Problems and Solutions related to AI Algorithms

Problems on Neural Networks

Question 1

Below is a diagram if a single artificial neuron (unit):

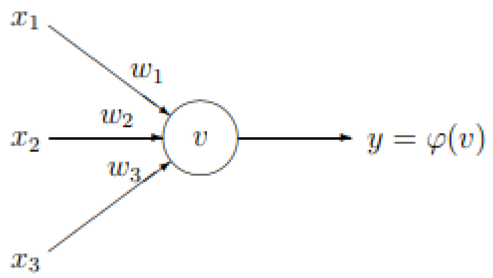


Figure 1: Single unit with three inputs.

The node has three inputs $\mathbf{x} = (x_1, x_2, x_3)$ that receive only binary signals (either 0 or 1). How many different input patterns this node can receive? What if the node had four inputs? Five? Can you give a formula that computes the number of binary input patterns for a given number of inputs?

Answer: *For three inputs the number of combinations of 0 and 1 is 8:*

x_1	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1
x_3	0	0	0	0	1	1	1	1

and for four inputs the number of combinations is 16:

x_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

You may check that for five inputs the number of combinations will be 32. Note that $8 = 2^3$, $16 = 2^4$ *and* $32 = 2^5$ *(for three, four and five inputs). Thus, the formula for the number of binary input patterns is:*

$$2^n \;, \quad \textit{where } n \textit{ in the number of inputs}$$

Question 2

Consider the unit shown on Figure 1. Suppose that the weights corresponding to the three inputs have the following values:

w_1	=	2
w_2	=	-4
w_3	=	1

and the activation of the unit is given by the step-function:

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Calculate what will be the output value y of the unit for each of the followinginput patterns:

Pattern	P_1	P_2	P_3	P_4

x_1	1	0	1	1
x_2	0	1	0	1
x_3	0	1	1	1

Answer: *To find the output value y for each pattern we have to:*

a) *Calculate the weighted sum:* $v = \sum_i w_i x_i = w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3$

b) *Apply the activation function to v*

The calculations for each input pattern are:

$$P_1: \quad v = 2 \cdot 1 - 4 \cdot 0 + 1 \cdot 0 = 2\,, \ (2 > 0)\,, \quad y = \phi(2) = 1$$

$$P_2: \quad v = 2 \cdot 0 - 4 \cdot 1 + 1 \cdot 1 = -3\,, \ (-3 < 0)\,, \quad y = \phi(-3) = 0$$

$$P_3: \quad v = 2 \cdot 1 - 4 \cdot 0 + 1 \cdot 1 = 3\,, \ (3 > 0)\,, \quad y = \phi(3) = 1$$

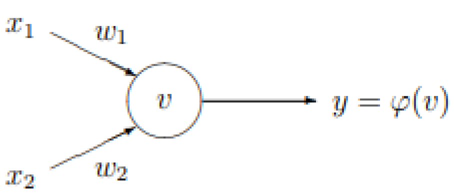
$$P_4: \quad v = 2 \cdot 1 - 4 \cdot 1 + 1 \cdot 1 = -1\,, \ (-1 < 0)\,, \quad y = \phi(-1) = 0$$

Question 3

Logical operators (i.e. NOT, AND, OR, XOR, etc) are the building blocks of any computational device. Logical functions return only two possible values, true or false, based on the truth or false values of their arguments. For example, operator AND returns true only when all its arguments are true, otherwise (if any of the arguments is false) it returns false. If we denote truth by 1 and false by 0, then logical function AND can be represented by the following table:

x_1 :	0	1	0	1
x_2 :	0	0	1	1
x_1 AND x_2 :	0	0	0	1

This function can be implemented by a single-unit with two inputs:



if the weights are $w_1 = 1$ and $w_2 = 1$ and the activation function is:

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 2 \\ 0 & \text{otherwise} \end{cases}$$

Note that the threshold level is 2 ($v \geq 2$).

a) Test how the neural AND function works.

