**Aim:** 1) Run the Regression code algorithm on training samples.

2) Get the learned linear function.

3) Get the predicted results for the user inputs.

**Description:** Method of least squares is used to fit the experimental/trained data to linear straight line.

**Procedure:**

1. Linreg.cpp and linreg.h are downloaded from blackboard and modified to remove all the compilation errors. (especially in including header files).
2. Linear\_assignment2.cpp file takes the values of experimental/trained data in three different forms like x and y arrays or 2D point form or empty instance form.
3. Entered trained values are used to calculate the Learned linear function.
4. Learned linear function is displayed on console.
5. User is asked to enter the number of inputs for which values are to be predicted.
6. For each user input value, predicted result is calculated by the linreg\_assignment 2.cpp program and displayed on to the console.
7. Run The code as right click on c++ project>>run as local C\C++ Application.
8. Enter the number of user inputs and input values. Estimated values are displayed in console.

**Code :**

//============================================================================

// Name : MachineLearning Assignment-2

// Author : Harshitha Yendapally

// Version :1

// Copyright : Blackboard Submitted file to UTSA

// Description : Program is about learning the linear function and predicting

// unknown values.

//============================================================================

**#include** <iomanip>

**#include** "linreg.h"

**#include**<iostream>

**using** **namespace** std;

//Points are entered in array format.(one of the formats among 3 is choosen)

//Inputs gathered for training the system are entered

**double** x[] = {19.7,19.1,18.2,5.2,4.3,9.3,3.6,14.8,11.9,9.3,2.8,

9.9,15.4,2.7,10.6,16.6,11.4};

**double** y[] = {19.7,19.3,18.6,7.9,4.4,9.6,8.0,15.7,15.4,9.8,10.3,

11.2,16.8,5.1,12.2,18.9,12.2};

//Points entered in point format

Point2D p[] = { Point2D(19.7, 19.7), Point2D(19.1, 19.3), Point2D(18.2, 18.6),

Point2D(5.2, 7.9), Point2D(4.3, 4.4), Point2D(9.3, 9.6),

Point2D(3.6, 8.0), Point2D(14.8, 15.7), Point2D(11.9, 15.4),

Point2D(9.3, 9.8), Point2D(2.8, 10.3),Point2D(9.9, 11.2),

Point2D(15.4, 16.8),Point2D(2.7, 5.1),Point2D(10.6, 12.2),

Point2D(16.6, 18.9),Point2D(11.4, 12.2)};

**int** **main**()

{

cout << "Learned Linear Function Analysis" << **endl**;

LinearRegression lr\_xy(x, y, 17); // learned linear function calculation

cout << "Number of x,y pairs = " << lr\_xy.items() << **endl**;

cout << lr\_xy << **endl**;

cout << "Coefficient of Determination = "

<< lr\_xy.getCoefDeterm() << **endl**;

cout << "Coefficient of Correlation = "

<< lr\_xy.getCoefCorrel() << **endl**;

cout << "Standard Error of Estimate = "

<< lr\_xy.getStdErrorEst() << **endl**;

cout << "\nLearned Linear Function Analysis when inputs are entered in point-2D format." << **endl**;

LinearRegression lr\_2D(p, 17); // learned linear function calculation

cout << "Number of x,y pairs = " << lr\_2D.items() << **endl**;

cout << lr\_2D<< **endl**;

cout << "Coefficient of Determination = "

<< lr\_2D.getCoefDeterm() << **endl**;

cout << "Coefficient of Correlation = "

<< lr\_2D.getCoefCorrel() << **endl**;

cout << "Standard Error of Estimate = "

<< lr\_2D.getStdErrorEst() << **endl**;

cout << "\nLearned Linear Function Analysis when inputs are entered in empty instance format." << **endl**;

LinearRegression lr\_empt; // learned linear function calculation

**for** (**int** i = 0; i < 17; i++)

lr\_empt.addPoint(p[i]);

cout << "Number of x,y pairs = " << lr\_empt.items() << **endl**;

cout << lr\_empt<< **endl**;

cout << "Coefficient of Determination = "

<< lr\_empt.getCoefDeterm() << **endl**;

cout << "Coefficient of Correlation = "

<< lr\_empt.getCoefCorrel() << **endl**;

cout << "Standard Error of Estimate = "

