หมายเหตุ ตัวอย่างที่ให้นี้เป็นเพียงส่วนหนึ่งของสิ่งที่สอนในเรื่อง fuzzy logic, genetic algorithm, particle swarm optimization เท่านั้นไม่ได้ครอบคลุมเนื้อหาทั้งหมด

ตัวอย่างที่ 1 สมมุติให้ระบบมีกฎ 2 กฎ โดยที่แต่ละกฎจะมีอินพุต 2 อินพุต และแต่ละอินพุตในแต่ละกฎมีพจน์ภาษาดังรูปที่ 1.1 โดยที่มีกฎดังนี้คือ

กฎ 1: If x_1 is $\boldsymbol{L}_1$ และ x_2 is $\boldsymbol{H}_2$, then y is $\boldsymbol{L}$

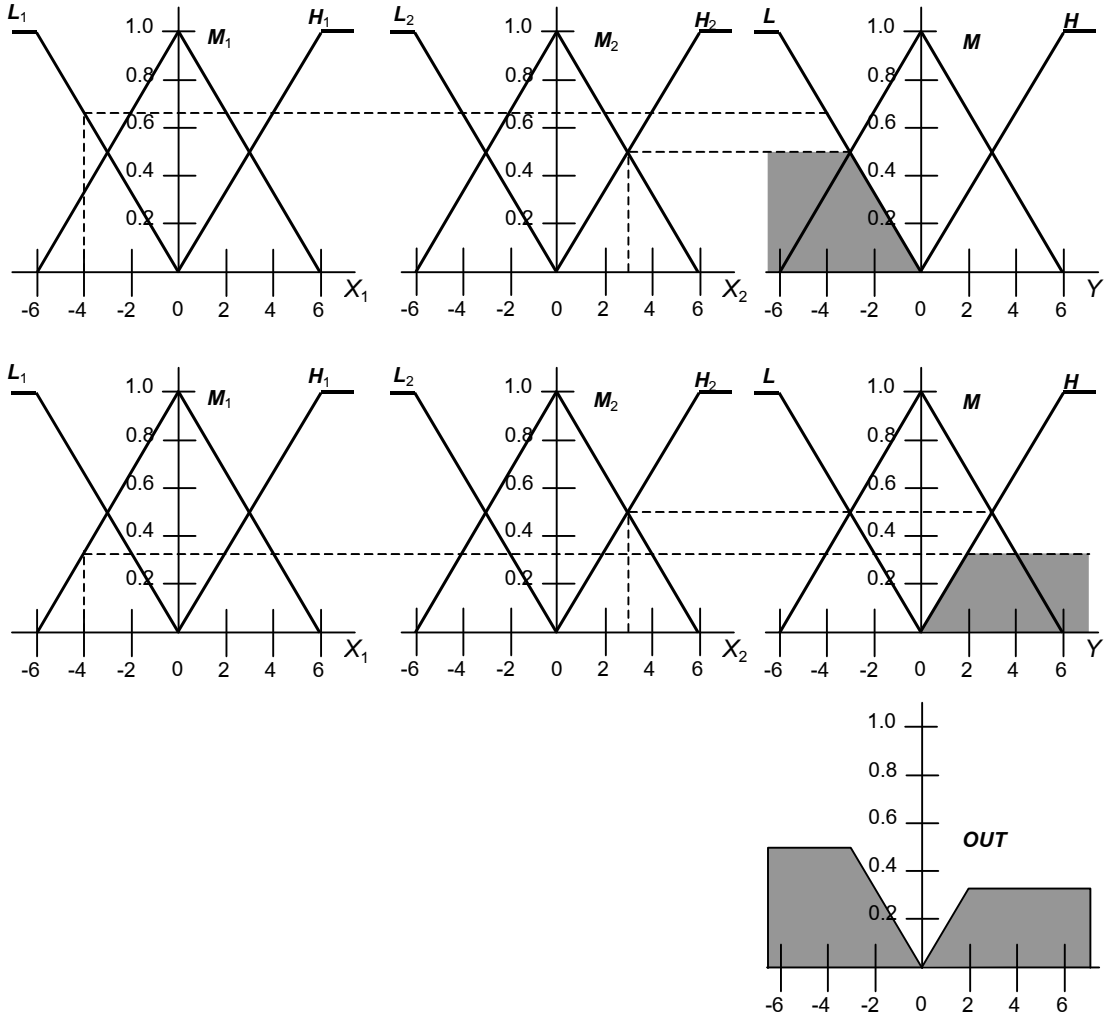
กฎ 2: If x_1 is $\boldsymbol{M}_1$ และ x_2 is $\boldsymbol{M}_2$, then y is $\boldsymbol{H}$

