BRAINWARE UNIVERSITY

Name of the Paper	:	Soft Computing Lab
Lab Paper Code	:	BCA692B
Registration No.	:	18013000073 of 2018-2019
Student Name	:	ABHIJIT BARIK
University Roll Number	:	18010301001
Course	:	BCA
Semester	:	6th
Department	:	Computational Science

Assignment 4

In [14]:
```
pip install geneticalgorithm
```

```
Requirement already satisfied: geneticalgorithm in c:\users\hp\desktop\ml-data-science\sample_project_1\env\lib\site-packages (1.0.2)
Requirement already satisfied: numpy in c:\users\hp\desktop\ml-data-science\sample_project_1\env\lib\site-packages (from geneticalgorithm) (1.19.1)
Requirement already satisfied: func-timeout in c:\users\hp\desktop\ml-data-science\sample_project_1\env\lib\site-packages (from geneticalgorithm) (4.3.5)
Note: you may need to restart the kernel to use updated packages.
```

Q1).Write a program to design a Pitts neural network model using AND gate

In [5]:
```
#AND gate using Pitts model
vx1 = [0,0,1,1]
vx2 = [0,1,0,1]
weights = [1,1]
threshold = len(weights)
bias = 0
#For AND only activation is when x1 = x2 = 1
# So, threshold value of activation >= 2 (both active)
def AND_neuron(input_x):
     s = 0
     for i in range (0, len(input_x)):
         s = s + input_x[i]*weights[i]

     s_final = s + bias
     if(s_final >= threshold):
         y = 1

     else:
         y = 0
     return y
print("Logical AND using Pitts Neural Network")
for i,j in zip(vx1,vx2):
     print("X1 =",i,"& X2 =",j," Then Y =",AND_neuron([i,j]))
```

```
Logical AND using Pitts Neural Network
X1 = 0 & X2 = 0  Then Y = 0
X1 = 0 & X2 = 1  Then Y = 0
X1 = 1 & X2 = 0  Then Y = 0
X1 = 1 & X2 = 1  Then Y = 1
```

Q2) Write a program to design a Hebb's Learning model using AND-OR gate

In [9]:
```
# Hebbs AND, OR networks
import numpy as np
def bipolar_sigmoid(x):
 y = (np.exp(x)-1)/(np.exp(x)+1)
 return y
# Initializing training sets and targets
vx1 = [-1, -1, 1, 1]
vx2 = [-1, 1, -1, 1]
target_AND = [-1, -1, -1, 1]
target_OR = [-1, 1, 1, 1]
bias = 1
# Initializing weights of x1, x2 and the bias

weights_AND = [0,0,0]
weights_OR = [0,0,0]
input_vectors = []
for i in range(0,len(target_AND)):
    input_vectors.append([vx1[i],vx2[i],bias])
# New_Weights = Old_Weights + Input_Vector(i) * target(i)
def adjust_weights(input_vectors,target_arr,weight_arr):
    for i in range(0,len(input_vectors)):
        temp = []
        for j in range(0,len(input_vectors[i])):
             temp.append(input_vectors[i][j]*target_arr[i])

        for k in range(0,len(weight_arr)):
             weight_arr[k] = weight_arr[k] + temp[k]
    return weight_arr
adjust_weights(input_vectors,target_AND,weights_AND)
adjust_weights(input_vectors,target_OR,weights_OR)
# Testing (bias passed with input_x)

def Hebbs_Neuron(input_x,weights):
     s_final = 0
     for Xi,Wi in zip(input_x,weights):
         s_final = s_final + Xi*Wi
     # Passing through the activation function
     y = bipolar_sigmoid(s_final)
     return y
print("Logical AND using Hebbs Neural Network")
for i,j,k in zip(vx1,vx2,[1,1,1,1]):
    print("X1 =",i,"& X2 =",j," Then Y =",Hebbs_Neuron([i,j,k],weights_AND))
print("Logical OR using Hebbs Neural Network")
for i,j,k in zip(vx1,vx2,[1,1,1,1]):
    print("X1 =",i,"|| X2 =",j," Then Y =",Hebbs_Neuron([i,j,k],weights_OR))
```

```
Logical AND using Hebbs Neural Network
X1 = -1 & X2 = -1  Then Y = -0.9950547536867306
X1 = -1 & X2 = 1  Then Y = -0.7615941559557649
X1 = 1 & X2 = -1  Then Y = -0.7615941559557649
X1 = 1 & X2 = 1  Then Y = 0.7615941559557649
Logical OR using Hebbs Neural Network
X1 = -1 || X2 = -1  Then Y = -0.7615941559557649
X1 = -1 || X2 = 1  Then Y = 0.7615941559557649
X1 = 1 || X2 = -1  Then Y = 0.7615941559557649
X1 = 1 || X2 = 1  Then Y = 0.9950547536867305
```