Answer:

$$
\begin{array}{llll}
P_1: & v = 1\cdot 0 + 1\cdot 0 = 0\ , & (0 < 2)\ , & y = \varphi(0) = 0 \\
P_2: & v = 1\cdot 1 + 1\cdot 0 = 1\ , & (1 < 2)\ , & y = \varphi(1) = 0 \\
P_3: & v = 1\cdot 0 + 1\cdot 1 = 1\ , & (1 < 2)\ , & y = \varphi(1) = 0 \\
P_4: & v = 1\cdot 1 + 1\cdot 1 = 2\ , & (2 = 2)\ , & y = \varphi(2) = 1
\end{array}
$$

b) Suggest how to change either the weights or the threshold level of this single-unit in order to implement the logical OR function (true when at least one of the arguments is true):

x_1 :	0	1	0	1
x_2 :	0	0	1	1
x_1 OR x_2 :	0	1	1	1

Answer: *One solution is to increase the weights of the unit:* $w_1 = 2$ *and* $w_2 = 2$*:*

$$\begin{array}{llll} P_1: & v = 2\cdot 0 + 2\cdot 0 = 0\,, & (0 < 2)\,, & y = \varphi(0) = 0 \\ P_2: & v = 2\cdot 1 + 2\cdot 0 = 2\,, & (2 = 2)\,, & y = \varphi(2) = 1 \\ P_3: & v = 2\cdot 0 + 2\cdot 1 = 2\,, & (2 = 2)\,, & y = \varphi(2) = 1 \\ P_4: & v = 2\cdot 1 + 2\cdot 1 = 4\,, & (4 > 2)\,, & y = \varphi(4) = 1 \end{array}$$

Alternatively, we could reduce the threshold to 1:

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

c) The XOR function (exclusive or) returns true only when one of the arguments is true and another is false. Otherwise, it returns always false. This can be represented by the following table:

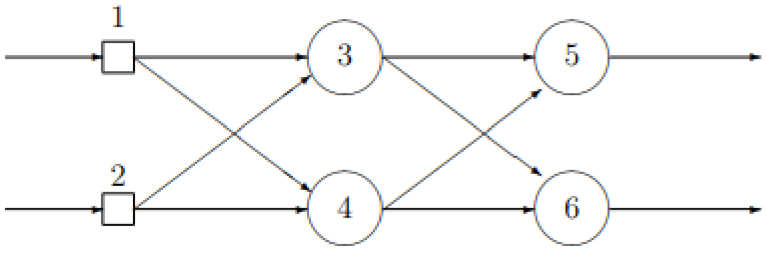
x_1 :	0	1	0	1
x_2 :	0	0	1	1
x_1 XOR x_2 :	0	1	1	0

Do you think it is possible to implement this function using a single unit? A network of several units?

Answer: *This is a difficult question, and it puzzled scientists for some time because it is actually impossible to implement the XORfunction neither by a single unit nor by a single-layer feed-forward net-work (single-layer perceptron). This was known as the XOR problem. The solution was found using a feed-forward network with a hidden layer. The XOR network uses two hidden nodes and one output node.*

Question 4

The following diagram represents a feed-forward neural network with onehidden layer



A weight on connection between nodes i and j is denoted by w_{ij}, such as w_{13} is the weight on the connection between nodes 1 and 3. The following table lists all the weights in the network:

$w_{13} = -2$ $w_{23} = 3$	$w_{35} = 1$ $w_{45} = -1$
$w_{14} = 4$ $w_{24} = -1$	$w_{36} = -1$ $w_{46} = 1$

Each of the nodes 3, 4, 5 and 6 uses the following activation function:

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

where v denotes the weighted sum of a node. Each of the input nodes (1and 2) can only receive binary values (either 0 or 1). Calculate the outputof the network (y_5 and y_6) for each of the input patterns:

Pattern:	P_1	P_2	P_3	P_4
Node 1:	0	1	0	1
Node 2:	0	0	1	1

Answer: *In order to find the output of the network it is necessary to calculate weighted sums of hidden nodes 3 and 4:*

$$v_3 = w_{13}x_1 + w_{23}x_2 \ , \ \ v_4 = w_{14}x_1 + w_{24}x_2$$

Then find the outputs from hidden nodes using activation function ϕ:

$$y_3 = \phi(v_3) \ , \quad y_4 = \phi(v_4) \ .$$

Use the outputs of the hidden nodes y_3 and y_4 as the input values to the output layer (nodes 5 and 6), and find weighted sums of output nodes 5 and 6:

$$v_5 = w_{35}y_3 + w_{45}y_4 \ , \ \ v_6 = w_{36}y_3 + w_{46}y_4 \ .$$

Finally, find the outputs from nodes 5 and 6 (also using ϕ):

$$y_5 = \phi(v_5) \ , \quad y_6 = \phi(v_6) \ .$$

The output pattern will be (y_5, y_6). Perform these calculation for each input pattern:

P_1*: Input pattern* $(0, 0)$

$$v_3 = -2 \cdot 0 + 3 \cdot 0 = 0, \quad y_3 = \phi(0) = 1$$

$$v_4 = 4 \cdot 0 - 1 \cdot 0 = 0, \quad y_4 = \phi(0) = 1$$

$$v_5 = 1 \cdot 1 - 1 \cdot 1 = 0, \quad y_5 = \phi(0) = 1$$

$$v_6 = -1 \cdot 1 + 1 \cdot 1 = 0, \quad y_6 = \phi(0) = 1$$

The output of the network is $(1, 1)$.

P_2: Input pattern $(1, 0)$

$$
\begin{aligned}
v_3 &= -2 \cdot 1 + 3 \cdot 0 = -2, & y_3 &= \phi(-2) = 0 \\
v_4 &= 4 \cdot 1 - 1 \cdot 0 = 4, & y_4 &= \phi(4) = 1 \\
v_5 &= 1 \cdot 0 - 1 \cdot 1 = -1, & y_5 &= \phi(-1) = 0 \\
v_6 &= -1 \cdot 0 + 1 \cdot 1 = 1, & y_6 &= \phi(1) = 1
\end{aligned}
$$

The output of the network is $(0, 1)$.

P_3: Input pattern $(0, 1)$

$$
\begin{aligned}
v_3 &= -2 \cdot 0 + 3 \cdot 1 = 3, & y_3 &= \phi(3) = 1 \\
v_4 &= 4 \cdot 0 - 1 \cdot 1 = -1, & y_4 &= \phi(-1) = 0 \\
v_5 &= 1 \cdot 1 - 1 \cdot 0 = 1, & y_5 &= \phi(1) = 1 \\
v_6 &= -1 \cdot 1 + 1 \cdot 0 = -1, & y_6 &= \phi(-1) = 0
\end{aligned}
$$

The output of the network is $(1, 0)$.

P_4: Input pattern $(1, 1)$

$$
\begin{aligned}
v_3 &= -2 \cdot 1 + 3 \cdot 1 = 1, & y_3 &= \phi(1) = 1 \\
v_4 &= 4 \cdot 1 - 1 \cdot 1 = 3, & y_4 &= \phi(3) = 1 \\
v_5 &= 1 \cdot 1 - 1 \cdot 1 = 0, & y_5 &= \phi(0) = 1 \\
v_6 &= -1 \cdot 1 + 1 \cdot 1 = 0, & y_6 &= \phi(0) = 1
\end{aligned}
$$

The output of the network is $(1, 1)$.

Question 5

What is a training set and how is it used to train neural networks?

Answer: *Training set is a set of pairs of input patterns with corresponding desired output patterns. Each pair represents how the network is supposed to respond to a particular input. The network is trained to respond correctly to each input pattern from the training set. Training algorithms that use training sets are called supervised learning algorithms. We may think ofa supervised learning as learning with a teacher, and the training set as aset of examples. During training the network, when presented with input patterns, gives 'wrong' answers (not desired output). The error is used to adjust the weights in the network so that next time the error was smaller. This procedure is repeated using many examples (pairs of inputs and desired outputs) from the training set until the error becomes sufficiently small.*

Question 6

What is an epoch?

Answer: *An epoch is when all of the data in the training set is presentedto the neural network once.*

Question 7

Describe the main steps of the supervised training algorithm?

Answer:

- *Initially, set all the weights to some random values*
- *Repeat (for many epochs):*
  - ***a)*** *Feed the network with an input from one of the examples in the training set*
  - ***b)*** *Compute the error between the output of the network and the desired output*
  - ***c)*** *Correct the error by adjusting the weights of the nodes*
- *Until the error is sufficiently small*

Question 8

Suppose that a credit card company decided to deploy a new system for assessing credit worthiness of its customers. The new system is using a feed-forward neural network with a supervised learning algorithm. Suggest in a form of essay what should the bank have before the system can be used?Discuss problems associated with this requirement.

Answer: *The answer should mention that the company should get holdof* **historical data** *about its customers who already took credit in the past. This data will be used as a* **training set** *for the neural network. It is important that the data is* **representative** *and covers as many types of customers as possible. This is because the network will not be able to produce an accurate answer for a customer very different from those in the training set.*

Problem on Genetic Algorithm

*Suppose there is equality **a** + 2**b** + 3**c** + 4**d** = 30, genetic algorithm will be used to find the value of **a**, **b**, **c**, and **d** that satisfy the above equation. First we should formulate the objective function, for this problem the objective is minimizing the value of function **f(x)** where **f(x)** = ((**a** + 2**b** + 3**c** + 4**d**) - 30). Since there are four variables in the equation, namely **a**, **b**, **c**, and **d**, we can compose the chromosome as follow: To speed up the computation, we can restrict that the values of variables **a**, **b**, **c**, and **d** are integers between 0 and 30.*

a	b	c	d

Step 1. Initialization

*For example we define the number of chromosomes in population are 6, then we generate random value of gene **a**, **b**, **c**, **d** for 6 chromosomes*

Chromosome[1] = [**a**;**b**;**c**;**d**] = [12;05;23;08]

Chromosome[2] = [**a**;**b**;**c**;**d**] = [02;21;18;03]

Chromosome[3] = [**a**;**b**;**c**;**d**] = [10;04;13;14]

Chromosome[4] = [**a**;**b**;**c**;**d**] = [20;01;10;06]

Chromosome[5] = [**a**;**b**;**c**;**d**] = [01;04;13;19]

Chromosome[6] = [**a**;**b**;**c**;**d**] = [20;05;17;01]

Step 2. Evaluation

We compute the objective function value for each chromosome produced in initialization step:

***F_obj**[1] = Abs((12 + 2*05 + 3*23 + 4*08) - 30)*
= Abs((12 + 10 + 69 + 32) - 30)
= Abs(123 - 30)
= 93

***F_obj**[2] = Abs((02 + 2*21 + 3*18 + 4*03) - 30)*
= Abs((02 + 42 + 54 + 12) - 30)
= Abs(110 - 30)
= 80

***F_obj**[3] = Abs((10 + 2*04 + 3*13 + 4*14) - 30)*
= Abs((10 + 08 + 39 + 56) - 30)
= Abs(113 - 30)
= 83

***F_obj**[4] = Abs((20 + 2*01 + 3*10 + 4*06) - 30)*
= Abs((20 + 02 + 30 + 24) - 30)
= Abs(76 - 30)
= 46

***F_obj**[5] = Abs((01 + 2*04 + 3*13 + 4*19) - 30)*
= Abs((01 + 08 + 39 + 76) - 30)

$= Abs(124 - 30)$

$= 94$

F_obj$[6] = Abs((20 + 2*05 + 3*17 + 4*01) - 30)$

$= Abs((20 + 10 + 51 + 04) - 30)$

= Abs(85 - 30)
= 55

Step 3. Selection

1. *The fittest chromosomes have higher probability to be selected for the next generation. To compute fitness probability we must compute the fitness of each chromosome. To avoid divide by zero problem, the value of* ***F_obj*** *is added by 1.*
Fitness*[1] = 1 / (1+****F_obj****[1])*
= 1 / 94
= 0.0106
Fitness[2] = 1 / (1+**F_obj**[2])
= 1 / 81
= 0.0123
Fitness[3] = 1 / (1+**F_obj**[3])
= 1 / 84
= 0.0119
Fitness[4] = 1 / (1+**F_obj**[4])
= 1 / 47
= 0.0213
Fitness[5] = 1 / (1+**F_obj**[5])
= 1 / 95
= 0.0105
Fitness[6] = 1 / (1+**F_obj**[6])
= 1 / 56
= 0.0179

Total *= 0.0106 + 0.0123 + 0.0119 + 0.0213 + 0.0105 + 0.0179*
= 0.0845

The probability for each chromosomes is formulated by: ***P****[i] =* ***Fitness****[i] /* ***Total***
P*[1] = 0.0106 / 0.0845*
= 0.1254
P*[2] = 0.0123 / 0.0845*
= 0.1456
P*[3] = 0.0119 / 0.0845*
= 0.1408
P*[4] = 0.0213 / 0.0845*
= 0.2521
P*[5] = 0.0105 / 0.0845*
= 0.1243
P*[6] = 0.0179 / 0.0845*
= 0.2118

From the probabilities above we can see that Chromosome 4 that has the highest fitness, this chromosome has highest probability to be selected for next generation chromosomes. For the *selection process we use roulette wheel, for that we should compute the cumulative probability values:*

***C**[1] = 0.1254*

***C**[2] = 0.1254 + 0.1456*
= 0.2710

***C**[3] = 0.1254 + 0.1456 + 0.1408*
= 0.4118

***C**[4] = 0.1254 + 0.1456 + 0.1408 + 0.2521*
= 0.6639

***C**[5] = 0.1254 + 0.1456 + 0.1408 + 0.2521 + 0.1243*
= 0.7882

***C**[6] = 0.1254 + 0.1456 + 0.1408 + 0.2521 + 0.1243 + 0.2118*
= 1.0

*Having calculated the cumulative probability of selection process using roulette-wheel can be done. The process is to generate random number **R** in the range 0-1 as follows.*

***R**[1] = 0.201*
***R**[2] = 0.284*
***R**[3] = 0.099*
***R**[4] = 0.822*
***R**[5] = 0.398*
***R**[6] = 0.501*

*If random number **R[1]** is greater than **C**[1] and smaller than **C**[2] then select **Chromosome**[2] as a chromosome in the new population for next generation:*

***NewChromosome**[1] = **Chromosome**[2]*
NewChromosome[2] = **Chromosome**[3]
NewChromosome[3] = **Chromosome**[1]
NewChromosome[4] = **Chromosome**[6]
NewChromosome[5] = **Chromosome**[3]
NewChromosome[6] = **Chromosome**[4]

Chromosomes in the population thus became:

Chromosome[1] = [02;21;18;03]

Chromosome[2] = [10;04;13;14]

Chromosome[3] = [12;05;23;08]

Chromosome[4] = [20;05;17;01]

Chromosome[5] = [10;04;13;14]

Chromosome[6] = [20;01;10;06]

*In this example, we use one-cut point, i.e. randomly select a position in the parent chromosome then exchanging sub-chromosome. Parent chromosome which will mate israndomly selected and the number of mate Chromosomes is controlled using **crossover_rate (ρc)** parameters. Pseudo-code for the crossover process is as follows:*

```
begin
    k← 0;
    while(k<population) do
    R[k] = random(0-1);
    if(R[k]< ρc) then

      select Chromosome[k] as parent;
    end;

    k = k + 1;
    end;

end;
```

*Chromosome **k** will be selected as a parent if **R[k]<ρc**. Suppose we set that the crossover rate is 25%, then Chromosome number **k** will be selected for crossover if random generated value for Chromosome **k** below 0.25. The process is as follows: First we generate a random number **R** as the number of population.*

***R**[1] = 0.191*
***R**[2] = 0.259*
***R**[3] = 0.760*
***R**[4] = 0.006*
***R**[5] = 0.159*
***R**[6] = 0.340*

For random number **R** above, parents are **Chromosome**[1], **Chromosome**[4] and **Chromosome**[5] will be selected for crossover.

Chromosome[1] >< **Chromosome**[4]
Chromosome[4] >< **Chromosome**[5]
Chromosome[5] >< **Chromosome**[1]

After chromosome selection, the next process is determining the position of the crossover point. This is done by generating random numbers between 1 to (length of Chromosome – 1).In this case, generated random numbers should be between 1 and 3. After we get thecrossover point, parents Chromosome will be cut at crossover point and its gens will be interchanged. For example we generated 3 random number and we get:

***C**[1] = 1*
***C**[2] = 1*
***C**[3] = 2*

Then for first crossover, second crossover and third crossover, parent's gens will be cut at gen number 1, gen number 1 and gen number 3 respectively, e.g.

Chromosome[1] = **Chromosome**[1] >< **Chromosome**[4]

= [02;21;18;03] >< [20;05;17;01]
= [02;05;17;01]

Chromosome[4] = **Chromosome**[4] >< **Chromosome**[5]

= [20;05;17;01] >< [10;04;13;14]

= [20;04;13;14]

Chromosome[5] = **Chromosome**[5] >< **Chromosome**[1]

= [10;04;13;14] >< [02;21;18;03]
= [10;04;18;03]

Thus Chromosome population after experiencing a crossover process:

Chromosome[1] = [02;05;17;01]

Chromosome[2] = [10;04;13;14]

Chromosome[3] = [12;05;23;08]

Chromosome[4] = [20;04;13;14]

Chromosome[5] = [10;04;18;03]

Chromosome[6] = [20;01;10;06]

