ECE 657 (problem 1 -3)

Meng Tianao.

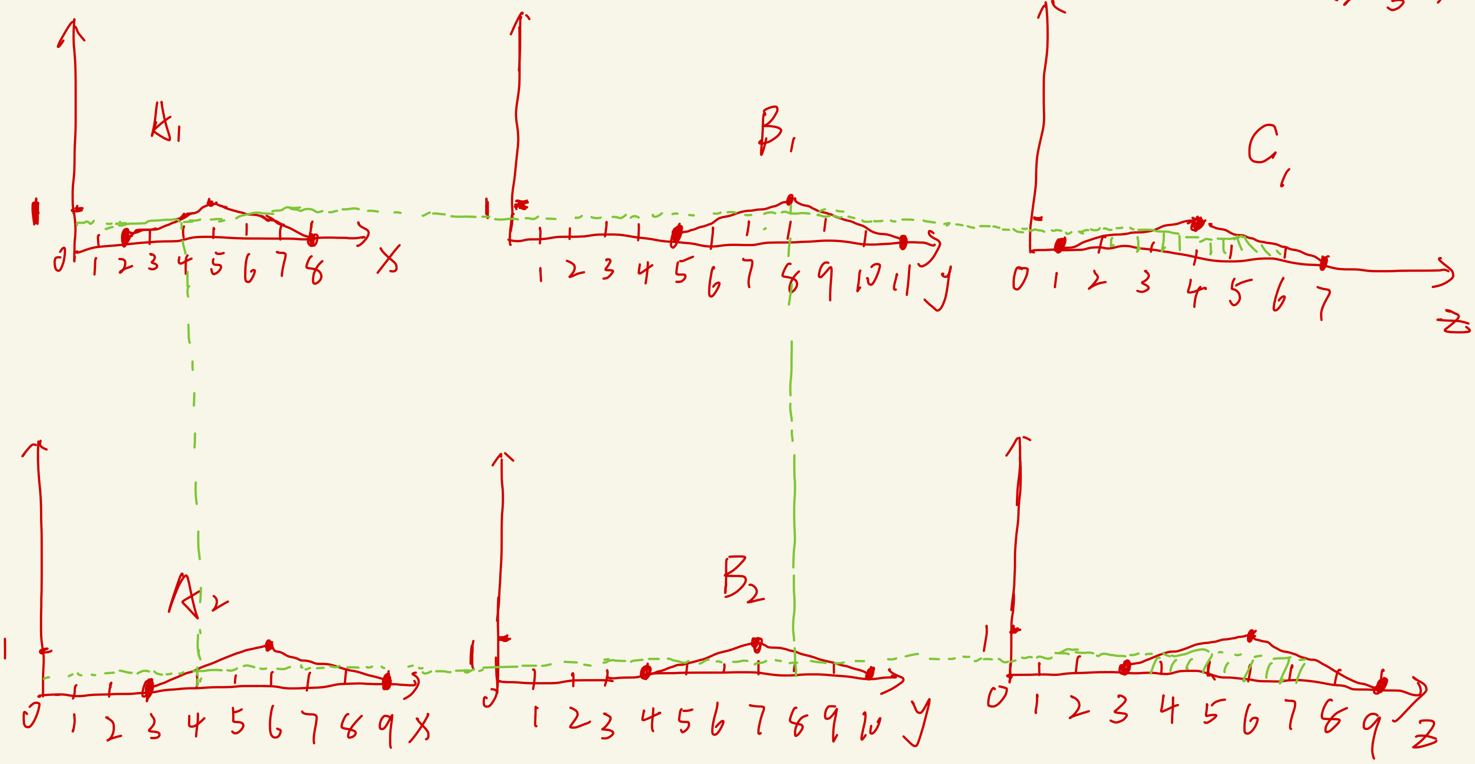
20810553

1、 Mamdani inferencing system : max- min operator for aggregation part.

Rule 1: if x is A_1 and y is B_1 , then z is C_1

Rule 2: if x is A_2 and y is B_2 , then z is C_2

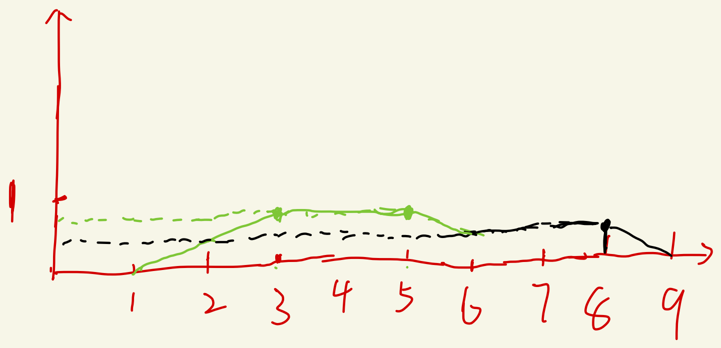
when $x_0(t_1) = 4$ $y_0(t_1) = 8$, then $\mu_{A_1} = \frac{2}{3}$, $\mu_{B_1} = 1$, $\mu_{A_2} = \frac{1}{3}$, $\mu_{B_2} = \frac{2}{3}$



when $\mu_{C_1} = \frac{2}{3}$, then $z_1 = 3$, $z_2 = 5$

when $\mu_{C_2} = \frac{1}{3}$, then $z_1 = 4$ $z_2 = 8$

Then we get inference membership function.



when we use MOM defuzzification strategy , the value of the control output is $\frac{3+5}{1+1} = 4$

when we use LOM , the value of the control output is 5.