และสมมุติให้อินพุตที่เข้ามาที่ fuzzification interface มีค่าเป็น –4 และ 3 จะเห็นได้ว่า

$$\alpha_1 = \min\big(\boldsymbol{L}_1(-4), \boldsymbol{H}_2(3)\big) = \min(0.67, 0.5) = 0.5$$

และ $$\alpha_2 = \min\big(\boldsymbol{M}_1(-4), \boldsymbol{M}_2(3)\big) = \min(0.33, 0.5) = 0.5$$

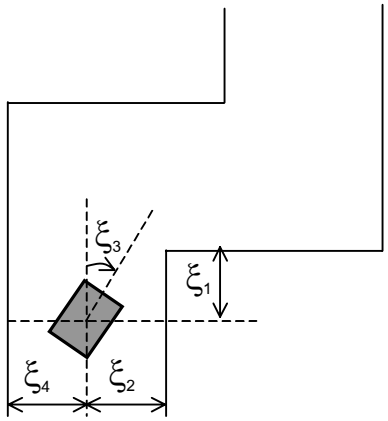
ฟัซซีเอาพุตของกฎที่ 1 และ 2 และฟัซซีเอาพุตรวม (***OUT***) แสดงในรูปที่ 7.14 เช่นกัน สมมุติว่าในส่วนของ (defuzzification interface) ใช้วิธีการแปลงโดยใช้ centroid จะได้ค่า output เท่ากับ 1.2



รูปที่ 1.1 ระบบควบคุมฟัซซีโดยใช้วิธี Mamdani

■

ตัวอย่างที่ 2 ต้องการสร้างระบบที่ควบคุมรถให้ผ่านทางโค้งในขณะที่มีความเร็วคงที่ โดยให้อินพุตมีค่า ξ_1 เป็นระยะของรถจนถึงขอบทางโค้ง ξ_2 เป็นระยะของรถจนถึงไหล่ทางข้างขวา ξ_3 เป็นมุมของรถ และ ξ_4 เป็นระยะของรถถึงไหล่ทางข้างซ้าย ดังแสดงในรูปที่ 2.1



รูปที่ 2.1 อินพุตของระบบควบคุมฟัซซีของการควบคุมรถผ่านทางโค้ง

ตารางที่ 2.1 กฎและค่าคงที่ที่ใช้ในระบบควบคุมฟัซซี

กฎ	ξ_1	ξ_2	ξ_3	ξ_4	p_0	p_1	p_2	p_3	p_4
1	—	—	outwards	small	3.000	0.000	0.000	-0.045	-0.004
2	—	—	forward	small	3.000	0.000	0.000	-0.030	-0.090
3	small	small	outwards	—	3.000	-0.041	0.004	0.000	0.000
4	small	small	forward	—	0.303	-0.026	0.061	-0.050	0.000
5	small	small	inwards	—	0.000	-0.025	0.070	-0.075	0.000
6	small	big	outwards	—	3.000	-0.066	0.000	-0.034	0.000
7	small	big	forward	—	2.990	-0.017	0.000	-0.021	0.000
8	small	big	inwards	—	1.500	0.025	0.000	-0.050	0.000
9	medium	small	outwards	—	3.000	-0.017	0.005	-0.036	0.000
10	medium	small	forward	—	0.053	-0.038	0.080	-0.034	0.000
11	medium	small	inwards	—	-1.220	-0.016	0.047	-0.018	0.000
12	medium	big	outwards	—	3.000	-0.027	0.000	-0.044	0.000
13	medium	big	forward	—	7.000	-0.049	0.000	-0.041	0.000
14	medium	big	inwards	—	4.000	-0.025	0.000	-0.100	0.000
15	big	small	outwards	—	0.370	0.000	0.000	-0.007	0.000
16	big	small	forward	—	-0.900	0.000	0.034	-0.030	0.000
17	big	small	inwards	—	-1.500	0.000	0.005	-0.100	0.000
18	big	big	outwards	—	1.000	0.000	0.000	-0.013	0.000
19	big	big	forward	—	0.000	0.000	0.000	-0.006	0.000
20	big	big	inwards	—	0.000	0.000	0.000	-0.010	0.000

ให้ η ความเร็วของการหมุน (rotation speed) พวงมาลัยและให้เซตสากล $X_1 = [0,\ 150]$ เซนติมิเตอร์ $X_2 = [0,\ 150]$ เซนติมิเตอร์ $X_3 = [-90,\ 90]$ องศา และ $X_4 = [0,\ 150]$ เซนติมิเตอร์ และให้กฎที่ j สำหรับ $1 \leq j \leq 20$ มีลักษณะดังนี้