Step 5. Mutation

*Number of chromosomes that have mutations in a population is determined by the **mutation_rate** parameter. Mutation process is done by replacing the gen at random position with a new value. The process is as follows. First we must calculate the total length of gen in the population. In this case the total length of gen is **total_gen = number_of_gen_in_Chromosome * number of population***
*= 4 * 6*
= 24

Mutation process is done by generating a random integer between 1 and total_gen (1 to 24). If generated random number is smaller than mutation_rate(ρm) variable then marked the position of gen in chromosomes. Suppose we define ρm 10%, it is expected that 10% (0.1) of total_gen in the population that will be mutated:
*number of mutations = 0.1 * 24*
= 2.4
≈ 2

Suppose generation of random number yield 12 and 18 then the chromosome which have mutation are Chromosome number 3 gen number 4 and Chromosome 5 gen number 2. The value of mutated gens at mutation point is replaced by random number between 0-30. Suppose generated random number are 2 and 5 then Chromosome composition after mutationare:

Chromosome[1] = [02;05;17;01]

Chromosome[2] = [10;04;13;14]

Chromosome[3] = [12;05;23;**02**]

Chromosome[4] = [20;04;13;14]

Chromosome[5] = [10;**05**;18;03]

Chromosome[6] = [20;01;10;06]

Finishing mutation process then we have one iteration or one generation of the genetic algorithm. We can now evaluate the objective function after one generation:

Chromosome[1] = [02;05;17;01]

F_obj_[1] = Abs((02 + 2*05 + 3*17 + 4*01) - 30)_
= Abs((2 + 10 + 51 + 4) - 30)
= Abs(67 - 30)
_= 37_

Chromosome[2] = [10;04;13;14]

F_obj_[2] = Abs((10 + 2*04 + 3*13 + 4*14) - 30)_
= Abs((10 + 8 + 33 + 56) - 30)
= Abs(107 - 30)
_= 77_

Chromosome[3] = [12;05;23;02]

F_obj_[3] = Abs((12 + 2*05 + 3*23 + 4*02) - 30)_
= Abs((12 + 10 + 69 + 8) - 30)
= Abs(87 - 30)
_= 47_

Chromosome[4] = [20;04;13;14]

F_obj_[4] = Abs((20 + 2*04 + 3*13 + 4*14) - 30)_
= Abs((20 + 8 + 39 + 56) - 30)
= Abs(123 - 30)
_= 93_

Chromosome[5] = [10;05;18;03]

F_obj_[5] = Abs((10 + 2*05 + 3*18 + 4*03) - 30)_
= Abs((10 + 10 + 54 + 12) - 30)
= Abs(86 - 30)
_= 56_

Chromosome[6] = [20;01;10;06]

F_obj_[6] = Abs((20 + 2*01 + 3*10 + 4*06) - 30)_
= Abs((20 + 2 + 30 + 24) - 30)
= Abs(76 - 30)
_= 46_

From the evaluation of new Chromosome we can see that the objective function is decreasing, this means that we have better Chromosome or solution compared with previousChromosome generation. New Chromosomes for next iteration are:

Chromosome[1] = [02;05;17;01]

Chromosome[2] = [10;04;13;14]

Chromosome[3] = [12;05;23;02]

Chromosome[4] = [20;04;13;14]

Chromosome[5] = [10;05;18;03]

Chromosome[6] = [20;01;10;06]

These new Chromosomes will undergo the same process as the previous generation of Chromosomes such as evaluation, selection, crossover and mutation and at the end it produce new generation of Chromosome for the next iteration. This process will be repeated until a *predetermined number of generations. For this example, after running 50 generations, bestchromosome is obtained:*

Chromosome = [07; 05; 03; 01]

This means that: ***a*** *= 7,* ***b*** *= 5,* ***c*** *= 3,* ***d*** *= 1*

If we use the number in the problem equation:

$\mathbf{a} + 2\mathbf{b} + 3\mathbf{c} + 4\mathbf{d} = 30$

$7 + (2 * 5) + (3 * 3) + (4 * 1) = 30$

We can see that the value of variable ***a****,* ***b****,* ***c*** *and* ***d*** *generated by genetic algorithm can satisfythat equality.*

REFERENCE

[1] Mitsuo Gen, Runwei Cheng, "Genetic Algorithms And Engineering Design", John Wiley& Sons, 1997.

[2] Denny Hermawanto, "Genetic Algorithm for Solving Simple Mathematical Equality Problem"