2.(a) Based on the description of this question, we use proposition calculus approach to get the if A then B relationship.

$$R:\ A \to B \qquad \bar{A} \vee (A \wedge B)$$

$$x\text{-gyro bias} = A = \left\{ \frac{0.2}{1.7} + \frac{0.4}{1.8} + \frac{0.6}{1.9} + \frac{0.8}{2.0} + \frac{0.6}{2.1} + \frac{0.4}{2.2} + \frac{0.2}{2.3} \right\}$$

$$\text{accelerator bias} = B = \left\{ \frac{0.1}{0.25} + \frac{0.4}{0.27} + \frac{0.9}{0.3} + \frac{0.4}{0.33} + \frac{0.1}{0.35} \right\}$$

$$N_{A\wedge B}(a_i, b_i) = \begin{array}{c} \\ \\ \end{array} \begin{array}{cccccc} & & b_i & & & \\ 0.25 & 0.27 & 0.3 & 0.33 & 0.35 & \\ 0.1 & 0.2 & 0.2 & 0.2 & 0.1 & 1.7 \\ 0.1 & 0.4 & 0.4 & 0.4 & 0.1 & 1.8 \\ 0.1 & 0.4 & 0.6 & 0.4 & 0.1 & 1.9 \\ 0.1 & 0.4 & 0.8 & 0.4 & 0.1 & 2.0 \quad a_i \\ 0.1 & 0.4 & 0.6 & 0.4 & 0.1 & 2.1 \\ 0.1 & 0.4 & 0.4 & 0.4 & 0.1 & 2.2 \\ 0.1 & 0.2 & 0.2 & 0.2 & 0.1 & 2.3 \end{array}$$

$$\bar{A} = \left\{ \frac{0.8}{1.7} + \frac{0.6}{1.8} + \frac{0.4}{1.9} + \frac{0.2}{2.0} + \frac{0.4}{2.1} + \frac{0.6}{2.2} + \frac{0.8}{2.3} \right\}$$

$$N_{\bar{A}}(a_i) = \begin{bmatrix} 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.4 & 0.4 & 0.4 & 0.4 & 0.4 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.4 & 0.4 & 0.4 & 0.4 & 0.4 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix}$$

Then we can get
$$N_R = \begin{bmatrix} 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix}$$

(b) (i) Max- min composition $T = A' \circ R$

$$N_R = \begin{bmatrix} 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix}$$

$$N_A = \begin{bmatrix} 0 \\ 0.5 \\ 0.7 \\ 0.95 \\ 0.7 \\ 0.5 \\ 0 \end{bmatrix}$$

$$N_T = \max_{\text{rows}} \left(\min_{\text{columns}} \left(\begin{bmatrix} 0 \\ 0.5 \\ 0.7 \\ 0.95 \\ 0.7 \\ 0.5 \\ 0 \end{bmatrix} \begin{bmatrix} 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix} \right) \right)$$

$$= \max_{\text{rows}} \left(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 0.5 & 0.5 & 0.8 & 0.5 & 0.5 \end{bmatrix}$$

(b) ② Max - product composition.

$$N_R = \begin{bmatrix} 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix}$$

$$N_A = \begin{bmatrix} 0 \\ 0.5 \\ 0.7 \\ 0.95 \\ 0.7 \\ 0.5 \\ 0 \end{bmatrix}$$

$$N_T = \max_{\text{row}} \left(\begin{bmatrix} 0 & 0.5 & 0.7 & 0.95 & 0.7 & 0.5 & 0 \end{bmatrix} \circ \begin{bmatrix} 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix} \right)$$

$$= \max_{\text{row}} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\ 0.28 & 0.28 & 0.42 & 0.28 & 0.28 \\ 0.19 & 0.38 & 0.76 & 0.38 & 0.19 \\ 0.28 & 0.28 & 0.42 & 0.28 & 0.28 \\ 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0.3 & 0.38 & 0.76 & 0.38 & 0.3 \end{bmatrix}$$

3. GP works with computer programs whose fitness are determined by their ability to solve a computational problem.

Before start we need two steps

Determination of terminal set

Determination of function set

Determination of fitness measure (e.g. reverse of difference between output and target)

Determination of parameters for GP run. (e.g. population size)

Determination of termination criterion (sum of errors is less than 0.01)

Furthermore, there are two mutation operation in GP.

① A function can only replace a function, and a terminal can only replace a terminal

② An entire sub tree of the parse tree population can replace another subtree.

Then, crossover in GP

① Given two individuals, pick a point in each parse tree independently

② Swap the subtrees at the picked points.

③ repair the parse tree if the associated program is invalid.

From above discussion. Let's now see the difference between GA.

Firstly, I want to say that they share a similar basic structure of an evolutionary algorithm; However, GA always compiles the parameters into binary code (e.g. individuals) while GP always compile individuals into tree structures. Then, variation operators (crossover and mutation) operates on different data structures in GA and GP; therefore, variation operations also operates in different way. There are four types of mutation operators in GA (i.e. binary mutation, uniform mutation, non-uniform mutation and boundary mutation) while there are only two types mutation in GP. Similarly, there are four types of crossover operators in GA (i.e. single-point crossover, multi-point crossover, uniform crossover and heuristic crossover), while GP uses one type of crossver which shares the same concept with single-point crossover). As for applications, GA is always used in optimization problem, while GP is always for computational problem, which you can get idea from two given examples in the last lecture slide of ECE 657.

In a word, the most fundamental difference is that. in GA, represent a solution as a string of numbers, while in GP, solutions are computer programs.