<< lr\_empt.getStdErrorEst() << **endl**;

**int** i,n;

**double** in1,result;

//code to find the estimated values of test samples provided by user

cout << "enter number of user values for which estimated results are to be predicted"<<**endl**;

cin>>n;

**for**(i=0;i<n;i++){

cout <<"enter input "<<**endl**;

cin>>in1;

result=(in1\*(lr\_xy.getB()))+(lr\_xy.getA());

cout<<"predicted value for "<<in1<<" is "<<result<<**endl**;

cout<<"\n";

}

cout<<"end of the user values.program exited.";

**return** 0;

}

**Results:**

Learned Linear Function Analysis

Number of x,y pairs = 17

f(x) = 3.79973 + ( 0.814419 \* x )

Coefficient of Determination = 0.894356

Coefficient of Correlation = 0.945704

Standard Error of Estimate = 1.67252

Learned Linear Function Analysis when inputs are entered in point-2D format.

Number of x,y pairs = 17

f(x) = 3.79973 + ( 0.814419 \* x )

Coefficient of Determination = 0.894356

Coefficient of Correlation = 0.945704

Standard Error of Estimate = 1.67252

Learned Linear Function Analysis when inputs are entered in empty instance format.

Number of x,y pairs = 17

f(x) = 3.79973 + ( 0.814419 \* x )

Coefficient of Determination = 0.894356

Coefficient of Correlation = 0.945704

Standard Error of Estimate = 1.67252

enter number of user values for which estimated results are to be predicted

3

enter input

18.8

predicted value for 18.8 is 19.1108

enter input

15.6

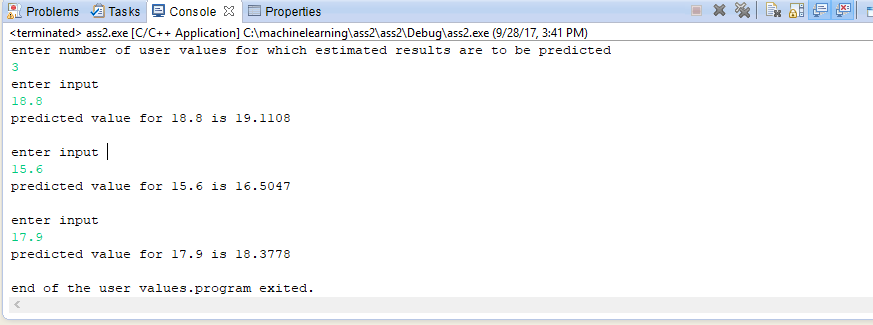
predicted value for 15.6 is 16.5047

enter input

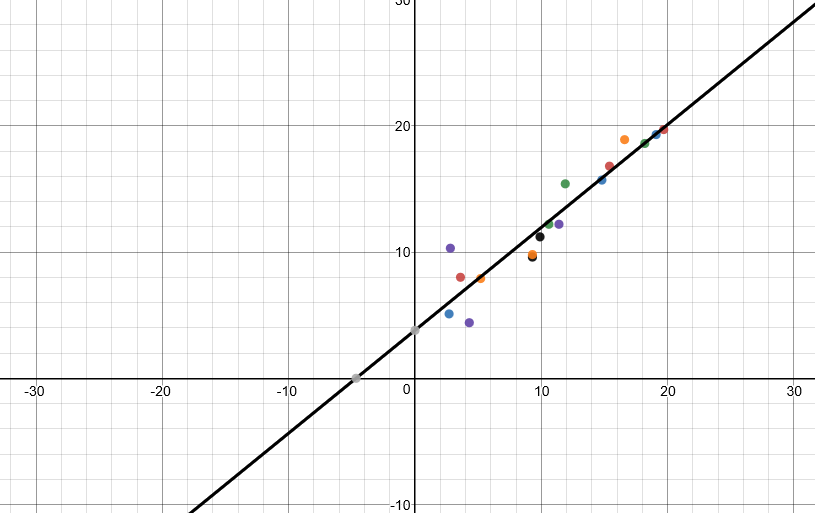
17.9

predicted value for 17.9 is 18.3778

end of the user values.program exited.



**Graph of Trained values and Learned linear function:**



**Graph of User inputs and predicted values:**