Q3) Write a program to design an Adaline Network model

In [10]:
```
#AND-NOT (NAND) gate using ADALINE Network
import numpy as np
def binary_step(x):
 if(x >= 0):
     return 1
 else:
     return -1
x1 = [1, 1, -1, -1]
x2 = [1, -1, 1, -1]
target = [-1, 1, 1, 1]
w1 = 0.5
w2 = 0.5
bias = 0.1
l_rate = 0.1
error = [0, 0, 0, 0]
no_epoch = 80
def train_network(x1,x2,target,w1,w2,bias,l_rate,error,no_epoch):
 values_store = []
 for epoch in range(no_epoch):
     s_error = 0
     for i in range(4):
         Yin = bias + x1[i]*w1 + x2[i]*w2
         diff = target[i] - Yin
         w1 = w1 + l_rate*diff*x1[i]
         w2 = w2 + l_rate*diff*x2[i]
         bias = bias + l_rate*diff
         error[i] = diff*diff
         s_error = s_error + error[i]
     values_store.append([w1, w2, epoch+1, s_error])
 print("Training complete, Epochs =",no_epoch)
 return w1,w2,bias,error,values_store
w1,w2,bias,error,values_store = train_network(x1, x2, target, w1,w2, bias, l_rate, error, no_epoch)
def predict(inputs_x,weights,bias):
 s = 0
 for i in range(0,len(inputs_x)):
     s = s + inputs_x[i]*weights[i]
 s_final = s + bias
 y = binary_step(s_final)
 return y
print("Logical NAND using Adaline Neural Network")
for i,j in zip(x1,x2):
 print("X1 =",i,"NAND X2 =",j," Then Y =",predict([i,j],[w1,w2],bias))
```

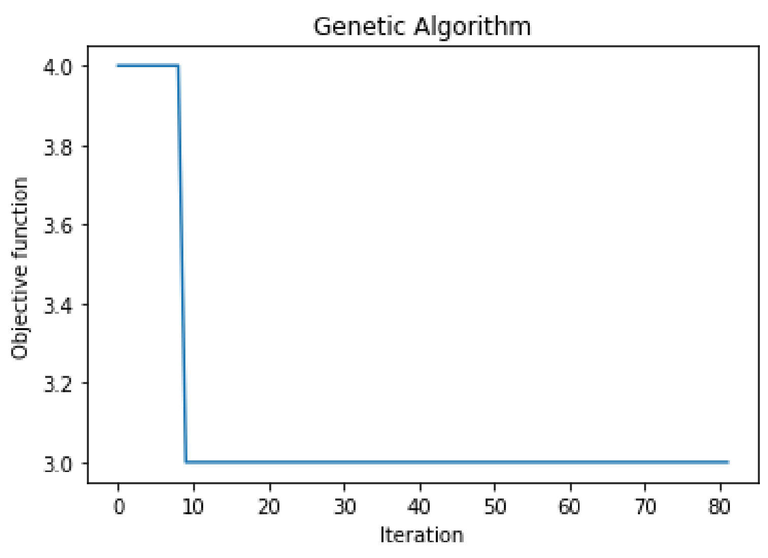
```
Training complete, Epochs = 80
Logical NAND using Adaline Neural Network
X1 = 1 NAND X2 = 1  Then Y = -1
X1 = 1 NAND X2 = -1  Then Y = 1
X1 = -1 NAND X2 = 1  Then Y = 1
X1 = -1 NAND X2 = -1  Then Y = 1
```

Q4) Write a program to design a genetic algorithm using python (When boundary is integer variable).

In [12]:
```
# GA where boundary variable is integer
import numpy as np
from geneticalgorithm import geneticalgorithm as ga
def f(x):
     return np.sum(x)
z=np.array([[1,10]]*3)
y=ga(function=f,dimension=3,variable_type='int',variable_boundaries=z)
y.run()
```

```
The best solution found:
[1. 1. 1.]

Objective function:
3.0
```

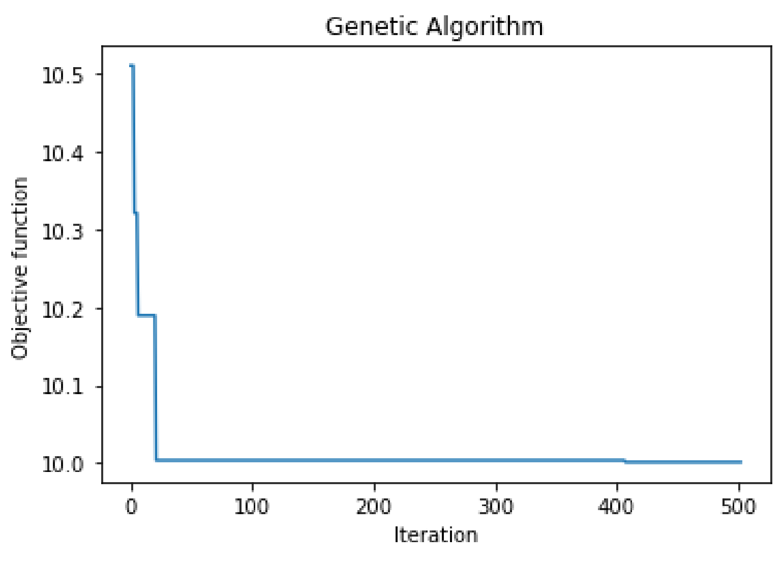


Q5) Write a program to design a genetic algorithm using python (When boundary is mixed variable).

In [13]:
```
# GA where boundary is mixed variable
import numpy as np
from geneticalgorithm import geneticalgorithm as ga
def f(x):
 return np.sum(x)
z=np.array([[5,6],[5.0,15.0]])
var_type=np.array([['int'],['real']])
y=ga(function=f,dimension=2,variable_type_mixed=var_type,variable_boundaries=z)
y.run()
```

```
The best solution found:
[5.         5.00149709]

Objective function:
10.001497088348916
```



In [ ]: