SOFT COMPUTING LABORATORY

LAB MANUAL

Academic Year : 2018-2019
Course Code : BCS208
Regulations : IARE-R18
Semester : II
Branch : CSE

PreparedBy

Ms. K Sai Saranya
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
Dundigal, Hyderabad – 500 043
1. PROGRAM OUTCOMES:

<table>
<thead>
<tr>
<th>M.TECH-PROGRAM OUTCOMES(POS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PO1</td>
</tr>
<tr>
<td>PO2</td>
</tr>
<tr>
<td>PO3</td>
</tr>
<tr>
<td>PO4</td>
</tr>
<tr>
<td>PO5</td>
</tr>
<tr>
<td>PO6</td>
</tr>
<tr>
<td>PO7</td>
</tr>
</tbody>
</table>

2. PROGRAM SPECIFIC OUTCOMES:

<table>
<thead>
<tr>
<th>PROGRAM SPECIFIC OUTCOMES(PEO's)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEO1</td>
</tr>
<tr>
<td>PEO2</td>
</tr>
<tr>
<td>PEO3</td>
</tr>
<tr>
<td>PEO4</td>
</tr>
</tbody>
</table>
3. ATTAINMENT OF PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES:

<table>
<thead>
<tr>
<th>S. No</th>
<th>Experiment</th>
<th>Program Outcomes Attained</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create a perceptron with appropriate number of inputs and outputs. Train it using fixed increment learning algorithm until no change in weights is required. Output the final weights</td>
<td>PO1</td>
</tr>
<tr>
<td>2</td>
<td>Write a program to implement artificial neural network without back propagation. Write a program to implement artificial neural network with back propagation.</td>
<td>PO2</td>
</tr>
<tr>
<td>3</td>
<td>Implement Union, Intersection, Complement and Difference operations on fuzzy sets. Also create fuzzy relation by Cartesian product of any two fuzzy sets and perform max-min composition on any two fuzzy relations.</td>
<td>PO3</td>
</tr>
<tr>
<td>4</td>
<td>Implement travelling sales person problem (tsp) using genetic algorithms.</td>
<td>PO4</td>
</tr>
<tr>
<td>5</td>
<td>Plot the correlation plot on dataset and visualize giving an overview of relationships among data on soya bins data. Analysis of covariance: variance (ANOVA), if data have categorical variables on iris data.</td>
<td>PO4</td>
</tr>
<tr>
<td>6</td>
<td>Implement linear regression and multi-regression for a set of data points</td>
<td>PO5</td>
</tr>
<tr>
<td>7</td>
<td>Implement crisp partitions for real-life iris dataset</td>
<td>PO5</td>
</tr>
<tr>
<td>8</td>
<td>Write a program to implement Hebb’s rule Write a program to implement Delta rule.</td>
<td>PO7</td>
</tr>
<tr>
<td>9</td>
<td>Write a program to implement logic gates.</td>
<td>PO4</td>
</tr>
<tr>
<td>10</td>
<td>Implement svm classification by fuzzy concepts.</td>
<td>PO3</td>
</tr>
</tbody>
</table>
**SYLLABUS:**

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Category</th>
<th>Hours / Week</th>
<th>Credits</th>
<th>Maximum Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCSB19</td>
<td>Core</td>
<td>L 0 T 0 P 4 C 2 CIA 30 SEE 70</td>
<td>Total 100</td>
<td></td>
</tr>
</tbody>
</table>

Contact Classes: Nil  Total Tutorials: Nil  Total Practical Classes: 36  Total Classes: 36

**OBJECTIVES:**
The course should enable the students to:
I. Understand Fuzzy concepts
II. Learn neural networks with back propagation and without preparation
III. Learn the operators of genetic algorithms
IV. Practice on crisp partitions

**LIST OF EXPERIMENTS**

**Week-1** PERCEPTRON
Create a perceptron with appropriate number of inputs and outputs. Train it using fixed increment learning algorithm until no change in weights is required. Output the final weights

**Week-2** ARTIFICIAL NEURAL NETWORKS
Write a program to implement artificial neural network without back propagation.
Write a program to implement artificial neural network with back propagation.

**Week-3** FUZZY SETS
Implement Union, Intersection, Complement and Difference operations on fuzzy sets. Also create fuzzy relation by Cartesian product of any two fuzzy sets and perform max-min composition on any two fuzzy relations.

**Week-4** GENETIC ALGORITHMS
Implement travelling sales person problem (TSP) using genetic algorithms.

**Week-5** COVARIANCE
Plot the correlation plot on dataset and visualize giving an overview of relationships among data on soya bins data. Analysis of covariance: variance (ANOVA), if data have categorical variables on iris data.

**Week-6** DATA FITTING BY REGRESSION
Implement linear regression and multi-regression for a set of data points.

**Week-7** CRISP MODEL
Implement crisp partitions for real-life iris dataset.

**Week-8** PERCEPTRON RULE
Write a program to implement Hebb’s rule Write a program to implement Delta rule.
Week-9

LOGIC GATES

Write a program to implement logic gates.

Week-10

CLASSIFICATION

Implement SVM classification by Fuzzy concepts.

Reference Books:


Web References:


SOFTWARE AND HARDWARE REQUIREMENTS FOR 18 STUDENTS:

SOFTWARE: Python

HARDWARE: 18 numbers of Intel Desktop Computers with 4 GB RAM
<table>
<thead>
<tr>
<th>S. No</th>
<th>List of Experiments</th>
<th>Page No</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PERCEPTRON</td>
<td>7-8</td>
</tr>
<tr>
<td>2</td>
<td>ARTIFICIAL NEURAL NETWORKS</td>
<td>9-11</td>
</tr>
<tr>
<td>3</td>
<td>FUZZY SETS</td>
<td>12-14</td>
</tr>
<tr>
<td>4</td>
<td>GENETIC ALGORITHMS</td>
<td>15-17</td>
</tr>
<tr>
<td>5</td>
<td>COVARIANCE</td>
<td>18-20</td>
</tr>
<tr>
<td>6</td>
<td>DATA FITTING BY REGRESSION</td>
<td>22-24</td>
</tr>
<tr>
<td>7</td>
<td>CRISP MODEL</td>
<td>25-26</td>
</tr>
<tr>
<td>8</td>
<td>PERCEPTRON RULE</td>
<td>27-28</td>
</tr>
<tr>
<td>9</td>
<td>LOGIC GATES</td>
<td>29-30</td>
</tr>
<tr>
<td>10</td>
<td>CLASSIFICATION</td>
<td>31-32</td>
</tr>
</tbody>
</table>
WEEK-1

DESCRIPTION: Write a program of Perceptron Training Algorithm.

Algorithm:
Start with a randomly chosen weight vector w0;
Let k=1;
While these exists input vector that are misclassified by: Wk-1 do
Let i be a misclassified input vector
Let Xk=class(i)i, implying that Wk-1.Xk<0
Update the weight vector to Wk= Wk-1 + nXk;
increment k;
End while;

Program:
#include<conio.h>
#include<iostream.h>
Void main( ) {
 clrscr( );
 int in[3],d,w[3],a=0;
 for(inti=0;i<3;i++)
 { 
 cout<<"n initialize the weight vector w"<<i;
 cin>>w[i] }
 for(i=0;i<3;i++)
 { 
 cout<<"n enter the input vector i"<<i;
 cin>>in[i];
 }
 cout<<"n enter the desined output";
 cin>>d;
 intans=1;
 while(ans= = 1)
 { 
 for (a= 0, i==0;i<3;i++)
 { 
 a = a + w[i] * in[i];
 }
 clrscr( );
 cout<<"n desired output is"<<d;
 cout<<"n actual output is "<<a;
 int e;
e=d-a;
cout<<"n error is "<<e;
cout<<"n press 1 to adjust weight else 0";
cin>>ans;
if (e<0)
 { 
 for(i=0;i<3;i++)
 { 
 w[i]=w[i]-1; 
 }
} else if (e>0)
{
 for(i=0;i<3;i++)
{
 w[i]=w[i]+1;
 }
}
getch( );

OUTPUT:
Desire output is 2
Actual output is 17
Error is -15
Press 1 to adjust weight else 0
WEEK-2
DESCRIPTION: Write a program for Back Propagation Algorithm

PROGRAM:

```c++
#include <iostream.h>
#include <conio.h>
void main ()
{
    int i ;
    float delta, com, coeff = 0.1;
    struct input
    {
        float val, out, wo, wi;
        int top;
    } s[3] ;
    cout<< "Enter the i/p value to target o/p" << "\n";
    for (i=0; i<3 ; i++)
        cin>> s [i].val>> s [i].top);
    i = 0;
    do
    {
        if (i = = 0)
        {
            W0 =-1.0;
            W1 =-0.3;
        }
        else
        {
            W0 = del [i-1]. w0 ;
            W1 = del [i-1]. wi ;
        }
        del [i]. aop = w0 + (wi * del [i]. val);
        del [i].out = del [i]. aop); 
        delta = (top – del [i]. out) * del [i].out * (1 – del [i].out);
        corr = coeff * delta * del [i].out); 
        del [i].w0 = w1 + corr;
        del [i]. w1 = w1 + corr;
        i++;
    }While ( i != 3)
    cout<< "VALUE"<<"Target"<<"Actual"<<"w0" "w1""\n; 
    for (i=0; i=3; i++)
    {
        cout<< s [i].val<< s[i].top<<s[i].out << s[i]. w0<< s[i]. w1;
        cout<< "\n";
    }
    getch ();
}
```
OUTPUT:
Back Propagation network
  1. Load data
  2. Learn from the data
  3. Compute output pattern
  4. Make new data file
  5. Save data
  6. Print data
  7. Change learning rate
  8. Exit

Enter your Choice (1-8)
WEEK-3

Fuzzy Logic:

Fuzzy logic is an organized method for dealing with imprecise data. It is a multivalued logic that allows intermediate values to be defined between conventional solutions. In classical set theory, the membership of elements in a set is assessed in binary terms according to a bivalent condition — an element either belongs or does not belong to the set. By contrast, fuzzy set theory permits the gradual assessment of the membership of elements in a set; this is described with the aid of a membership function valued in the real unit interval [0, 1]. Bivalent Set Theory can be somewhat limiting if we wish to describe a 'humanistic' problem mathematically.

1. **Union:**

Union of two fuzzy sets is denoted as

\[ \mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)) \]

Example

\[ A \cup B = \{(x_1, 0.4), (x_2, 0.7), (x_3, 1), (x_4, 0.2)\} \]

2. **Intersection**

Union of two fuzzy sets \(A\) and \(B\) is denoted by \(A \cap B\) and is defined as,
\[ \mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)) \]

Example \( \hat{A} \cap \hat{B} = \{(x_1,0.1),(x_2,0.3),(x_3,1),(x_4,0)\} \)

3) Compliment

Complement of a fuzzy set \( \hat{A} \) denoted by \( \hat{A}^c \) and is defined as,

Example \( \hat{A}^c = \{(x_1,0.9),(x_2,0.3),(x_3,0),(x_4,1)\} \)

Implementation of fuzzy relations (Max-Min Composition)

Consider two fuzzy relation; \( R (X \times Y) \) and \( S (Y \times Z) \), then a relation \( T (X \times Z) \), can be expressed as max-min composition.

\[
T = R \circ S \\
\mu_T(x, z) = \max \min[\mu_R(x, y), \mu_S(y, z)] = \vee[\mu_R(x, y) \land \mu_S(y, z)]
\]

If algebraic product is adopted, then max-product composition is adopted:

\[
T = R \circ S \\
\mu_T(x, z) = \max[\mu_R(x, y) \cdot \mu_S(y, z)] = \vee[\mu_R(x, y) \cdot \mu_S(y, z)]
\]

The max-min composition can be interpreted as indicating the strength of the existence of relation between the elements of \( X \) and \( Z \). Calculations of \( R \circ S \) are almost similar to matrix multiplication.

**Max-Min Composition:**

Let \( X, Y \) and \( Z \) be universal sets and let \( R \) and \( Q \) be relations that relate them as,

\[
R = \{ (x, y) | x \in X, y \in Y, R \subset X \times Y \} \\
Q = \{ (y, z) | y \in Y, z \in Z, Q \subset Y \times Z \}
\]

Then \( S \) will be a relation that relates elements of \( X \) with elements of \( Z \) as,
\[
S = R \circ Q \\
S = \{ (x, z) \mid x \in X, z \in Z, S \subseteq X \times Z \}
\]

Max min composition is then defined as,
WEEK-4

Description: Implement Travelling sales person using genetic Algorithm

PROGRAM:

```java
import java.util.*;
import java.text.*;

class TSP {
    int weight[][], n, tour[], finalCost;
    final int INF = 1000;
    public TSP() {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter no. of nodes:=>");
        n = s.nextInt();
        weight = new int[n][n];
        tour = new int[n - 1];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (i != j) {
                    System.out.print("Enter weight of " + (i + 1) + " to " + (j + 1) + ":=>");
                    weight[i][j] = s.nextInt();
                }
            }
        }
        System.out.println();
        System.out.println("Starting node assumed to be node 1.");
        eval();
    }
    public int COST(int currentNode, int inputSet[], int setSize) {
        if (setSize == 0)
            return weight[currentNode][0];
        int min = INF, minindex = 0;
        int setToBePassedOnToNextCallOfCOST[] = new int[n - 1];
        for (int i = 0; i < setSize; i++) {
            int k = 0; // initialise new set
            for (int j = 0; j < setSize; j++) {
                if (inputSet[i] != inputSet[j]) {
                    setToBePassedOnToNextCallOfCOST[k++] = inputSet[j];
                }
            }
            int temp = COST(inputSet[i], setToBePassedOnToNextCallOfCOST, setSize - 1);
            if (temp < min)
                min = temp, minindex = i;
        }
        for (int j = 0; j < setSize; j++) {
            if (inputSet[j] == inputSet[minindex])
                System.out.println("Cost saved by leaving node " + (j + 1) + " is " + min);
        }
        return min;
    }
    public void eval() {
        int k = 0;
        for (int i = 0; i < n; i++) {
            if (tour[i] == INF) {
                tour[i] = min;
                k = i;
            }
        }
        while (k != 0) {
            tour[k] = k;
            k = tour[k];
        }
        System.out.println("Tour: ");
        for (int i = 0; i < n - 1; i++) {
            System.out.print(tour[i] + " to ");
        }
        System.out.println(tour[n - 1] + " to ");
        finalCost = 0;
        for (int i = 0; i < n - 1; i++) {
            finalCost += weight[tour[i]][tour[i + 1]];
        }
        System.out.println("Cost: " + finalCost);
    }
}
```

if((weight[currentNode][inputSet[i]]+temp) < min)
    {
        min=weight[currentNode][inputSet[i]]+temp;
        minindex=inputSet[i];
    }
    }
return min;
}

public int MIN(int currentNode, int inputSet[], int setSize)
{
    if(setSize==0)
    return weight[currentNode][0];
    int min=INF, minindex=0;
    intsetToBePassedOnToNextCallOfCOST[]=new int[n-1];
    for(int i=0;i<setSize;i++)//considers each node of inputSet
    {
        int k=0;
        for(int j=0;j<setSize;j++)
        {
            if(inputSet[i]!=inputSet[j])
                setToBePassedOnToNextCallOfCOST[k++]=inputSet[j];
        }
        int temp=COST(inputSet[i],setToBePassedOnToNextCallOfCOST,setSize-1);
        if((weight[currentNode][inputSet[i]]+temp) < min)
        {
            min=weight[currentNode][inputSet[i]]+temp;
            minindex=inputSet[i];
        }
    }
return minindex;
}

public void eval()
{
    int dummySet[]=new int[n-1];
    for(int i=1;i<n;i++)
        dummySet[i-1]=i;
    finalCost=COST(0,dummySet,n-1);
    constructTour();
}

public void constructTour()
{
    int previousSet[]=new int[n-1];
    int nextSet[]=new int[n-2];
    for(int i=1;i<n;i++)
        previousSet[i-1]=i;
    int setSize=n-1;
    tour[0]=MIN(0,previousSet,setSize);
    for(int i=1;i<n-1;i++)
    {
        int k=0;
        for(int j=0;j<setSize;j++)
        {
            if(tour[i-1]!=previousSet[j])
nextSet[k++] = previousSet[j];
}
--setSize;
tour[i] = MIN(tour[i-1], nextSet, setSize);
for(int j = 0; j < setSize; j++)
previousSet[j] = nextSet[j];
}
display();
}
public void display()
{
    System.out.println();
    System.out.println("The tour is 1-");
    for(int i = 0; i < n-1; i++)
        System.out.println((tour[i]+1)+"-1");
    System.out.println("1");
    System.out.println("The final cost is "+finalCost);
}
}
class TSPExp
{
    public static void main(String args[])
    {
        TSP obj = new TSP();
    }
}
OUTPUT:
Enter no. of nodes:=>5
Enter weight of 1 to 2:=>4
Enter weight of 1 to 3:=>6
Enter weight of 1 to 4:=>3
Enter weight of 1 to 5:=>7
Enter weight of 2 to 1:=>3
Enter weight of 2 to 3:=>1
Enter weight of 2 to 4:=>7
Enter weight of 2 to 5:=>4
Enter weight of 3 to 1:=>7
Enter weight of 3 to 2:=>4
Enter weight of 3 to 4:=>3
Enter weight of 3 to 5:=>6
Enter weight of 4 to 1:=>8
Enter weight of 4 to 2:=>5
Enter weight of 4 to 3:=>3
Enter weight of 4 to 5:=>2
Enter weight of 5 to 1:=>4
Enter weight of 5 to 2:=>3
Enter weight of 5 to 3:=>2
Enter weight of 5 to 4:=>1

Starting node assumed to be node 1.
The tour is 1-2-3-4-5-1
The final cost is 14
WEEK-5

COVARIANCE

Plot the correlation plot on dataset and visualize giving an overview of relationships among data on soya bins data. Analysis of covariance: variance (ANOVA), if data have categorical variables on iris data.

PROBLEM DEFINATION:

a) Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris data.

SOURCE CODE:

```r
Image(x=seq(dim(x)[2])
Y<-seq(dim(y)[2])
Z=COR,xlab="x column",ylab="y column")
Library(gt3charts)
Data(iris)
Iris$species<-NULL
Iplotcorr(iris,roder=TRUE

PROBLEM DEFINATION:

b) Analysis of covariance: variance (ANOVA), if data have categorical variables on iris data.

SOURCE CODE:

```r
library(ggplot2)
data(iris)
str(iris)
ggplot(data=iris,aes(x=sepal.length,y=petal.length))+geom_point(size=2,colour="black")+geom_point(size=1,colour="white")+geom_smooth(aes(colour="black"),method="lm")+ggtitle("sepal.lengthvs petal.length")+xlab("sepal.length")+ylab("petal.length")+these(legend.position="none")
```
DATA FITTING BY REGRESSION

PROBLEM DEFINATION

Apply multiple regressions, if data have a continuous independent variable. Apply on above dataset.

SOURCE CODE:

```r
> mydata$rank <- factor(mydata$rank)
> mylogit <- glm(admit ~ gre + gpa + rank, data = mydata, family = "binomial")
> summary(mylogit)
```
> mydata$rank <- factor(mydata$rank)
> mylogit <- glm(admit ~ gre + gpa + rank, data = mydata, family = "binomial")
> summary(mylogit)

Call:
  glm(formula = admit ~ gre + gpa + rank, family = "binomial",
      data = mydata)

Deviance Residuals:
     Min       1Q   Median       3Q      Max
-1.6268  -0.8662  -0.6388   1.1490   2.0790

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.989979   1.139951  -3.500  0.000465 ***
gre          0.002264   0.001094   2.070  0.038465 *
gpa          0.804038   0.331819   2.423  0.015386 *
rank2       -0.675443   0.316490  -2.134  0.032829 *
rank3       -1.340204   0.345306  -3.881  0.000104 ***
rank4       -1.551464   0.417832  -3.713  0.000205 ***

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 499.98  on 399  degrees of freedom
Residual deviance: 458.52  on 394  degrees of freedom
AIC: 470.52

Number of Fisher Scoring iterations: 4
WEEK-7

DESCRIPTION: STUDY AND ANALYSIS OF CRISP LOGIC.

CRISP LOGIC :-

In older days, Crisp logic were used to handle the problem of binary value ie. 0 and 1.

Crisp logic is also known as traditional, conventional or binary logic.

Crisp logic have two valued logic first is true and other is false. Crisp logic is based on the reasoning which is exact and fixed. It is based on the logic of completely true and completely false.

We can defined completely true as one (1) and completely false as zero (0).

![Crisp Logic Diagram]

PROGRAM:

```java
import net.sourceforge.jFuzzyLogic.FIS;
import net.sourceforge.jFuzzyLogic.rule.FuzzyRuleSet;
/**
 * Test parsing an FCL file
 * @author pcingola@users.sourceforge.net
 */
public class TestTipper {
    public static void main(String[] args) throws Exception {
        // Load from 'FCL' file
        String fileName = "fcl/tipper.fcl";
        FIS fis = FIS.load(fileName, true);
        // Error while loading?
        if (fis == null) {
            System.err.println("Can't load file: " + fileName + ")
            return;
        }
    }
}
```
// Show
fis.chart();
    // Set inputs
fis.setVariable("service", 3);
fis.setVariable("food", 7)
    // Evaluate
fis.evaluate();
    // Show output variable's chart
fis.getVariable("tip").chartDefuzzifier(true);
    // Print ruleSet
System.out.println(fis);
}
Week -8

DESCRIPTION: Write a program to implement Hebb’s rule

PROGRAM:

```c
#include<iostream.h>
#include<conio.h>

void main()
{
    float n,w,t,net,div,a,al;
    cout<"consider o single neuron percetron with a single i/p";
    cin>\n    cout<"enter the learning coefficent";
    cin>\n    for (i=0;i<10;i++)
    {
        net = x+w;
        if(wt<0)
            a=0;
        else
            a=1;
        div=at+a+w;
        w=w+div;
        cout<"i+1 in fraction are i"<<a<<"change in weight"<<dw<<"adjustment at="<<w;
    }
}
```
OUTPUT:
Consider 0 single neuron perceptron with a single i/p
Enter the learning Co-efficient
i+1 in fraction are i1change in weight2adjustment at 3i+1
In fraction are i1change in weight4adjustment at i+1
In fraction are i1change in weight8adjustment at i= 15i+1
In fraction are i1change in weight16adjustment at=3li+1
In fraction are i1change in weight32adjustment at=63i+1
In fraction are i1change in weight64adjustment at=127i+1
In fraction are i1change in weight128adjustment at=255i+1
In fraction are i1change in weight256adjustment at=511i+1
In fraction are i1change in weight512adjustment at=1023i+1
In fraction are i1change in weight1024adjustment at=2047i+1
Consider 0 single neuron perceptron with a single i/p
DESCRIPTION: Write a program to implement of delta rule.

PROGRAM:

```cpp
#include<iostream.h>
#include<conio.h>
void main()
{
    clrscr();
    float input[3],d,weight[3],delta;
    for(int i=0;i<3 ; i++)
    {
        cout<<"\n initilize weight vector "<<i<<"\"t";  
        cin>>input[i];
    }
    cout<<"\n enter the desired output\t";  
    cin>>d;
    do
    {
        del=d-a;
        if(del<0)
            for(i=0 ;i<3 ;i++)
                w[i]=w[i]-input[i];
        else if(del>0)
            for(i=0;i<3;i++)
                weight[i]=weight[i]+input[i];
        for(i=0;i<3;i++)
        {
            val[i]=del*input[i];
            weight[i+1]=weight[i]+val[i];
        }
        cout<<"value of delta is "<<del;
        cout<<"\n weight have been adjusted";
    }while(del ≠ 0)
    if(del=0)
        cout<<"\n output is correct";
}
```
OUTPUT:
  Initialize weight vector 1
  Initialize weight vector 2
  Initialize weight vector 3
  Enter desired output
DESCRIPTION: Write a program to implement logic gates.

PROGRAM:
```
#include <iostream>
int main()
{
    char menu; //Menu control variable
    int result; //final output variable
    int dataValue1;
    int dataValue2;
    cout<< "enter your Boolean operator code: (A,O,N,X): ";
    cin>> menu;
    switch (menu) //Menu control variable
    {
    case 'A':
        cout<< "Enter first Boolean value:";
        cin>> dataValue1;
        cout<< "Enter second Boolean value:";
        cin>> dataValue2;
        if(dataValue1 == 1 && dataValue2 == 1)
        {
            result = 1;
        }
        else
        {
            result = 0;
        }
        cout<< "show result:" << result;
        break;
    case 'O':
        cout<< "Enter first Boolean value:";
        cin>> dataValue1;
        cout<< "Enter second Boolean value:";
        cin>> dataValue2;
        if(dataValue1 == 1 || dataValue2 == 1)
        {
            result = 1;
        }
        else
        {
            result = 0;
        }
        cout<< "show result:" << result;
        break;
    case 'N':
        cout<< "Enter first Boolean value:";
        cin>> dataValue1;
```
```
result = !dataValue1;
cout << "show result:" << result;
break;
case 'X':
cout << "Enter first Boolean value:";
cin >> dataValue1;
cout << "Enter second Boolean value:";
cin >> dataValue2;
if(dataValue1 == !dataValue1)
{
    result = 1;
} else
{
    result = 0;
}
cout << "show result:" << result;
break;
default:
result = 0;
break;
}//end switch
cin.ignore(2);
return 0;
}//end main
OUTPUT:

Enter your Boolean operator code : (A,O,N,X) : A
Enter First Boolean value : 1
Enter Second Boolean value : 1
Show Result : 1

Enter your Boolean operator code : (A,O,N,X) : O
Enter First Boolean value : 1
Enter Second Boolean value : 0
Show Result : 1
WEEK-10

Description: SVM Classification

Program:

```c++
#include<iostream.h>
#include<conio.h>

Void main() {
    clrscr();
    int in[3],d,w[3],a=0;
    for(int i=0;i<3;i++)
    {
        cout<<"initialize the weight vector w"<<i;
        cin>>w[i] 
    }
    for(i=0;i<3;i++)
    {
        cout<<"enter the input vector i"<<i;
        cin>>in[i];
    }
    cout<<"enter the desined output";
    cin>>d;
    int ans=1;
    while(ans= = 1)
    {
        for (a= 0, i==0;i<3;i++)
        {
            a = a + w[i] * in[i];
        }
        clrscr();
        cout<<"desired output is"<<d;
        cout<<"actual output is "<<a;
        int e;
        e=d-a;
        cout<<"error is "<<e;
        cout<<"press 1 to adjust weight else 0";
        cin>>ans;
        if (e<0)
        {
            for(i=0;i<3;i++)
            {
                w[i]=w[i]-1;
            }
        }
        else if (e>0)
        {
            for(i=0;i<3;i++)
            {
                w[i]=w[i]+1;
            }
        }
        getch();
    }
```
OUTPUT:
  Desired output is 2
  Actual output is 17
  Error is -15
  Press 1 to adjust weight else 0