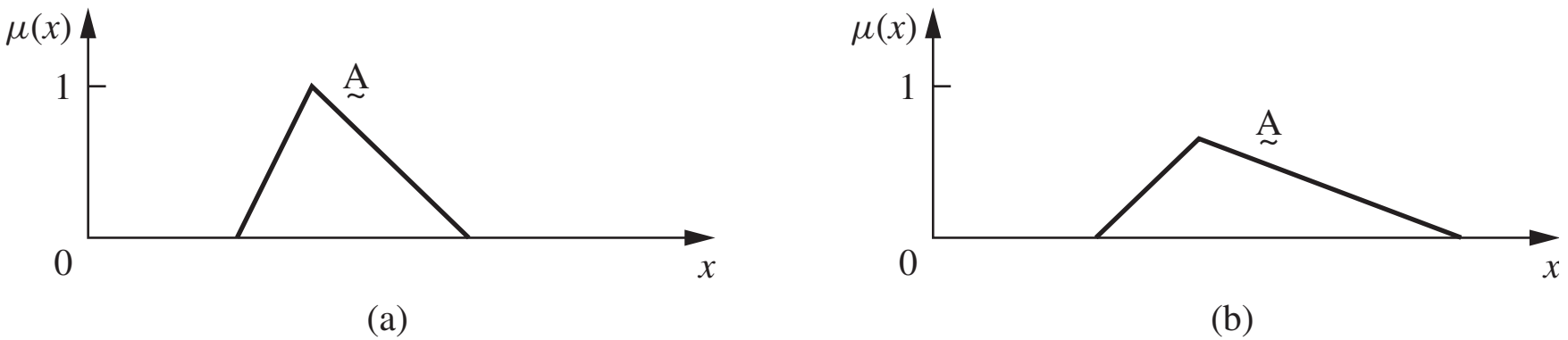


Figure 12: (a) Normal and (b) subnormal fuzzy sets.

More properties of membership functions:

- *Convex/non-convex fuzzy set:* A fuzzy set is said to be convex if membership function are strictly monotonically increasing/decreasing or strictly monotonically increasing and then decreasing, otherwise, a non-convex fuzzy set.

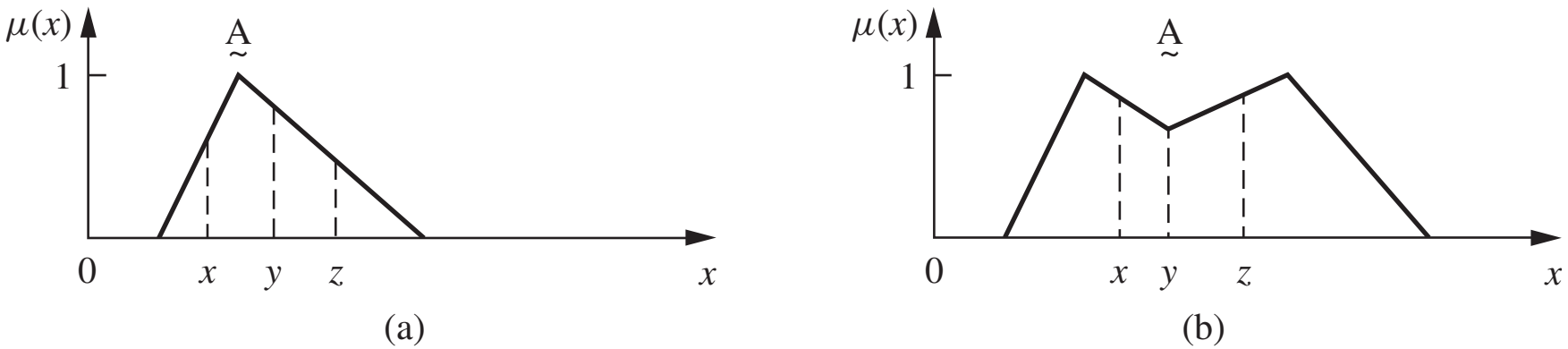


Figure 13: (a) convex and (b) non-convex fuzzy sets.

- Fuzzification is the process turning the crisp input to a fuzzy value (membership grade).

Linguistic Rules:

Rule 1: **If** distance is *small* **Then** speed is *low*

Rule 2: **If** distance is *medium* **Then** speed is *steady*

Rule 3: **If** distance is *large* **Then** speed is *high*

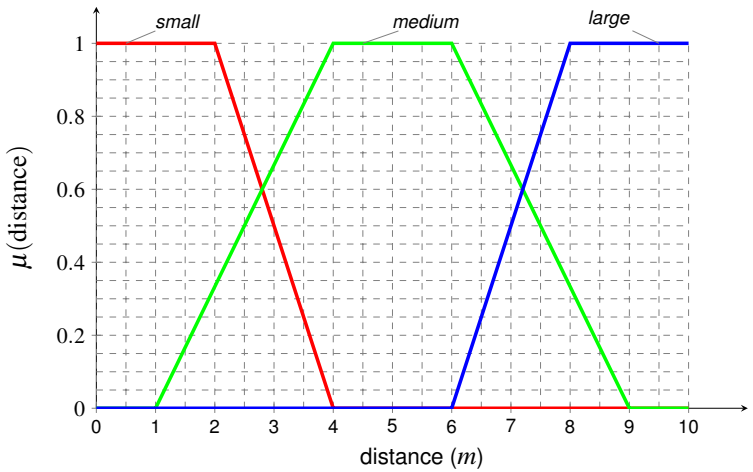


Figure 14: Fuzzification Process.

Fuzzifiers

- Fuzzification is the process turning the crisp input to a fuzzy value (membership grade).

Linguistic Rules:

Rule 1: **If** distance is *small* **Then** speed is *low*

Rule 2: **If** distance is *medium* **Then** speed is *steady*

Rule 3: **If** distance is *large* **Then** speed is *high*

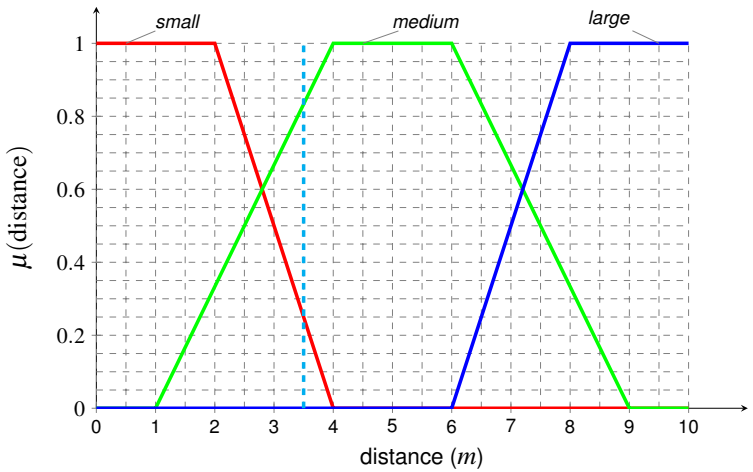


Figure 14: Fuzzification Process.

- Fuzzification is the process turning the crisp input to a fuzzy value (membership grade).

Linguistic Rules:

Rule 1: **If** distance is *small* **Then** speed is *low*

Rule 2: **If** distance is *medium* **Then** speed is *steady*

Rule 3: **If** distance is *large* **Then** speed is *high*

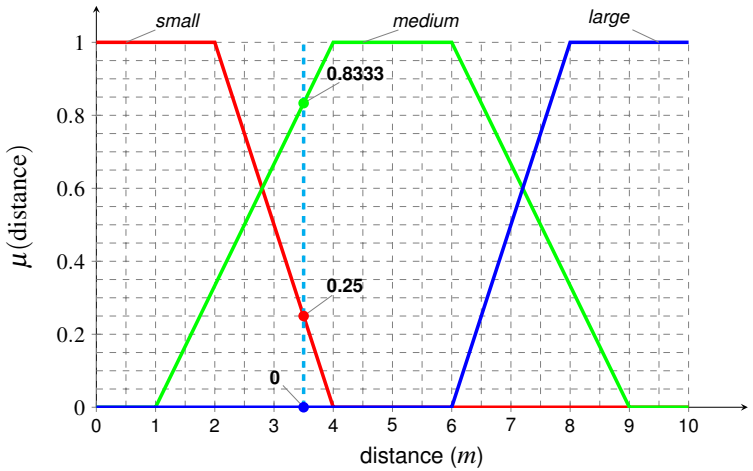


Figure 14: Fuzzification Process.

Common Membership Functions

- **Singleton membership function:** $\mu_{\tilde{A}}(x) = \begin{cases} 1, & \text{if } x = a \\ 0, & otherwise \end{cases}$
- **Triangular membership function:** $\mu_{\tilde{A}}(x) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{m-a}, & a < x \leq m \\ \frac{b-x}{b-m}, & m < x < b \\ 0, & x \geq b \end{cases}$ where $a \leq m \leq b$

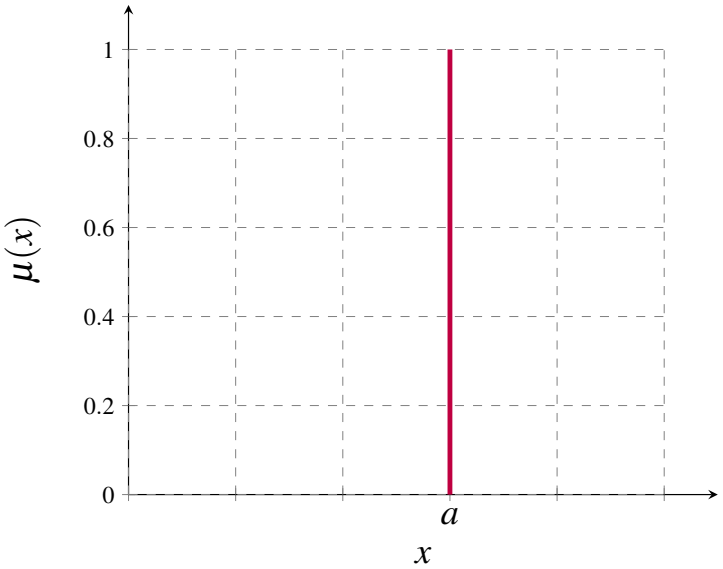


Figure 15: Singleton membership function.

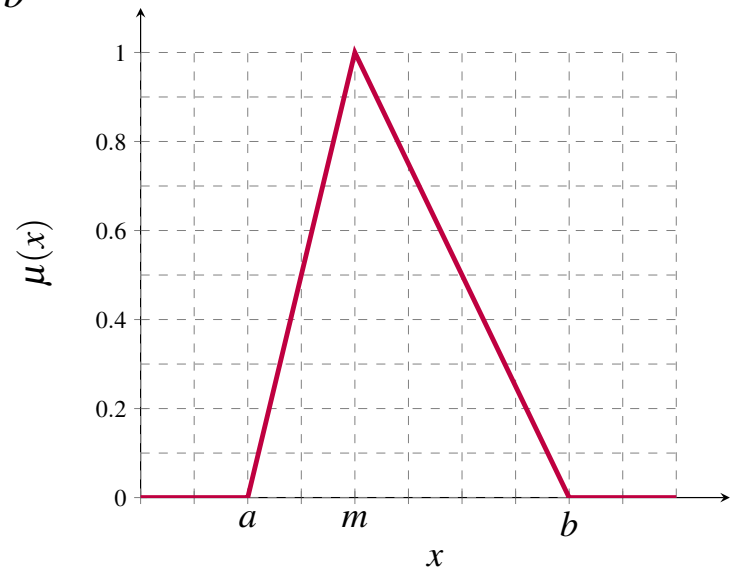


Figure 16: Triangular membership function.

Common Membership Functions

- **Trapezoidal membership function:** $\mu_{\tilde{A}}(x) = \begin{cases} 0, & x < a \text{ or } x > d \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b \leq x \leq c \\ \frac{d-x}{d-c}, & c \leq x \leq d \end{cases}$ where $a \leq b \leq c \leq d$
- **Gaussian membership function:** $\mu_{\tilde{A}}(x) = e^{-\frac{(x-m)^2}{2\sigma^2}}$

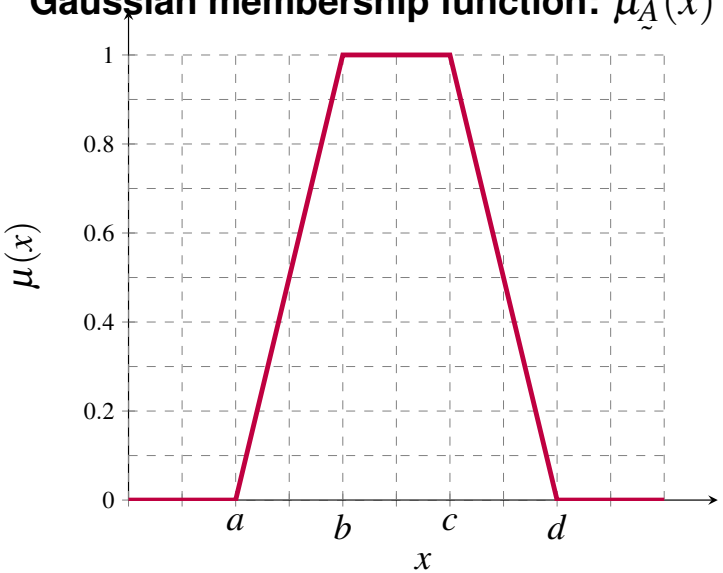


Figure 17: Trapezoidal membership function.

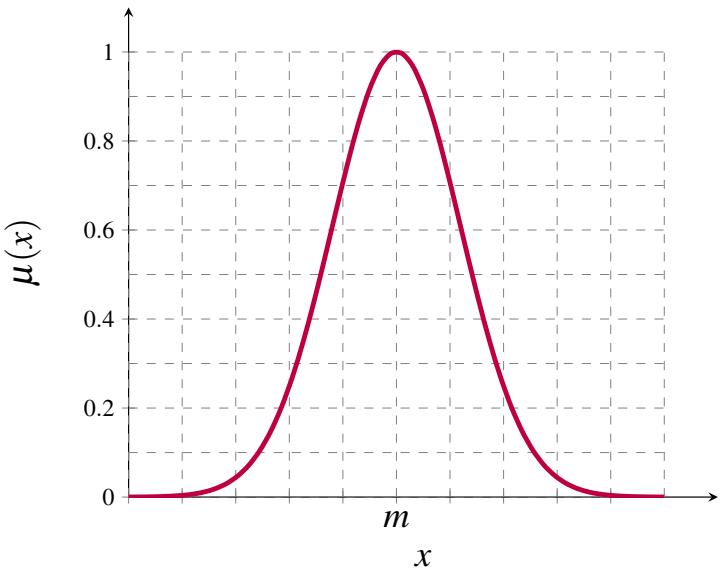


Figure 18: Gaussian membership function.

Common Membership Functions

- **Z-shaped membership function:** $\mu_{\tilde{A}}(x) = \begin{cases} 0, & x > d \\ \frac{d-x}{d-c}, & c \leq x \leq d \\ 1, & x < c \end{cases}$ where $c \leq d$
- **S-shaped membership function:** $\mu_{\tilde{A}}(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & x > b \end{cases}$ where $a \leq b$

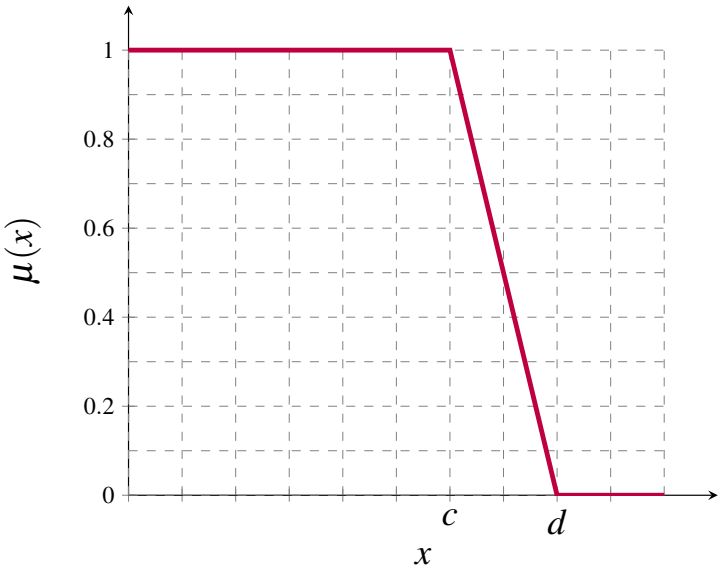


Figure 19: Z-shaped membership function.

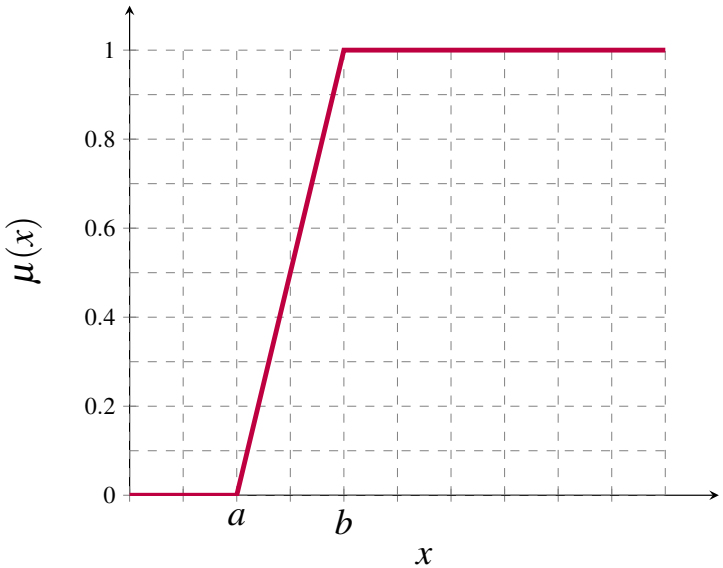
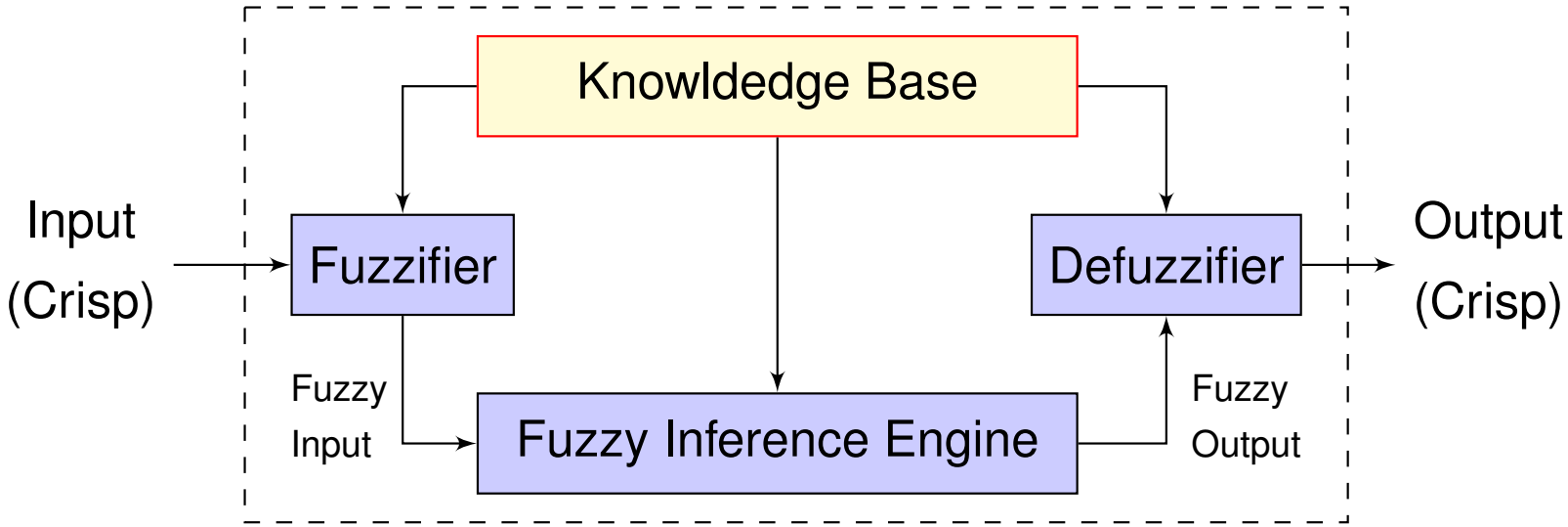


Figure 20: S-shaped membership function.

Knowledge Base



Recall the example - Linguistic Rules:

Rule 1: **If** distance is *small* **Then** speed is *low*

Rule 2: **If** distance is *medium* **Then** speed is *steady*

Rule 3: **If** distance is *large* **Then** speed is *high*

Knowledge Base

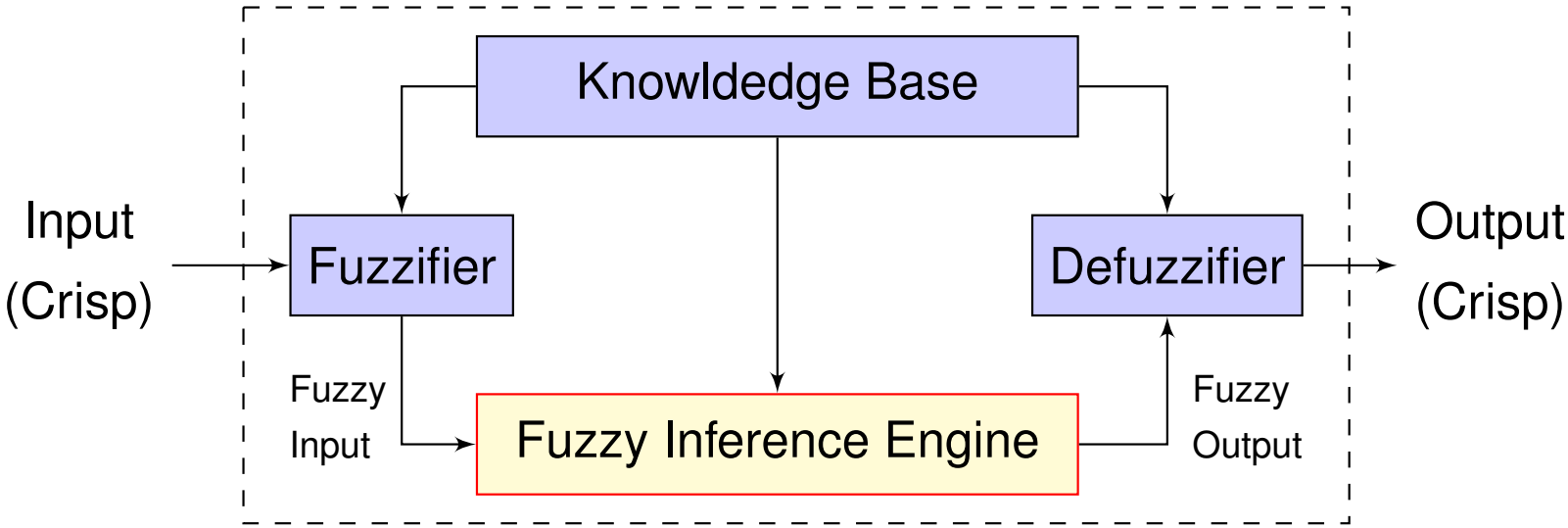
- The knowledge base is the rule base representing the expertise knowledge dealing with a specific problem.
- Linguistic rule: IF premise (antecedent) THEN conclusion (consequent).
- The knowledge base can have more than one rule.

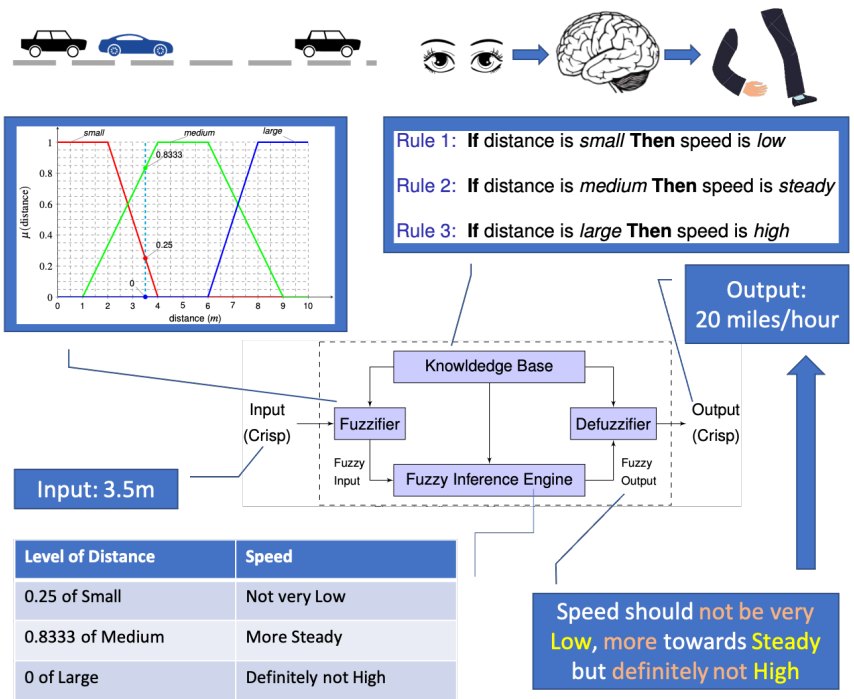
General rule format:

Rule i: **IF** $\underbrace{x_1 \text{ is } \underset{\sim}{A}_{i1} \text{ and/or } x_2 \text{ is } \underset{\sim}{A}_{i2} \text{ and/or } \cdots}_{antecedent}$ **THEN** $\underbrace{y \text{ is } \underset{\sim}{B}_i}_{consequent}$, $i = 1, 2, \ldots, r$

- $x_1, x_2, \cdots$ are the fuzzy/linguistic variables.
- y is the output of the fuzzy inference system.
- "and' and "or" are fuzzy operators.
- $\underset{\sim}{A}_{i1}, \underset{\sim}{A}_{i2}, \cdots$ are the fuzzy sets (associated with a linguistic variable) representing the i^{th} *antecedent* pairs.
- $\underset{\sim}{B}^i$ is the fuzzy set (associated with a linguistic variable) representing the i^{th} *consequent*.
- r is number of rules.

Fuzzy Inference Engine





Fuzzy Inference Engine

- **Fuzzy inference engine** is to produce the fuzzy output according to the crisp inputs based on the knowledge (knowledge base) represented by IF-THEN rule. This is the process of reasoning. It generally involves two processes, i.e., *rule evaluation* and *rule aggregation*
  - **Rule evaluation** (implication) is to apply the fuzzy set operators (AND, OR, NOT) to the antecedents to determine the firing strength of each rule.
  - **Rule aggregation** is to combine the output (consequents) fuzzy sets using the firing strengths obtained in the process of *rule evaluation*.
- There are three standard fuzzy set operations
  - *Fuzzy union* operation (*OR*), also known as *t-norm* or conjunction operator.
  - *Fuzzy intersection* operation (*AND*), also known as *t-conorm*, *s-norm* operation, disjunction operation.
  - *Fuzzy complement* operation (*NOT*).

Example - Linguistic Rules (max for "or" and min for "and"; $x = 3.5$, $y = 2$)

Rule 1: **If** x is *small* and y is *negative* **Then** z is *low*

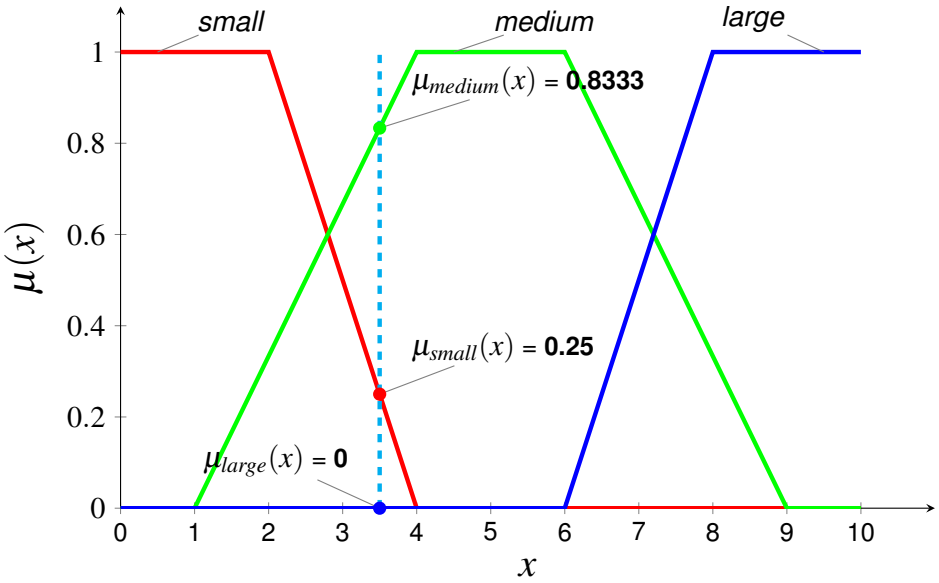
(Rule evaluation: $\min(\mu_{small}(x), \mu_{negative}(y)) = \min(0.8333, 0.0561) = 0.0561$)

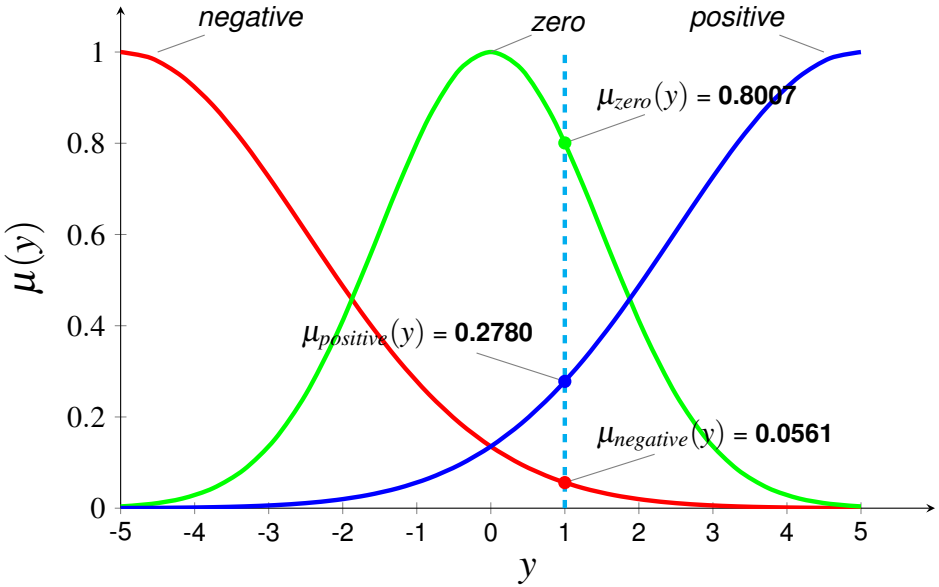
Rule 2: **If** x is *medium* or y is *zero* **Then** z is *middle*

(Rule evaluation: $\max(\mu_{medium}(x), \mu_{zero}(y)) = \max(0.25, 0.8007) = 0.8007$)

Rule 3: **If** x is *large* or y is not *positive* **Then** z is *high*

(Rule evaluation: $\max(\mu_{large}(x), \mu_{not\ positive}(y)) = \max(0, 1 - 0.2780) = 0.7220$)





Fuzzy Inference Engine

Fuzzy OR operator (fuzzy union operator)

- *Maximum:* $\mu_{\tilde{A}\cup\tilde{B}}(x,y) = \mu_{\tilde{A}}(x) \vee \mu_{\tilde{B}}(y) = \max(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(y))$
- *Algebraic sum:* $\mu_{\tilde{A}\cup\tilde{B}}(x,y) = \mu_{\tilde{A}}(x) + \mu_{\tilde{B}}(y) - \mu_{\tilde{A}}(x) \times \mu_{\tilde{B}}(y)$

Fuzzy AND operator (fuzzy intersection operator)

- *Minimum:* $\mu_{\tilde{A}\cap\tilde{B}}(x,y) = \mu_{\tilde{A}}(x) \wedge \mu_{\tilde{B}}(y) = \min(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(y))$
- *Product:* $\mu_{\tilde{A}\cap\tilde{B}}(x,y) = \mu_{\tilde{A}}(x) \times \mu_{\tilde{B}}(y)$

Fuzzy NOT operator (fuzzy complement operator)

- *Complement:* $\mu_{\overline{\tilde{A}}}(x) = 1 - \mu_{\tilde{A}}(x)$

Example (rule evaluation - discrete fuzzy sets): Consider the fuzzy sets $small = \left\{ \frac{0}{1} + \frac{0}{2} + \frac{1}{3} + \frac{0}{4} \right\}$ and $negative = \left\{ \frac{0}{1} + \frac{0.5}{2} + \frac{1}{3} + \frac{0.5}{4} + \frac{0}{5} \right\}$, and the following fuzzy rule:

Rule 1: **If** x is *small* and y is *negative* **Then** z is *low*.

Find the firing strength of Rule 1 when $x = 3$ and $y = 2$ where fuzzy "AND" operation is the minimum operator.

Solution:

Firing strength (Rule evaluation):

$$\mu_{small \cap negative}(3,2) = \min(\mu_{small}(3), \mu_{negative}(2)) = \min(1, 0.5) = 0.5$$

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-min

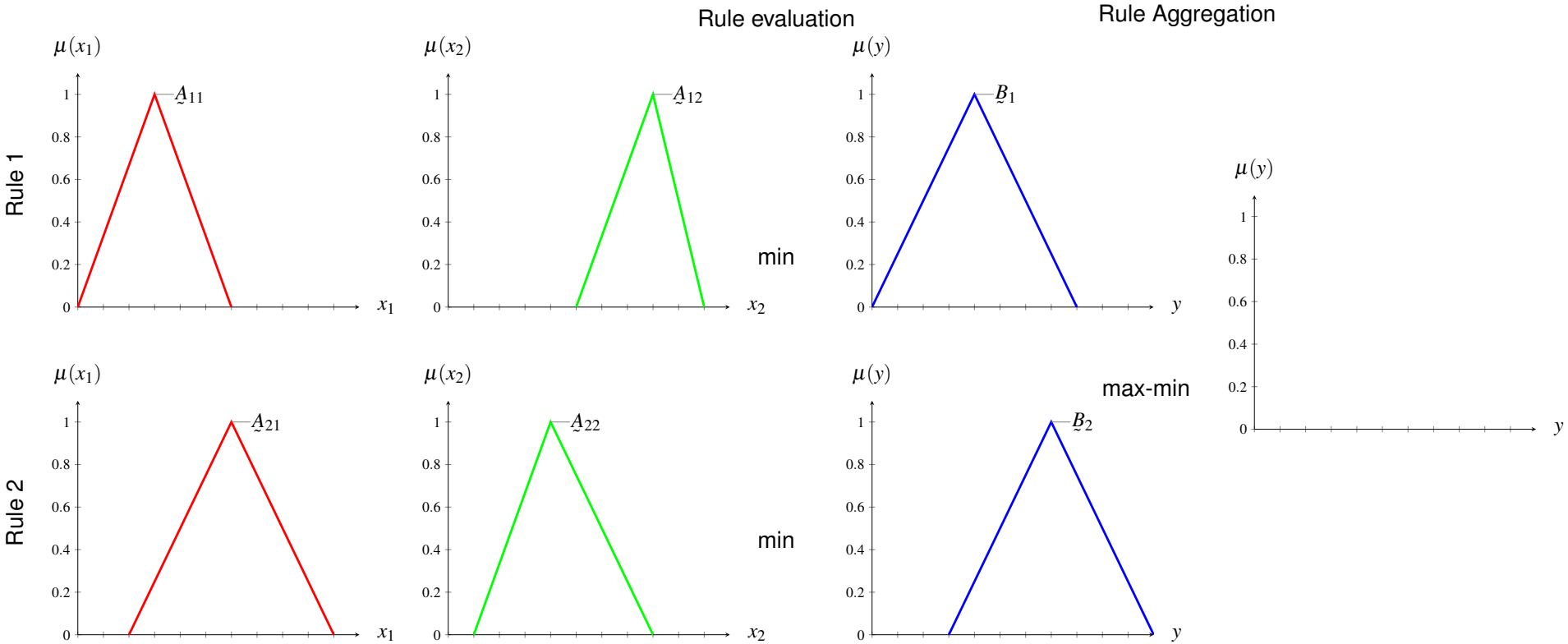


Figure 21: Mamdani (max-min) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Fuzzy Inference Engine

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-min

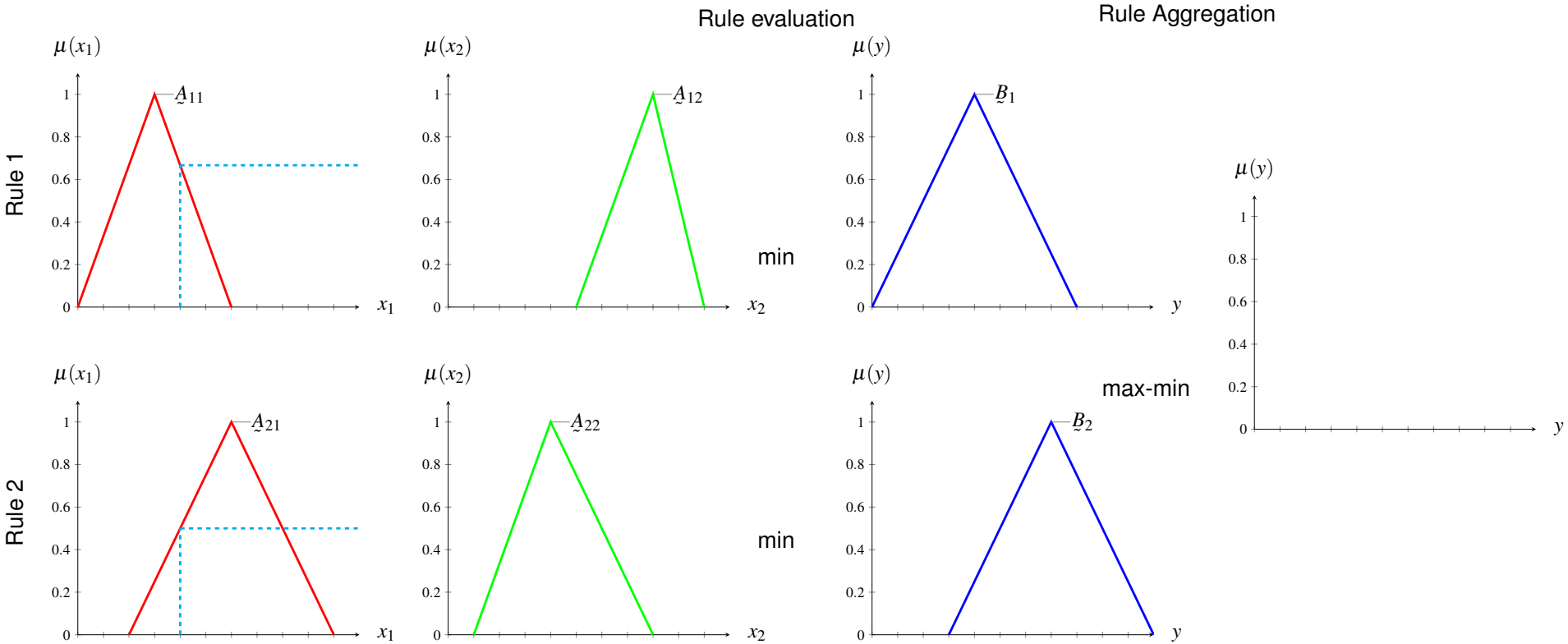


Figure 21: Mamdani (max-min) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-min

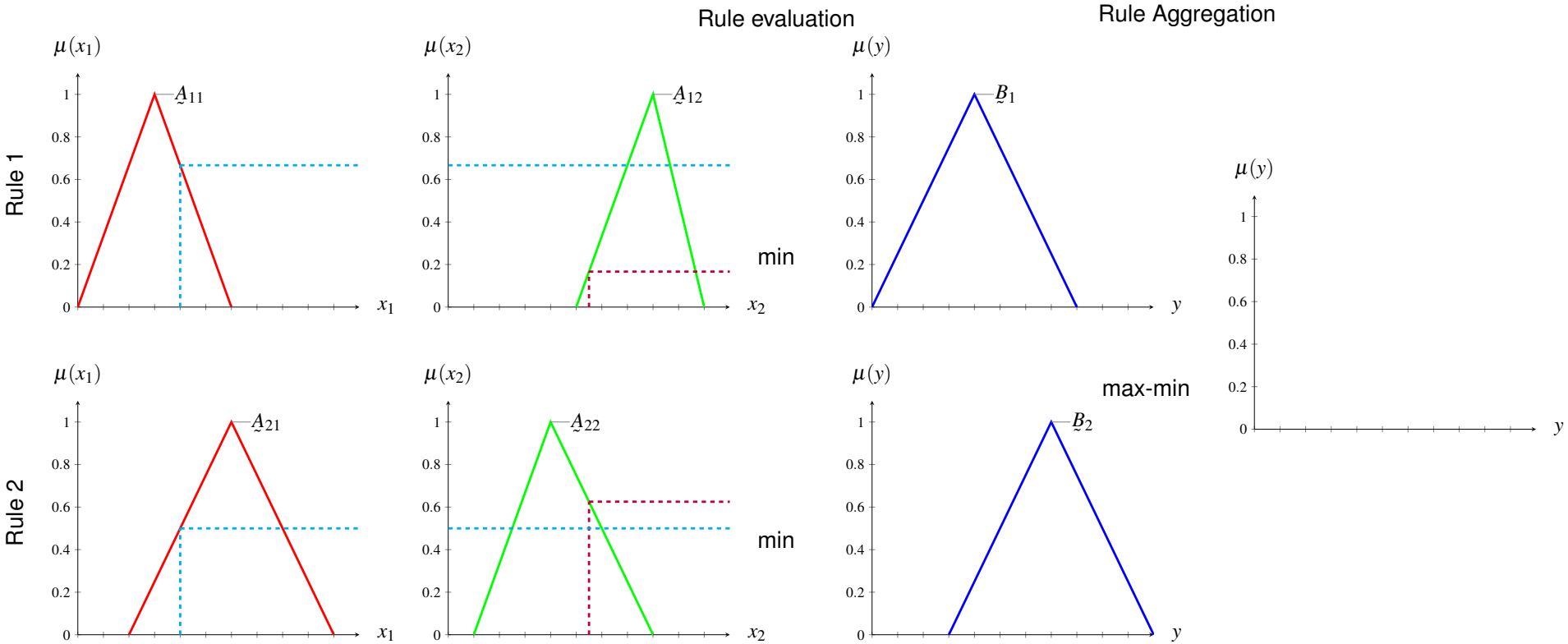


Figure 21: Mamdani (max-min) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Fuzzy Inference Engine

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-min

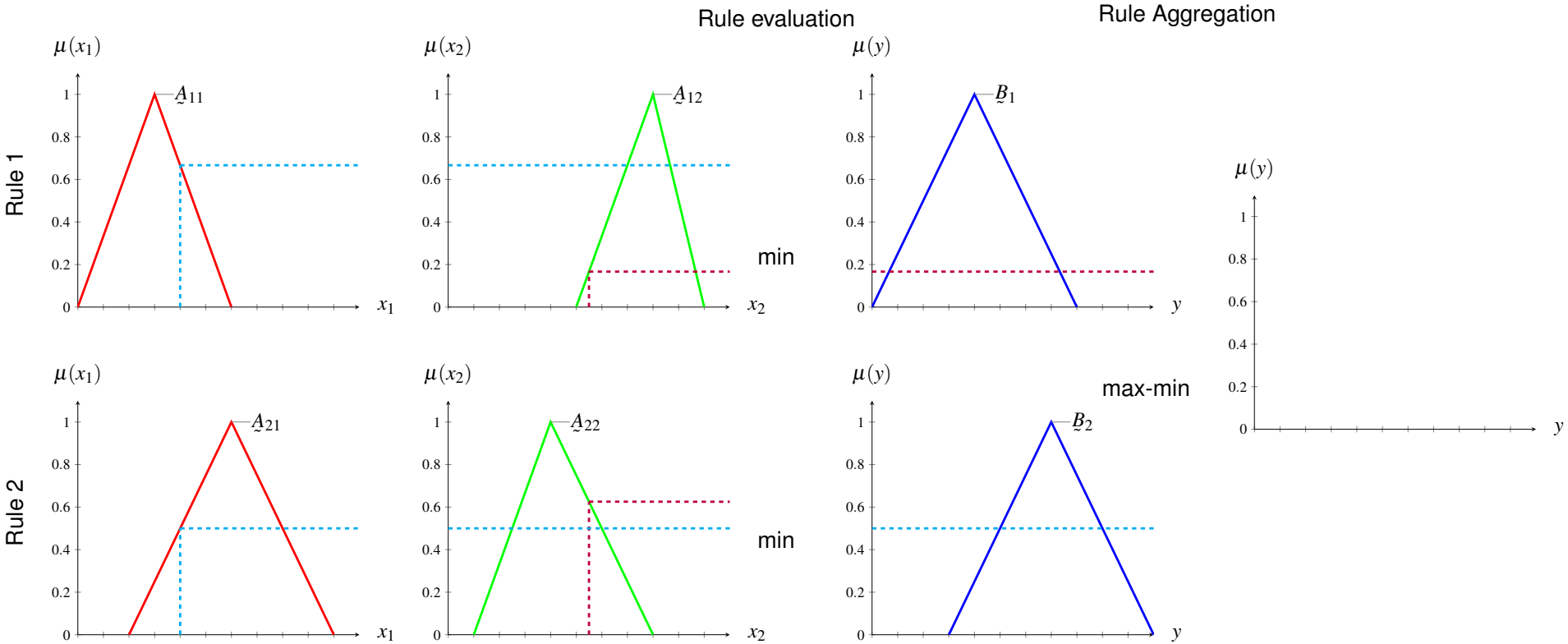


Figure 21: Mamdani (max-min) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Fuzzy Inference Engine

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-min

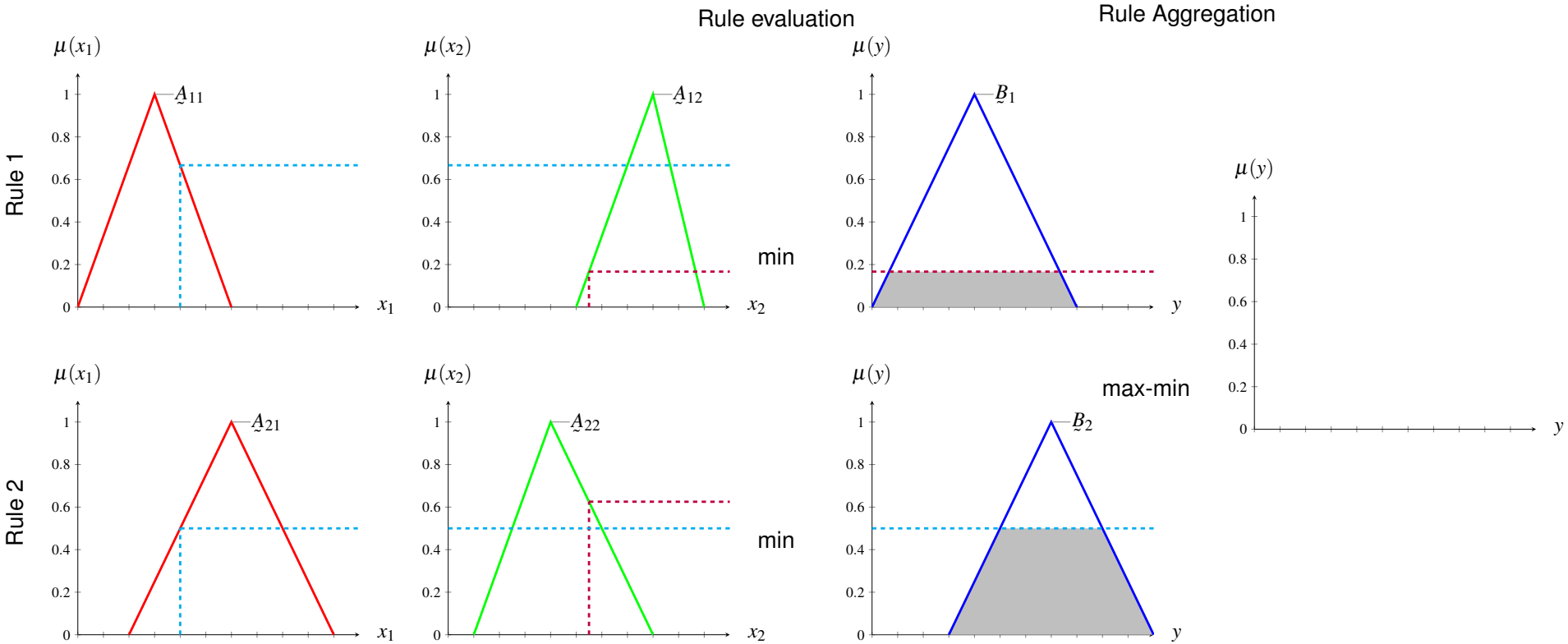


Figure 21: Mamdani (max-min) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Fuzzy Inference Engine

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-min

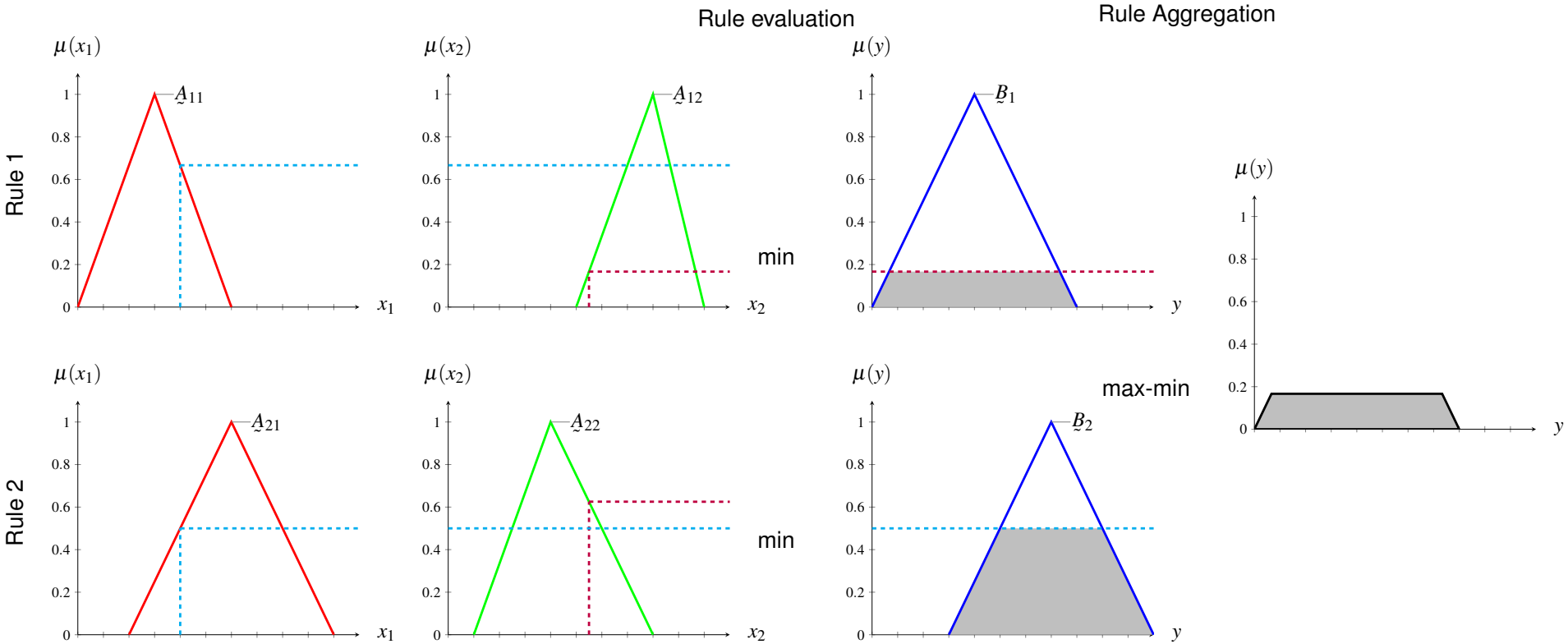


Figure 21: Mamdani (max-min) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-min

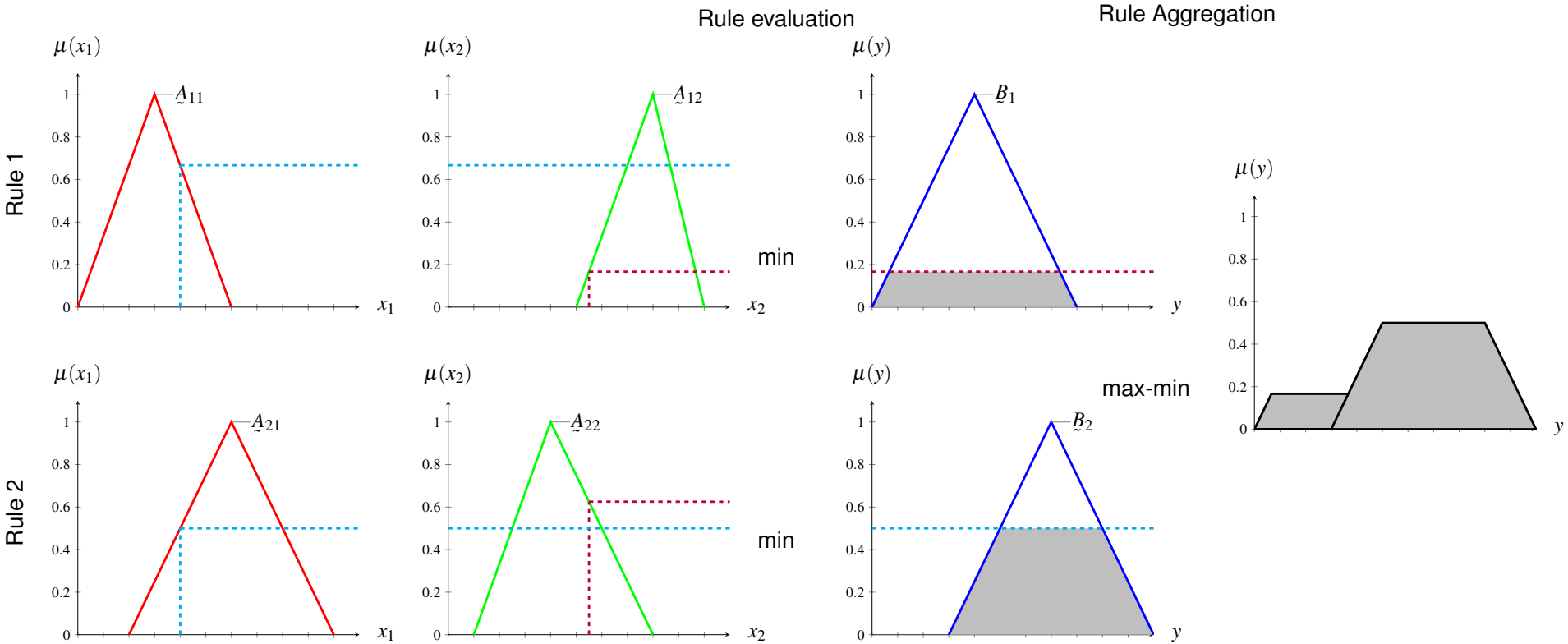


Figure 21: Mamdani (max-min) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Fuzzy Inference Engine

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-min

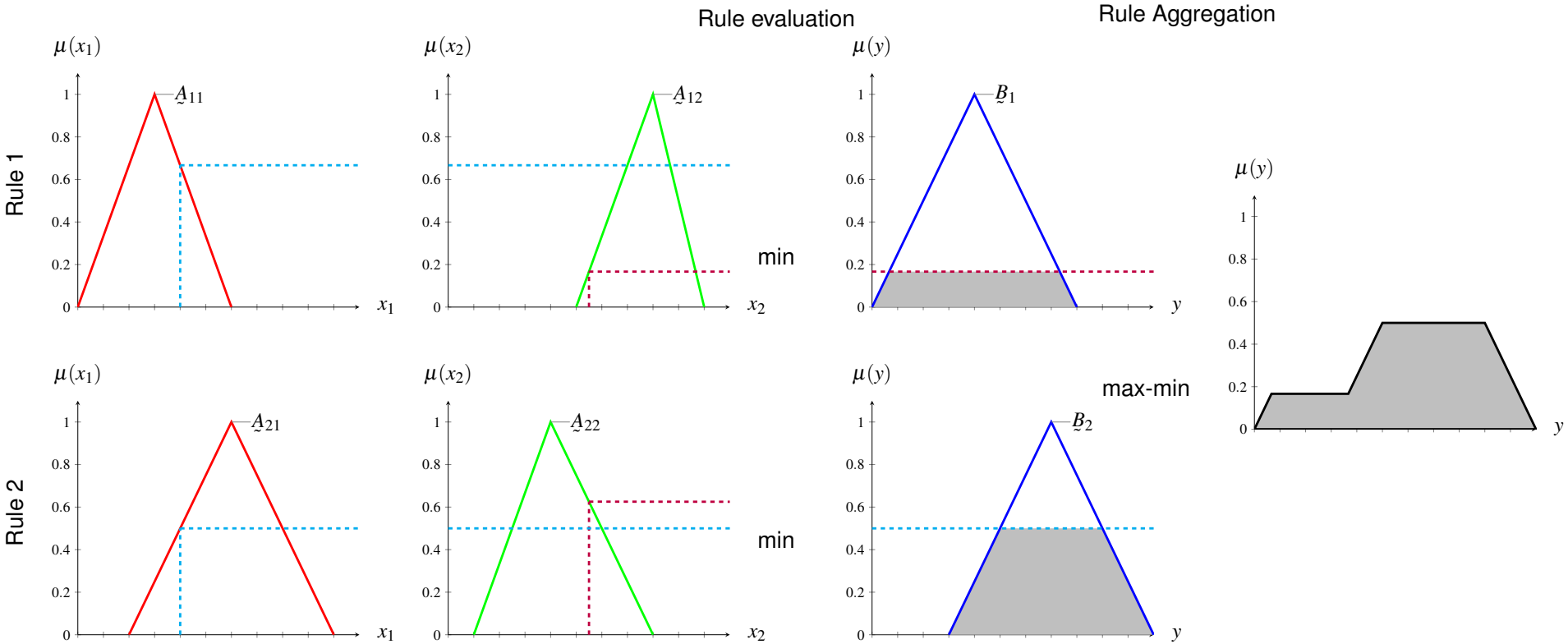


Figure 21: Mamdani (max-min) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Fuzzy Inference Engine

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-product

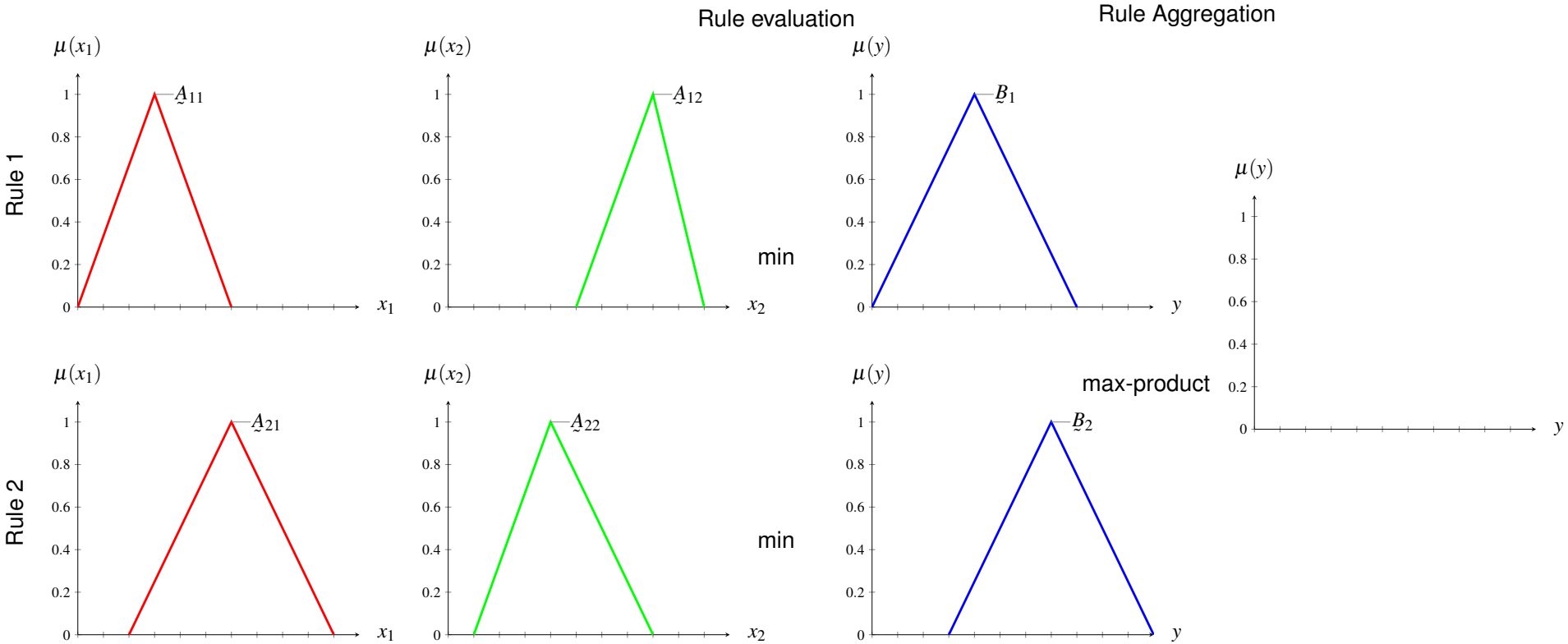


Figure 22: Mamdani (max-product) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Fuzzy Inference Engine

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-product

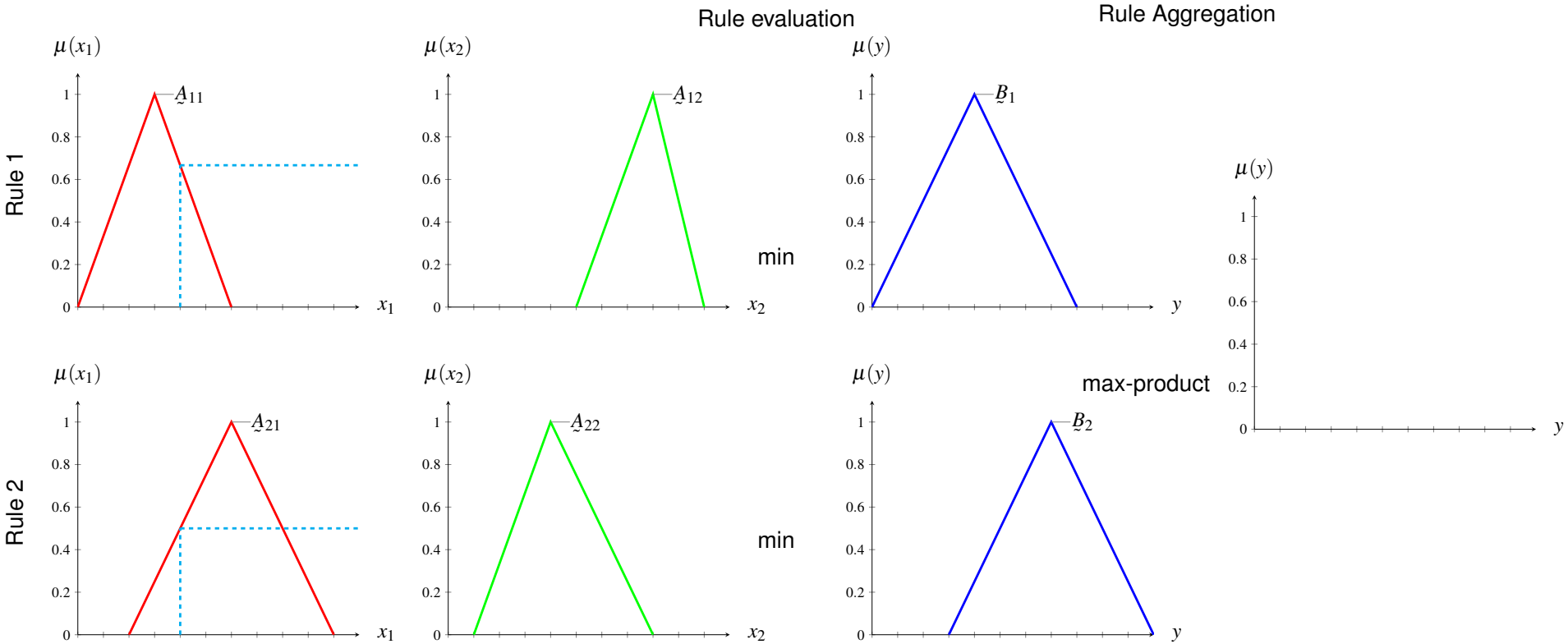


Figure 22: Mamdani (max-product) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Fuzzy Inference Engine

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-product

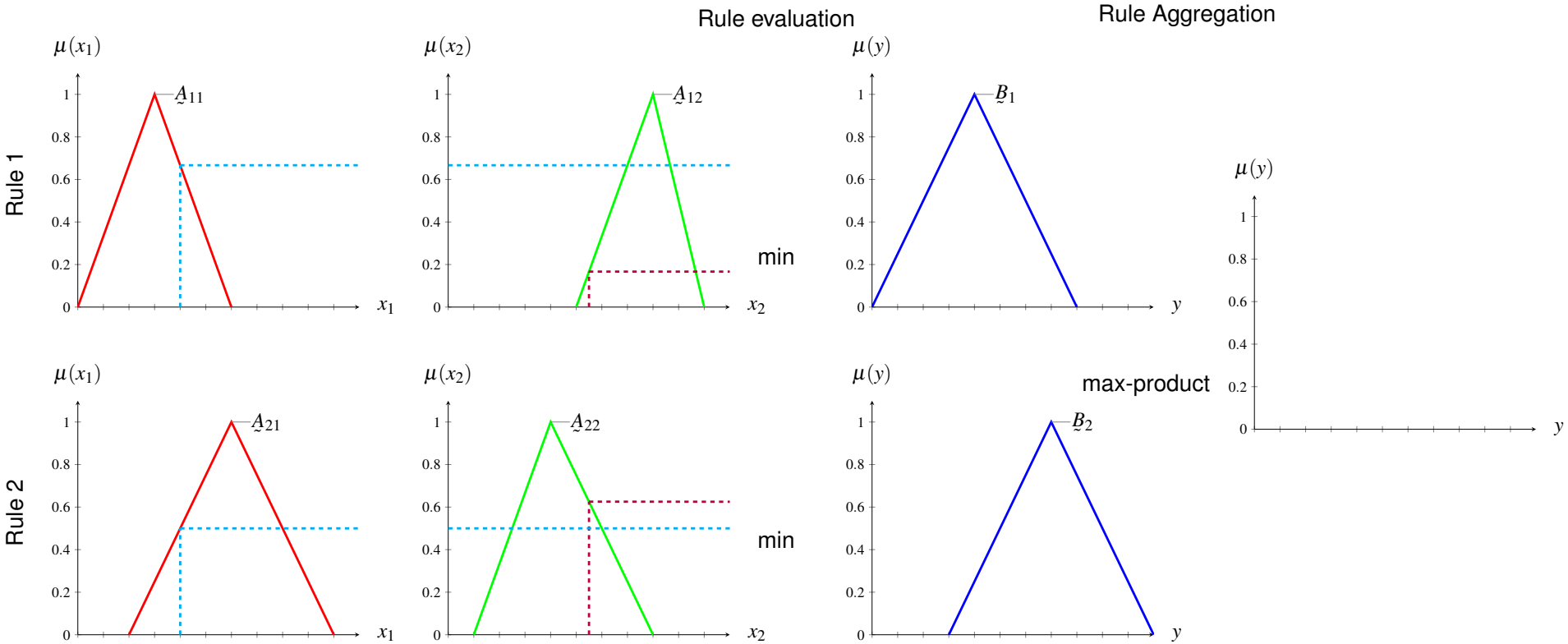


Figure 22: Mamdani (max-product) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Fuzzy Inference Engine

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-product

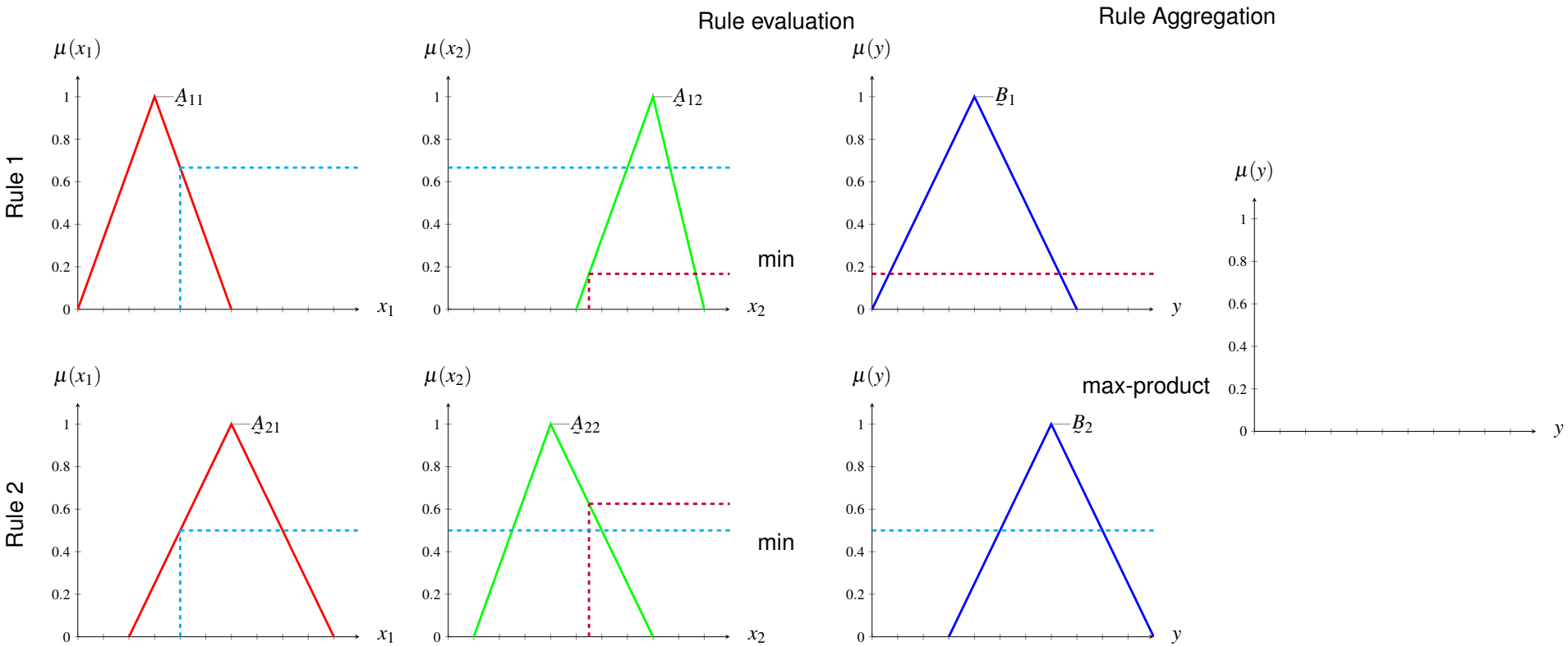


Figure 22: Mamdani (max-product) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Fuzzy Inference Engine

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-product

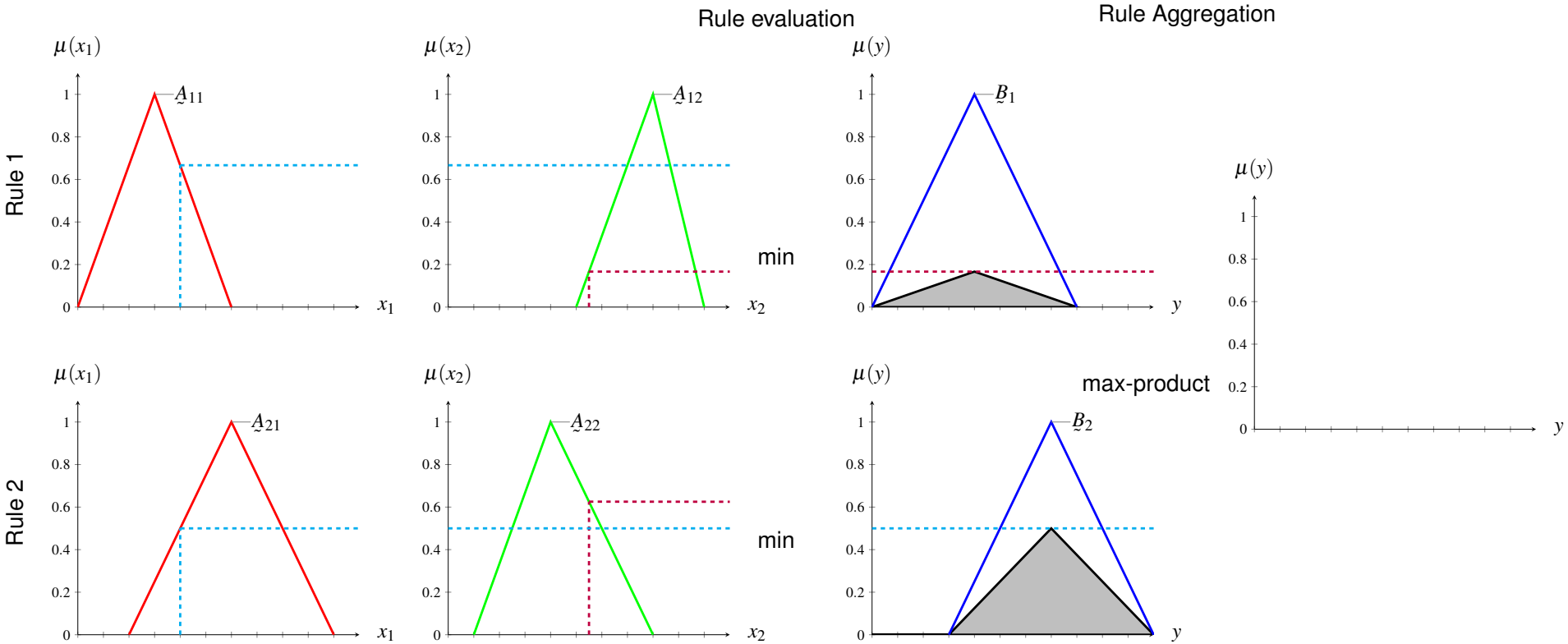


Figure 22: Mamdani (max-product) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Fuzzy Inference Engine

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-product

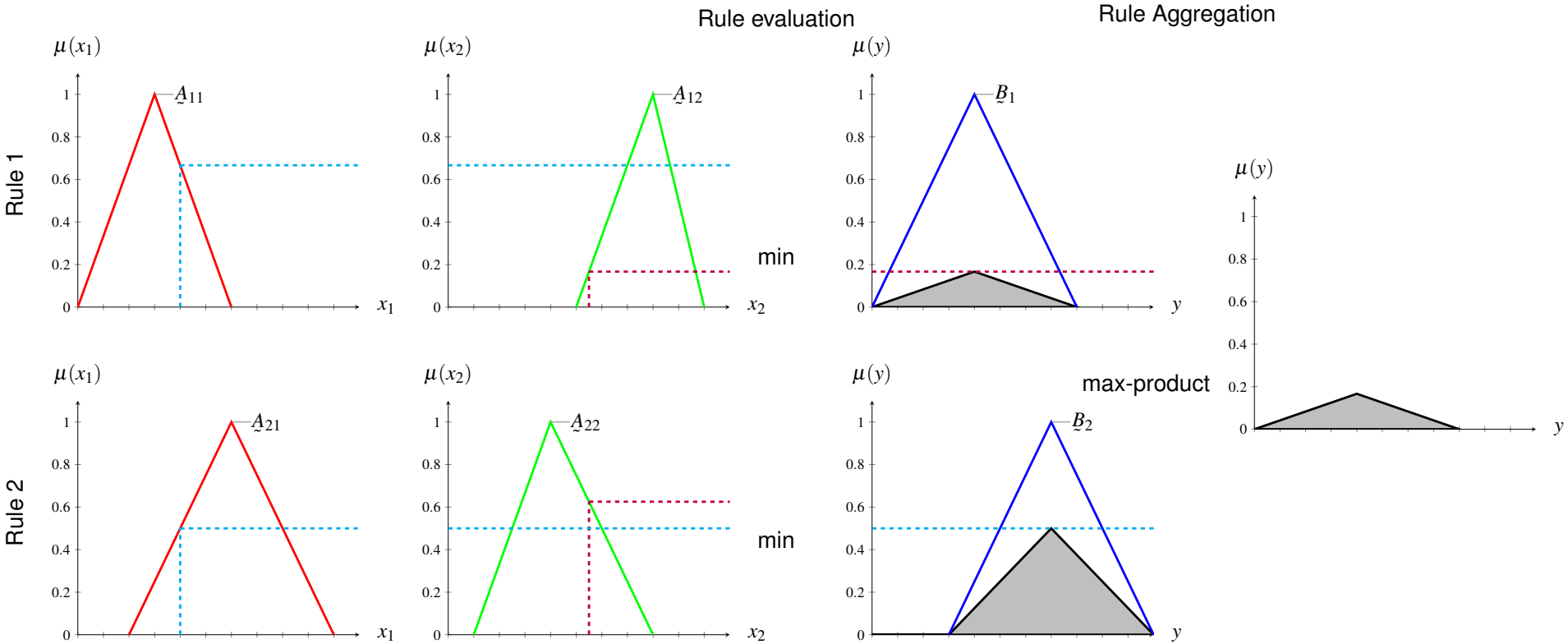


Figure 22: Mamdani (max-product) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-product

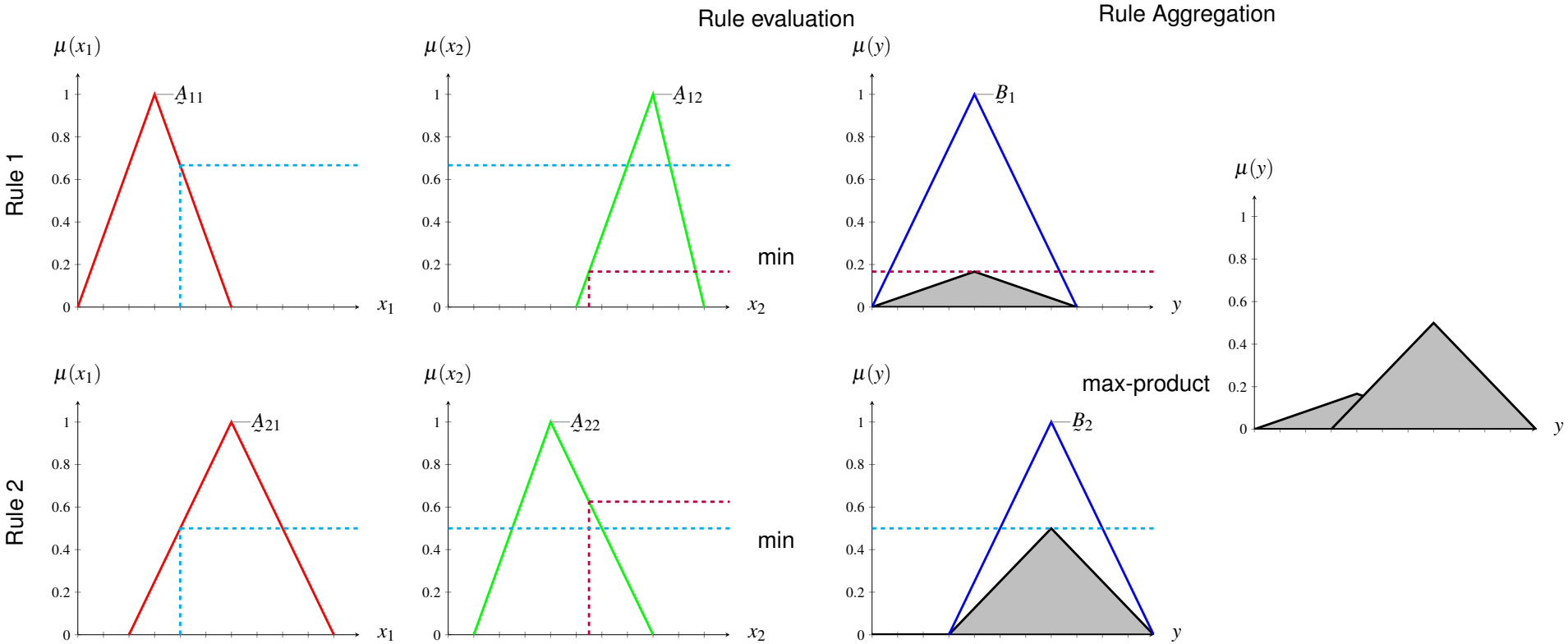


Figure 22: Mamdani (max-product) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Fuzzy Inference Engine

Rule Aggregation: Consider two simple rules:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$(Small) and x_2 is $\underset{\sim}{A}_{12}$(Large) **THEN** y is $\underset{\sim}{B}_1$(Negative)

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$(Large) and x_2 is $\underset{\sim}{A}_{22}$(Small) **THEN** y is $\underset{\sim}{B}_2$(Positive)

Fuzzy AND operator: min; Inference method: max-product

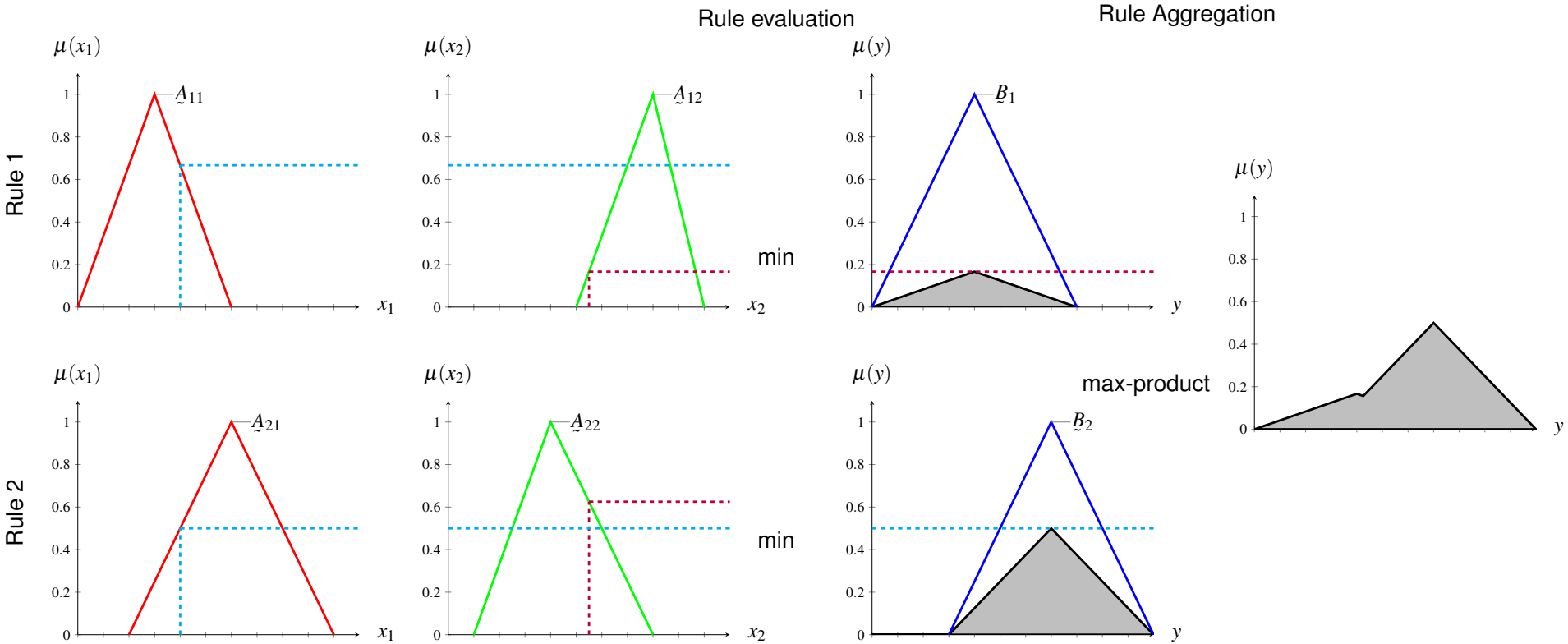
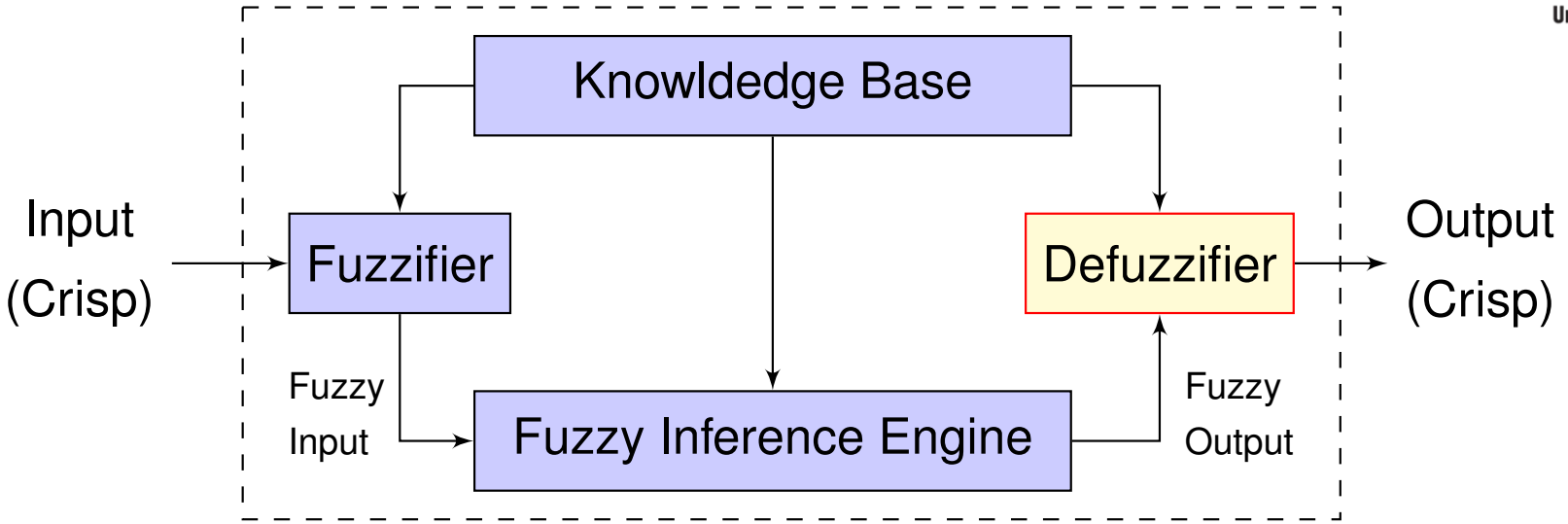
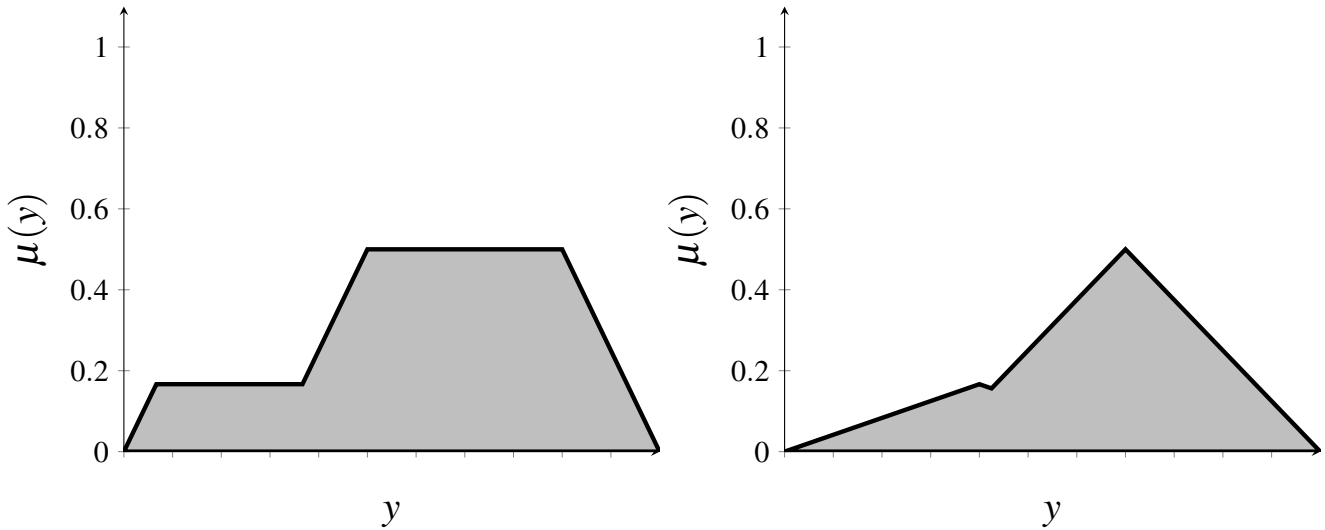


Figure 22: Mamdani (max-product) inference method with crisp inputs. **Grey regions: inferred fuzzy sets**.

Defuzzification



Fuzzy Output: Examples



Defuzzification

- Defuzzification is a process to convert the fuzzy output (an inferred membership function) to a crisp value.
- There are a number of methods available for defuzzification, e.g.,
  - *max membership principle*,
  - *centroid method*,
  - *weighted average method*,
  - *mean max membership*,
  - *center of sums*,
  - *center of largest area*,
  - *first (or last) of maxima*.

1. Max Membership Principle

- Also known as the *height method*.
- It is limited to peaked output functions.
- $\mu_{\tilde{C}}(z^*) \geq \mu_{\tilde{C}}(z) \forall z \in Z$ where z^* is the defuzzified value.

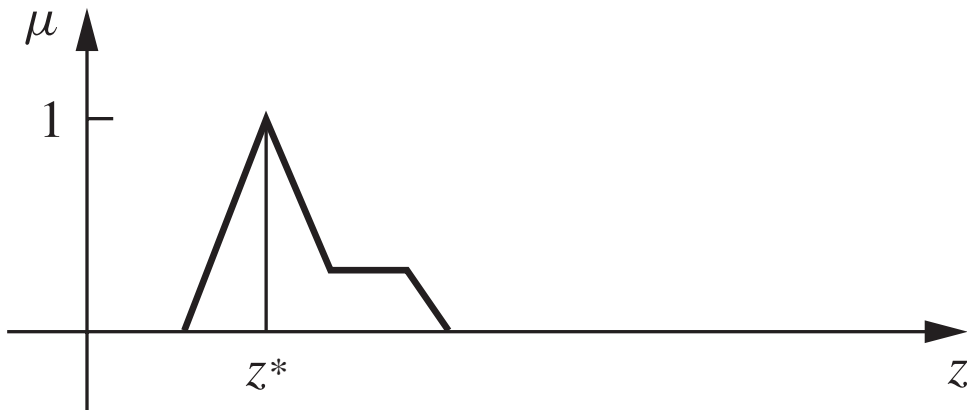


Figure 23: Max membership defuzzification method.

2. Centroid Method

- Also known as the *center of area* (COA) or *center of gravity* (COG).
- *Continuous form:* $z^* = \frac{\int \mu_{\tilde{C}}(z) z dz}{\int \mu_{\tilde{C}}(z) dz}$ where $\int$ denotes an algebraic integration.
- *Discrete form:* $z^* = \frac{\sum_{z_i \in Z} \mu_{\tilde{C}}(z_i) z_i}{\sum_{z_i \in Z} \mu_{\tilde{C}}(z_i)}$ where $\sum$ denotes an algebraic sum.

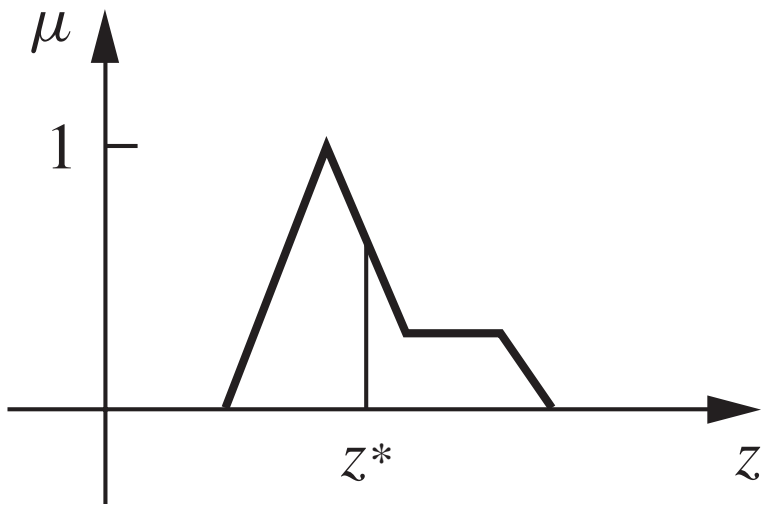


Figure 24: Centroid defuzzification method.

3. Weighted Average Method

- It is computational efficient, however, symmetrical output membership functions are required.
- $z^* = \frac{\sum \mu_{\tilde{C}}(\bar{z})\bar{z}}{\sum \mu_{\tilde{C}}(\bar{z})}$ where $\sum$ denotes an algebraic sum and $\bar{z}$ is the centroid of each symmetric inferred membership function.

Example: $z^* = \frac{0.5 \times a + 0.9 \times b}{0.5 + 0.9}$

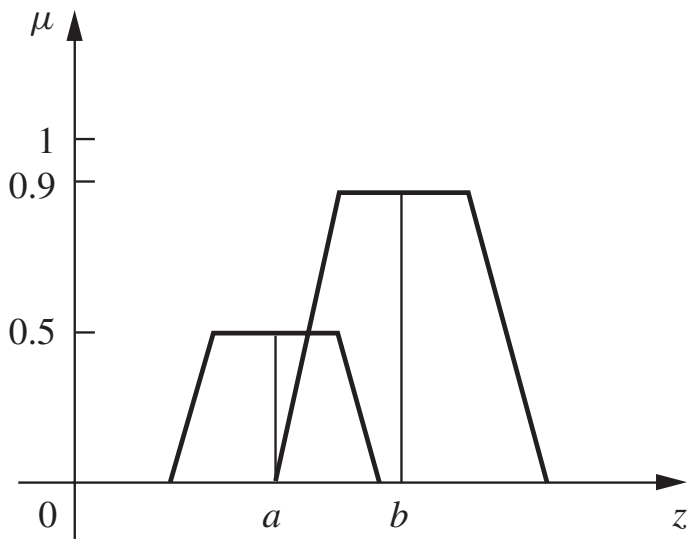


Figure 25: Weighted average defuzzification method.

4. Mean Max Membership

- Also known as *middle-of-maxima*.
- It is computational efficient.
- $z^* = \frac{a+b}{2}$.

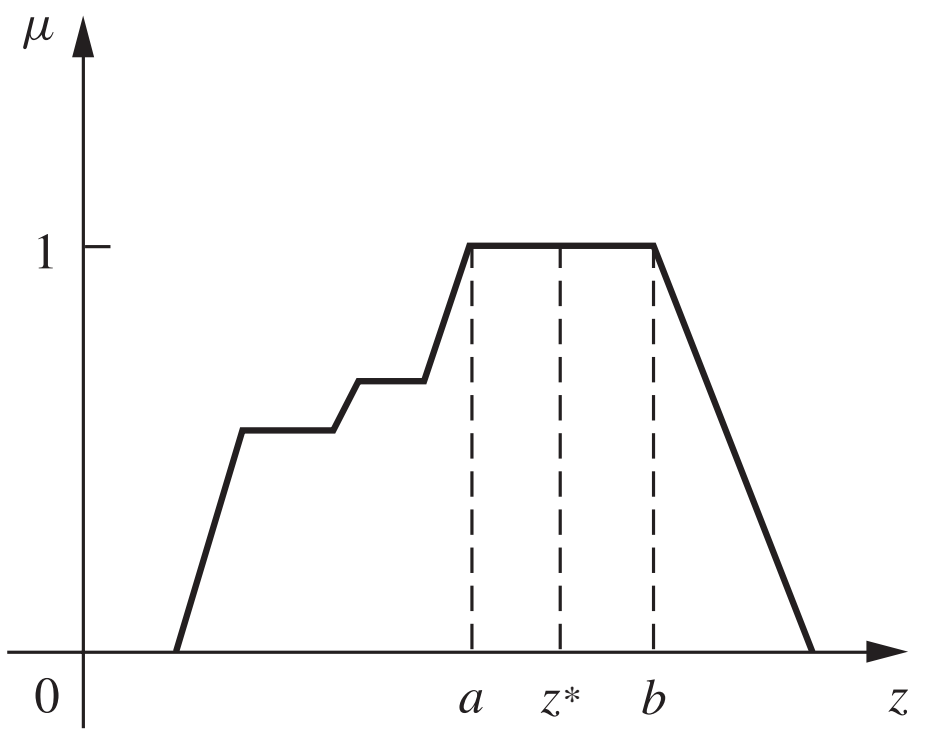


Figure 26: Mean max membership defuzzification method.

5. Center of Sums

- Faster than many methods. Not restricted to symmetric membership functions.
- This method finds the centroid of the individual output membership functions. The intersecting areas are included twice (*drawback*).
- *Continuous form:* $z^* = \frac{\sum_{k=1}^{n} \int \mu_{\tilde{C}_k}(z)\bar{z}_k dz}{\sum_{k=1}^{n} \int \mu_{\tilde{C}_k}(z) dz}$ where $\int$ denotes an algebraic integration, $\bar{z}_k$ is the centroid distance of the k^{th} inferred output membership functions.
- *Discrete form:* $z^* = \frac{\sum_{k=1}^{n} \sum_{z_i \in Z} \mu_{\tilde{C}_k}(z_i)\bar{z}_k}{\sum_{k=1}^{n} \sum_{z_i \in Z} \mu_{\tilde{C}_k}(z_i)}$ where $\sum$ denotes an algebraic sum.

Example: $z^* = \frac{4\times\frac{(4+8)\times 0.5}{2} + 8\times\frac{4\times 1}{2}}{\frac{(4+8)\times 0.5}{2} + \frac{4\times 1}{2}} = 5.6$

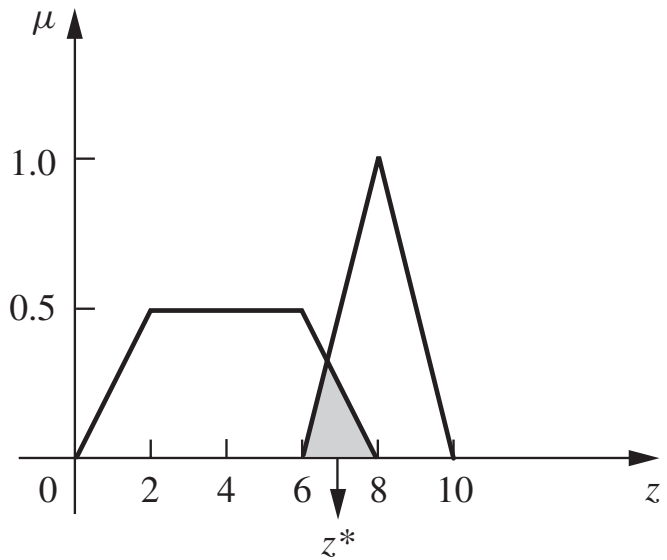


Figure 27: Center of sums defuzzification method.

6. Center of Largest Area

- It is the center of gravity method but the centroid is computed for the largest convex sub-region.
- *Continuous form:* $z^* = \dfrac{\int \mu_{\tilde{C}_m}(z) z dz}{\int \mu_{\tilde{C}_m}(z) dz}$ where $\int$ denotes an algebraic integration, $\tilde{C}_m$ is the largest convex sub-region of the inferred output membership functions.
- *Discrete form:* $z^* = \dfrac{\sum_{z_i \in Z} \mu_{\tilde{C}_m}(z_i) z_i}{\sum_{z_i \in Z} \mu_{\tilde{C}_m}(z_i)}$ where $\sum$ denotes an algebraic sum.

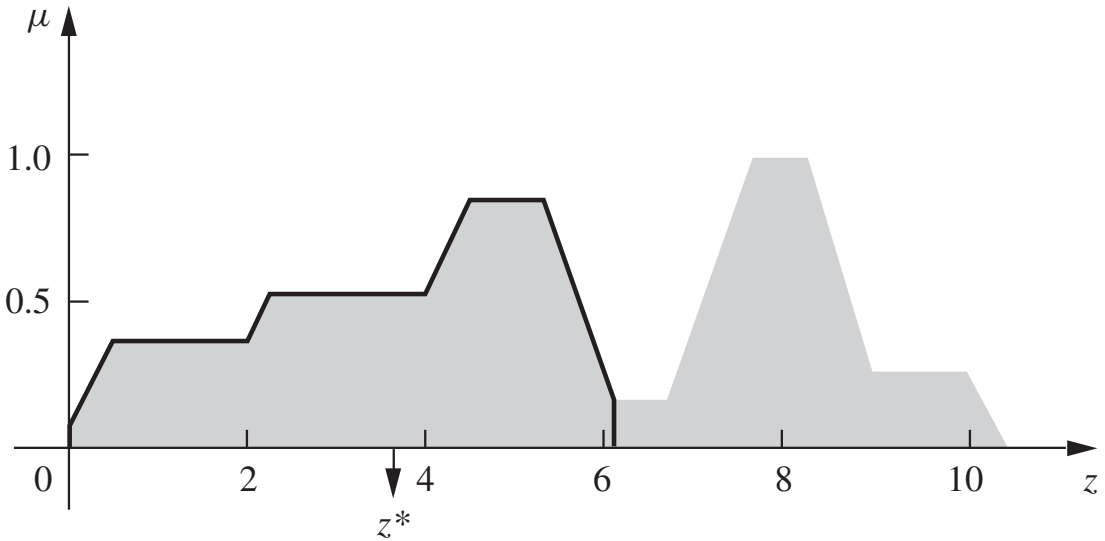


Figure 28: Center of largest area defuzzification method.

7. First (or last) of Maxima

- *The first of the maxima:* $z^* = \inf_{z \in Z} \{z \in Z | \mu_{\tilde{C}}(z) = \text{hgt}(\mu_{\tilde{C}})\}$.
- *The last of maxima:* $z^* = \sup_{z \in Z} \{z \in Z | \mu_{\tilde{C}}(z) = \text{hgt}(\mu_{\tilde{C}})\}$

  where *inf* and *sup* stand for *infimum* and *supremum*, respectively.

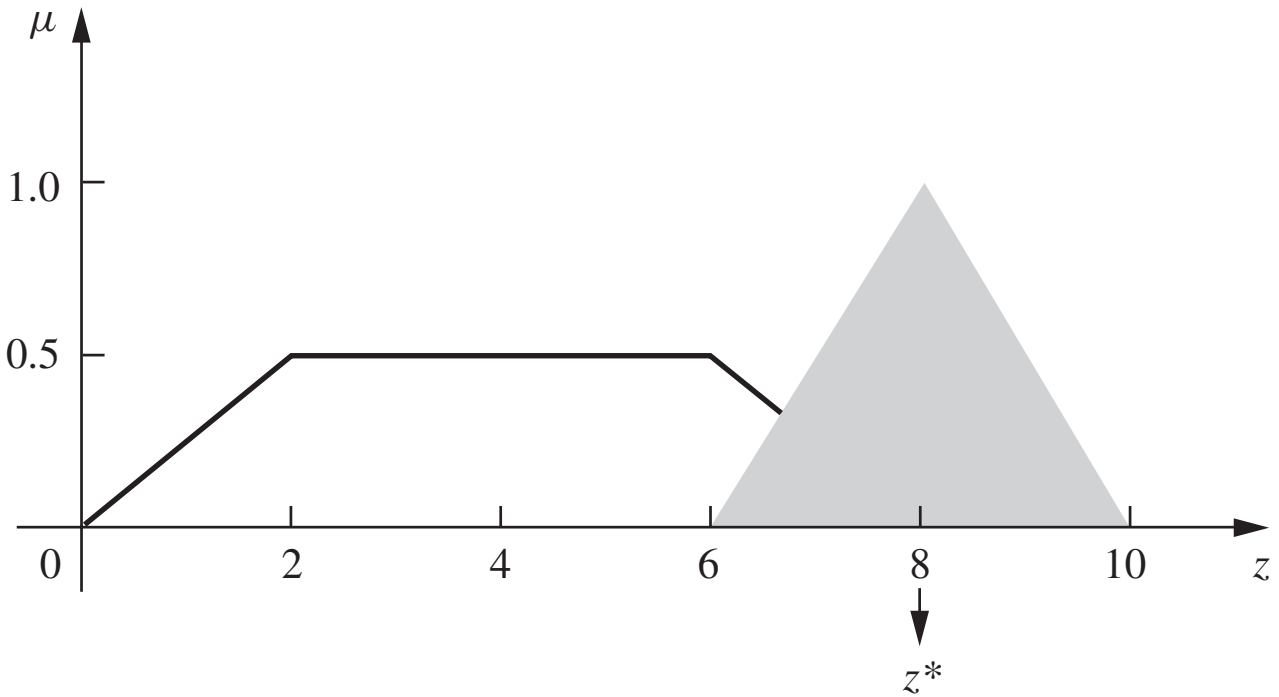


Figure 29: First (or last) of maxima defuzzification method.

Three Fuzzy Inference Systems

Three Fuzzy Inference Systems

- Three common fuzzy inference systems:
  - Mamdani fuzzy inference systems
  - Sugeno fuzzy inference systems (also known as Sugeno fuzzy models, TSK (Takagi, Sugeno, and Kang) fuzzy models)
  - Tsukamoto fuzzy inference systems (also known as Tsukamoto fuzzy models)
- The main difference is in the consequents of the IF-THEN rules
  - Mamdani FIS: Consequent membership function is a general membership function
  - Sugeno FIS: Consequent membership function is a mathematical function
  - Tsukamoto FIS: Consequent membership function is a monotonic membership function (a shoulder function)

1. Mamdani fuzzy inference systems

General rule format:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$ and/or x_2 is $\underset{\sim}{A}_{12}$ and/or $\cdots$ **THEN** y is $\underset{\sim}{B}_1$

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$ and/or x_2 is $\underset{\sim}{A}_{22}$ and/or $\cdots$ **THEN** y is $\underset{\sim}{B}_2$

$\vdots$

Rule r: **IF** x_1 is $\underset{\sim}{A}_{r1}$ and/or x_2 is $\underset{\sim}{A}_{r2}$ and/or $\cdots$ **THEN** y is $\underset{\sim}{B}_r$

- Each consequent is a membership function.
- Rule evaluation and defuzzification are done using any of the introduced methods.

2. Sugeno fuzzy inference systems

General rule format:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$ and/or x_2 is $\underset{\sim}{A}_{12}$ and/or $\cdots$ **THEN** y is $f_1(x_1, x_2, \cdots)$

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$ and/or x_2 is $\underset{\sim}{A}_{22}$ and/or $\cdots$ **THEN** y is $f_2(x_1, x_2, \cdots)$

$\vdots$

Rule r: **IF** x_1 is $\underset{\sim}{A}_{r1}$ and/or x_2 is $\underset{\sim}{A}_{r2}$ and/or $\cdots$ **THEN** y is $f_r(x_1, x_2, \cdots)$

2. Sugeno fuzzy inference systems

- Each consequent is a function, $f_i(x_1, x_2, \cdots)$, so, each rule has a crisp output.
- When $f_i(x_1, x_2, \cdots)$ is a constant, the Sugeno fuzzy inference system is reduced to Mamdani fuzzy inference system with output membership functions as singletons.
- Rule evaluation is done using any of the introduced methods.
- Defuzzification is obtained by *weighted average* of all functions (Weighted average defuzzification), i.e.,

$$y = \frac{w_1(x_1, x_2, \cdots) f_1(x_1, x_2, \cdots) + w_2(x_1, x_2, \cdots) f_2(x_1, x_2, \cdots) + \cdots + w_r(x_1, x_2, \cdots) f_r(x_1, x_2, \cdots)}{w_1(x_1, x_2, \cdots) + w_2(x_1, x_2, \cdots) + \cdots + w_r(x_1, x_2, \cdots)}$$
$$= \frac{\sum_{i=1}^{r} w_i(x_1, x_2, \cdots) f_i(x_1, x_2, \cdots)}{\sum_{i=1}^{r} w_i(x_1, x_2, \cdots)}$$

where $w_i(x_1, x_2, \cdots) = \mu_{\tilde{A}_{i1} \cap \tilde{A}_{i2} \cap \cdots}(x_1, x_2, \cdots)$, $i = 1, 2, \ldots, r$.