กฎ j: If ξ_1 is $\boldsymbol{A}_{j1,j}^{(1)}$, และ ξ_2 is $\boldsymbol{A}_{j2,j}^{(2)}$ และ ξ_3 is $\boldsymbol{A}_{j3,j}^{(3)}$ และ ξ_4 is $\boldsymbol{A}_{j4,j}^{(4)}$

$$\text{then } \eta = p_0 + p_1\xi_1 + p_2\xi_2 + p_3\xi_3 + p_4\xi_4$$

โดยที่ $\boldsymbol{A}_{j1,j}^{(1)} \in \{\text{small, medium, big}\}$ $\boldsymbol{A}_{j2,j}^{(2)} \in \{\text{small, big}\}$ $\boldsymbol{A}_{j3,j}^{(3)} \in \{\text{outwards, forward, inwards}\}$ และ $\boldsymbol{A}_{j4,j}^{(4)} \in \{\text{small}\}$ ซึ่งฟัซซีเซตหรือพจน์ภาษาเหล่านี้ถูกกำหนดไว้แล้ว กฎและค่าคงที่ที่ใช้ในการคำนวณแต่ละกฎแสดงในตารางที่ 2.1

สมมุติให้รถกำลังเข้าทางโค้งและวัดอินพุตทั้ง 4 ค่า ได้ $\xi_1 = 10$ เซนติมิเตอร์ $\xi_2 = 30$ เซนติมิเตอร์ $\xi_3 = 0$ องศา (นั่นคือไปตรง) และ $\xi_4 = 50$ เซนติมิเตอร์ เมื่อผ่านค่าเหล่านี้เข้าไปในระบบจะมีแค่กฎที่ 4 และ 7 เท่านั้นที่ค่า α มากกว่า 0 นั่นคือในกฎที่ 4 จากการอ่านค่าความเป็นสมาชิกของ ξ_1 ใน small เท่ากับ 0.8 ξ_2 ใน small เท่ากับ 0.25 ξ_3 ใน forward เท่ากับ 1 ทำให้ได้ $\alpha_4 = 0.25$ และ

$$\eta_4 = 0.303 - 0.026(10) + 0.061(30) - 0.050(0) + 0.000(50) = 1.873$$

และในกฎที่ 7 จากการอ่านค่าความเป็นสมาชิกของ ξ_1 ใน small เท่ากับ 0.8 ξ_2 ใน big เท่ากับ 0.167 ξ_3 ใน forward เท่ากับ 0 ทำให้ได้ $\alpha_4 = 0.167$ และ

$$\eta_7 = 2.990 - 0.017(10) + 0.000(30) - 0.021(0) + 0.000(50) = 2.820$$

และจะได้ค่าเอาพุตรวมเท่ากับ

$$\eta = \frac{0.25(1.873) + 0.167(2.820)}{0.25 + 0.167} = 2.252$$

■

ตัวอย่างที่ 3 Genetic algorithm example

Let us consider a diophantine (only integer solutions) equation: a+2b+3c+4d=30, where a,b,c,d are positive integers. Using a genetic algorithm, all that is needed is a little time to reach a solution (a,b,c,d). Of course you could ask, why not just use a brute force method (plug in every possible value for a,b,c,d given the constraints 1 =< a,b,c,d =< 30)? The architecture of GA systems allow for a solution to be reached quicker since "better" solutions have a better chance of surviving and procreating, as opposed to randomly throwing out solutions and seeing which ones work.

Let's start from the beginning. First we will choose 5 random initial solution sets, with constraints 1 =< a,b,c,d =< 30. (Note that we can choose smaller constraints for b,c,d, but for the sake of simplicity we shall use 30)

Chromosome	(a,b,c,d)
1	(1,28,15,3)

2	(14,9,2,4)
3	(13,5,7,3)
4	(23,8,16,19)
5	(9,13,5,2)

Table 3.1: 1st Generation Chromosomes and Their Contents

To calculate the fitness values, plug each solution set into the expression a+2b+3c+4d. Then, calculate the absolute value of the difference of each expression with 30, this will be our fitness value.

Chromosome	Fitness Value
1	|114-30|=84
2	|54-30|=24
3	|56-30|=26
4	|163-30|=133
5	|58-30|=28

Table 3.2: Fitness Values of 1st Generation Chromosomes (Solution sets)

Since values that are lower are closer to the desired answer (30), these values are more desirable. In this case, higher fitness values are not desirable, while lower ones are. In order to create a system where chromosomes with more desirable fitness values are more likely to be chosen as parents, we must first calculate the percentages that each chromosome has of being picked. One solution is to take the sum of the multiplicative inverses of the fitness values (0.135266), and calculate the percentages from there. (Note: ALL simulations were created using a random number generator)

Chromosome	Likelihood
1	(1/84)/0.135266 = 8.80%
2	(1/24)/0.135266 = 30.8%
3	(1/26)/0.135266 = 28.4%
4	(1/133)/0.135266 = 5.56%
5	(1/28)/0.135266 = 26.4%

Table 3.3: Parent Selection by Percentages

In order to pick our 5 pairs of parents (each of which will have 1 offspring, and thus 5 new solutions sets total), imagine that we had a 10000 sided die, and on 880 of those sides, chromosome 1 was labeled, and on 3080 of those sides, chromosome 2 was labeled, and on 2640 of those sides, chromosome 3 was labeled, and on 556 of those sides, chromosome 4

was labeled, and on 2640 of those sides, chromosome 5 was labeled. To choose our first pair we roll the die twice, and take those chromosomes to be our first two parents. Continuing in this fashion we have the following parents:

Father Chromosome	Mother Chromosome
3	1
5	2
3	5
2	5
5	3

Table 3.4: Simulated Selection of Parents

The offspring of each of these parents contains the genetic information of both father and mother. How this can be determined is very arbitrary. However for this case, we could use something called a "cross-over". Let us say a mother has the solution set a_1,b_1,c_1,d_1, and a father has the solution set a_2,b_2,c_2,d_2, then there can be six possible cross overs (| = dividing line):

Father Chromosome	Mother Chromosome	Offspring Chromosome
a_1 | b_1,c_1,d_1	a_2 | b_2,c_2,d_2	a_1,b_2,c_2,d_2 or a_2,b_1,c_1,d_1
a_1,b_1 | c_1,d_1	a_2,b_2 | c_2,d_2	a_1,b_1,c_2,d_2 or a_2,b_2,c_1,d_1
a_1,b_1,c_1 | d_1	a_2,b_2,c_2 | d_2	a_1,b_1,c_1,d_2 or a_2,b_2,c_2,d_1

Table 3.5: Generalization of Cross-overs Between Parents

There are many other ways in which parents can trade genetic information to create an offspring, crossing over is just one way. Where the dividing line would be located is completely arbitrary, and so is whether or not the father or mother will contribute to the left or right of the dividing line. Now let's apply this to our offspring:

Father Chromosome	Mother Chromosome	Offspring Chromosome
(13 | 5,7,3)	(1 | 28,15,3)	(13,28,15,3)
(9,13 | 5,2)	(14,9 | 2,4)	(9,13,2,4)
(13,5,7 | 3)	(9,13,5 | 2)	(13,5,7,2)
(14 | 9,2,4)	(9 | 13,5,2)	(14,13,5,2)
(13,5 | 7, 3)	(9,13 | 5, 2)	(13,5,5,2)

Table 3.6: Simulated Crossovers from Parent Chromosomes

Now we can calculate the fitness values for the new generation of offsprings.

Offspring Chromosome	Fitness Value
(13,28,15,3)	|126-30|=96
(9,13,2,4)	|57-30|=27
(13,5,7,2)	|57-30|=22
(14,13,5,2)	|63-30|=33
(13,5,5,2)	|46-30|=16

Table 3.7: Fitness Values of Offspring Chromosomes

The average fitness value for the offspring chromosomes were 38.8, while the average fitness value for the parent chromosomes were 59.4. Of course, the next generation (the offspring) are supposed to mutate, that is, for example we can change one of the values in the ordered quadruple of each chromosome to some random integer between 1 and 30. Progressing at this rate, one chromosome should eventually reach a fitness level of 0 eventually, that is when a solution is found. If you tried and simulated this yourself, depending on the mutations you may actually get a fitness average that is higher, but on the long-run, the fitness levels will decrease. For systems where the population is larger (say 50, instead of 5), the fitness levels should more steadily and stabily approach the desired level (0).

■

ตัวอย่างที่ 4 Particle Swarm Optimization

Here we show a simple example of evolving ANN with PSO. The problem is a benchmark function of classification problem: iris data set. Measurements of four attributes of iris flowers are provided in each data set record: sepal length, sepal width, petal length, and petal width. Fifty sets of measurements are present for each of three varieties of iris flowers, for a total of 150 records, or patterns.

A 3-layer ANN is used to do the classification. There are 4 inputs and 3 outputs. So the input layer has 4 neurons and the output layer has 3 neurons. One can evolve the number of hidden neurons. However, for demonstration only, here we suppose the hidden layer has 6 neurons. We can evolve other parameters in the feed-forward network. Here we only evolve the network weights. So the particle will be a group of weights, there are 4*6+6*3 = 42

weights, so the particle consists of 42 real numbers. The range of weights can be set to [-100, 100] (this is just a example, in real cases, one might try different ranges). After encoding the particles, we need to determine the fitness function. For the classification problem, we feed all the patterns to the network whose weights is determined by the particle, get the outputs and compare it the standard outputs. Then we record the number of misclassified patterns as the fitness value of that particle. Now we can apply PSO to train the ANN to get lower number of misclassified patterns as possible. There are not many parameters in PSO need to be adjusted. We only need to adjust the number of hidden layers and the range of the weights to get better results in different trials.

■