2. Sugeno fuzzy inference systems

Consider a two-rule Sugeno fuzzy model:

Rule 1: **IF** x is $\tilde{A}_1$ and y is $\tilde{B}_1$ **THEN** z is $f_1(x,y)$

Rule 2: **IF** x is $\tilde{A}_2$ and y is $\tilde{B}_2$ **THEN** z is $f_2(x,y)$

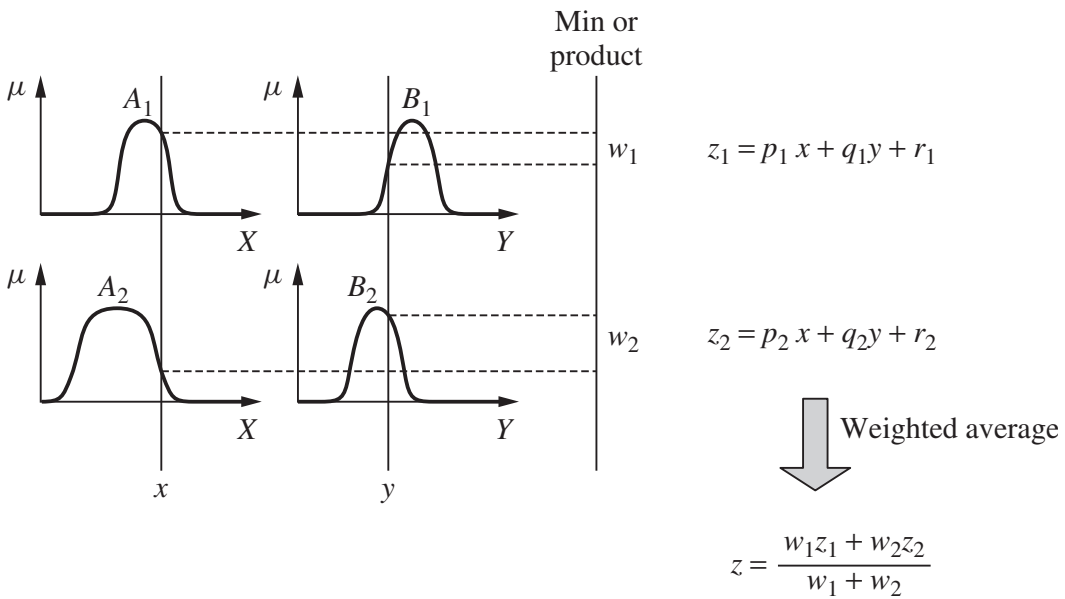


Figure 30: Weighted average defuzzification method for Sugeno fuzzy model.

$w_i(x,y) = \mu_{\tilde{A}_i \cap \tilde{B}_i}(x,y)$, i = 1, 2.

min: $w_i(x,y) = \min(\mu_{\tilde{A}_i}(x), \mu_{\tilde{B}_i}(y))$; product: $w_i(x,y) = \mu_{\tilde{A}_i}(x) \times \mu_{\tilde{B}_i}(y)$

2. Sugeno fuzzy inference systems

Example: An example of 2-input single-output Sugeno fuzzy model with 4 rules:

Rule 1: **IF** x is *Small* and y is *Small* **THEN** z is $-x+y+1$

Rule 2: **IF** x is *Small* and y is *Large* **THEN** z is $-y+3$

Rule 3: **IF** x is *Large* and y is *Small* **THEN** z is $-x+3$

Rule 4: **IF** x is *Large* and y is *Large* **THEN** z is $x+y+2$

$$z = \frac{w_1(-x+y+1)+w_2(-y+3)+w_3(-x+3)+w_4(x+y+2)}{w_1+w_2+w_3+w_4}$$

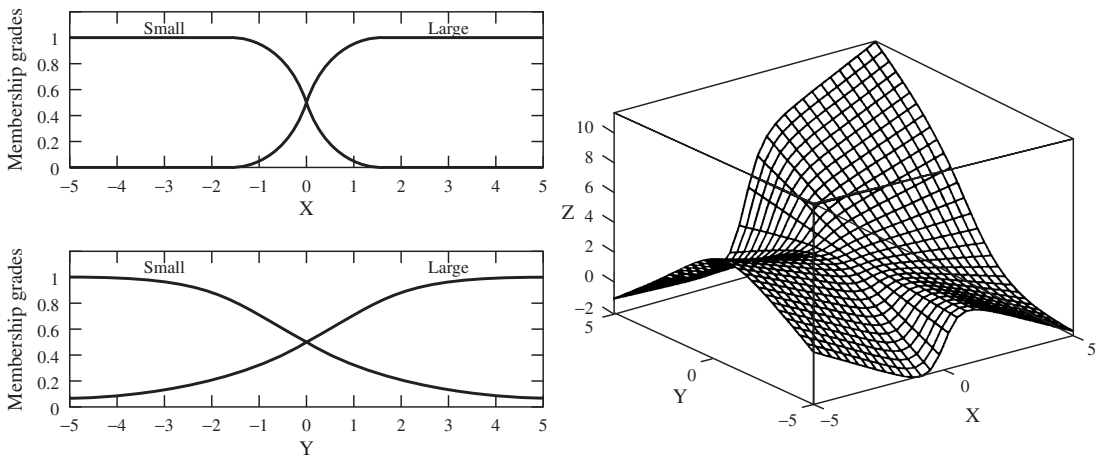


Figure 31: 2-input, single-output Sugeno fuzzy model with 4 rules. (a) Antecedent and consequent membership functions. (b) Overall output surface.

min: $w_1(x,y) = \min(\mu_{x_{Small}}(x), \mu_{y_{Small}}(y))$; $w_2(x,y) = \min(\mu_{x_{Small}}(x), \mu_{y_{Large}}(y))$; $w_3(x,y) = \min(\mu_{x_{Large}}(x), \mu_{y_{Small}}(y))$;
$w_4(x,y) = \min(\mu_{x_{Large}}(x), \mu_{y_{Large}}(y))$

product: $w_1(x,y) = \mu_{x_{Small}}(x) \times \mu_{y_{Small}}(y)$; $w_2(x,y) = \mu_{x_{Small}}(x) \times \mu_{y_{Large}}(y)$; $w_3(x,y) = \mu_{x_{Large}}(x) \times \mu_{y_{Small}}(y)$;
$w_4(x,y) = \mu_{x_{Large}}(x) \times \mu_{y_{Large}}(y)$

3. Tsukamoto fuzzy inference systems

General rule format:

Rule 1: **IF** x_1 is $\underset{\sim}{A}_{11}$ and/or x_2 is $\underset{\sim}{A}_{12}$ and/or $\cdots$ **THEN** y is $\underset{\sim}{C}_1$

Rule 2: **IF** x_1 is $\underset{\sim}{A}_{21}$ and/or x_2 is $\underset{\sim}{A}_{22}$ and/or $\cdots$ **THEN** y is $\underset{\sim}{C}_2$

$\vdots$

Rule r: **IF** x_1 is $\underset{\sim}{A}_{r1}$ and/or x_2 is $\underset{\sim}{A}_{r2}$ and/or $\cdots$ **THEN** y is $\underset{\sim}{C}_r$

3. Tsukamoto fuzzy inference systems

Consider a two-rule Tsukamoto fuzzy model:

Rule 1: **IF** x is $\underset{\sim}{A}_1$ and y is $\underset{\sim}{B}_1$ **THEN** z is $\underset{\sim}{C}_1$

Rule 2: **IF** x is $\underset{\sim}{A}_2$ and y is $\underset{\sim}{B}_2$ **THEN** z is $\underset{\sim}{C}_2$

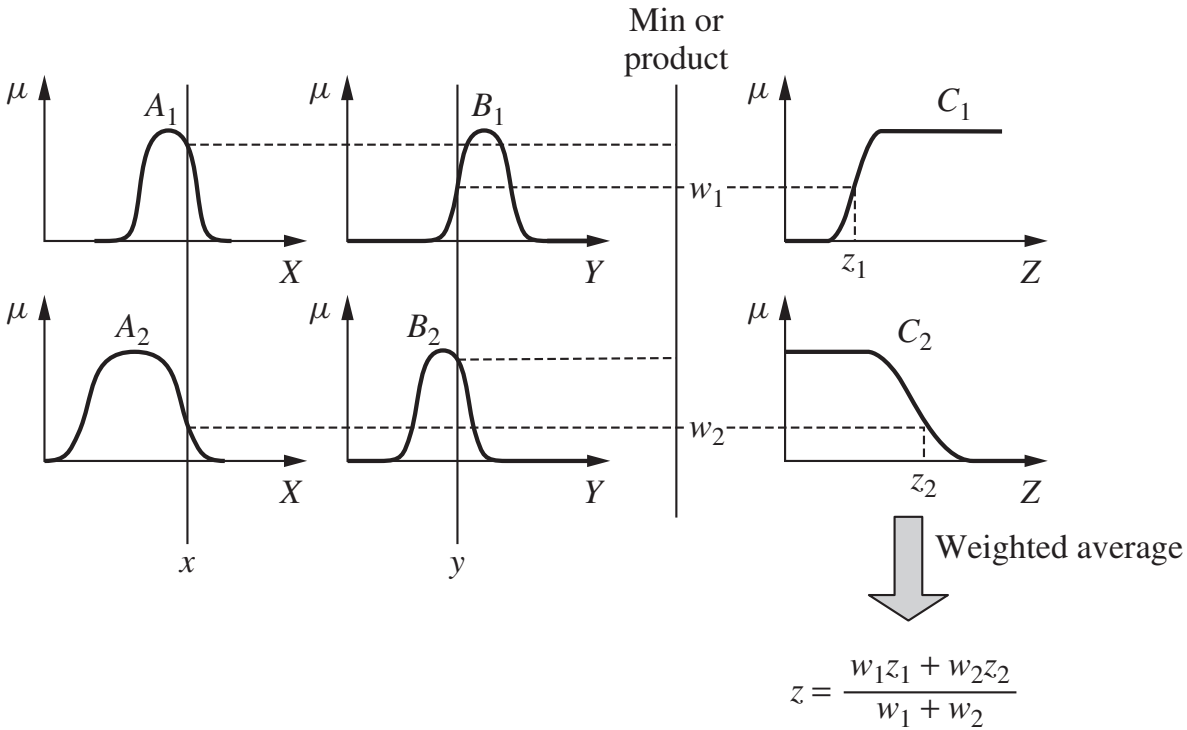


Figure 32: Weighted average defuzzification method for Tsukamoto fuzzy model.

3. Tsukamoto fuzzy inference systems

Example: An example of single-input single-output Tsukamoto fuzzy model with 3 rules:

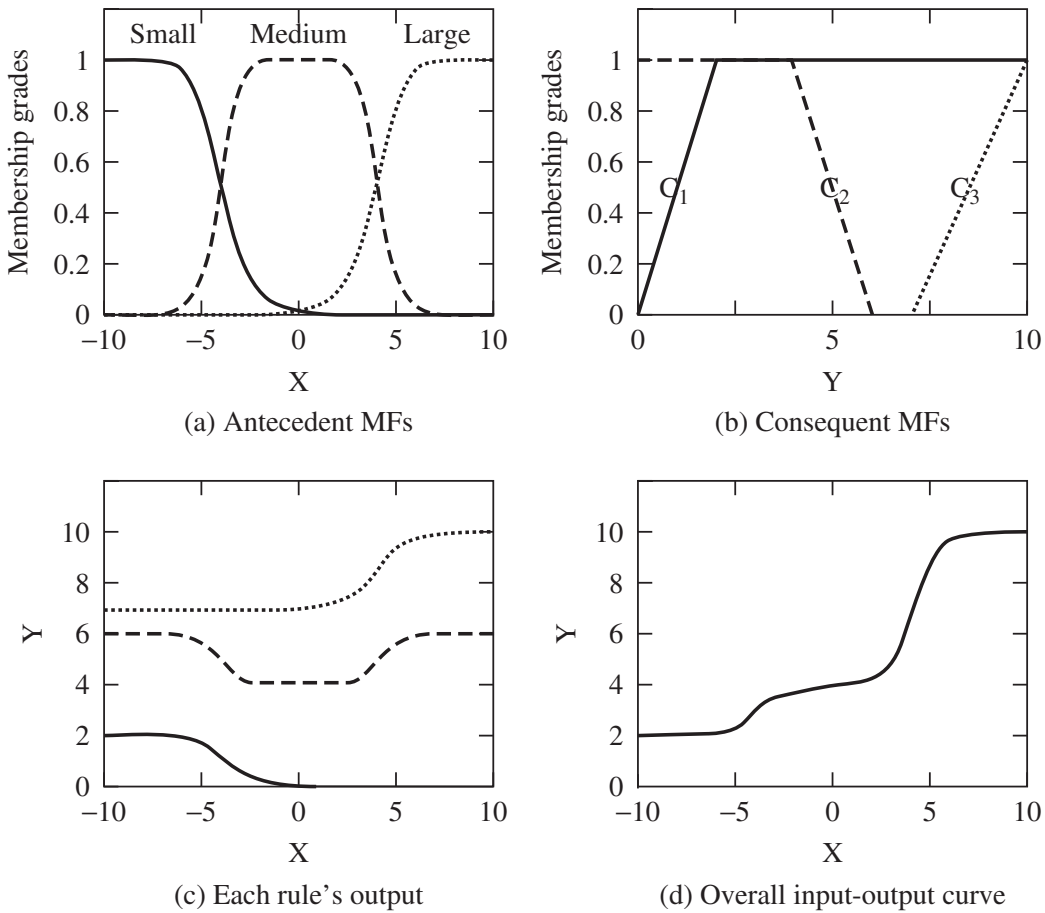
Rule 1: **IF** X is $Small$ **THEN** Y is C_1

Rule 2: **IF** X is $Medium$ **THEN** Y is C_2

Rule 3: **IF** X is $Large$ **THEN** Y is C_3

Inferred Output:

$$Y = \frac{\mu_{\text{Small}}(X)C_1(X)+\mu_{\text{Medium}}(X)C_2(X)+\mu_{\text{Large}}(X)C_3(X)}{\mu_{\text{Small}}(X)+\mu_{\text{Medium}}(X)+\mu_{\text{Large}}(X)}$$



(a) Antecedent MFs (b) Consequent MFs

(c) Each rule's output (d) Overall input-output curve

Figure 33: Single-input, single-output Tsukamoto fuzzy model with 3 rules.

Fuzzy Inference System and its Learning

Fuzzy Inference System and its Learning

1. Understand the problem and formulate the problem as an optimisation problem that help define the cost/fitness/objective function
2. Define the FIS, e.g., number of inputs and outputs, number of rules, AND/OR operations, input/output membership functions
3. Define the decisions variables, e.g., the parameters of the membership functions, the coefficients of the functions in the consequents, to be learnt so that to optimise the cost/fitness/objective function
4. Choose a suitable learning algorithm (numerical optimisation, nature-inspired learning algorithms) for learning