Visvesvaraya National Institute of Technology (VNIT), Nagpur

Machine Learning

Lab Report

Submitted by :
Shubham Dayanand Patil (BT19ECE084)
Semester 7

Submitted to :
Dr. Saugata Sinha
(Course Instructor)
Department of Electronics and Communication Engineering,
VNIT Nagpur

Contents

Experiment-1

Abstract: The implementation and findings of linear regression using the pseudo-inverse approach and gradient descent approach are shown in the following report..

Introduction: Based on the value of another variable, linear regression analysis makes predictions about the value of one variable. The term "dependent variable" refers to the variable you want to predict. The independent variable is the one you're using to predict the value of the other variable.

Problem Statement :

Estimating the number of fatalities in relation to the body's alcohol consumption.

Method: Pseudo Inverse Method:

$$X = (A^T.A)^{-1}.A^T b$$

It is a method of doing linear regression analytically with a least squares cost function. We don't need to use Gradient Descent to determine the value. This approach is effective and time-saving when dealing with a data set with few features. The strategy is based on the concept of maxima and minima in mathematics, which states that the derivative and partial derivative of any function are equal to zero at the maxima and minima points. As a result, we extract the partial derivative of the Cost function with respect to each weight using the Normal Equation method and equalise it to zero.

Gradient Decent

The Gradient Descent approach finds the best-fit line for a given training dataset with less iterations. The least amount of errors will be returned for each given m and c combination (MSE). The best fit line will be provided by that m/c combination.

Output: Pseudo Inverse Method : For Train Data :

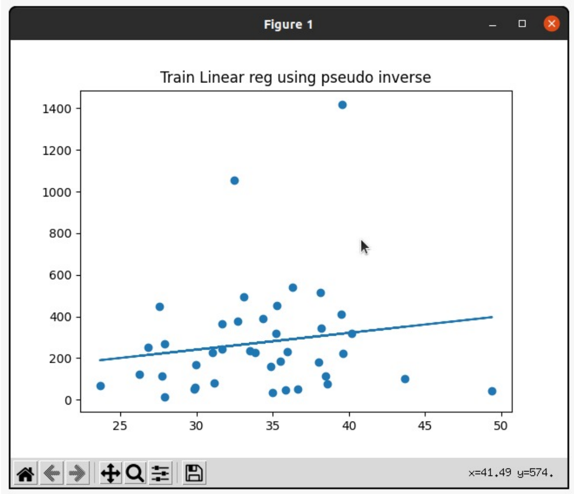


Figure 1: Plot of the linear curve and the train data

For Test Data :

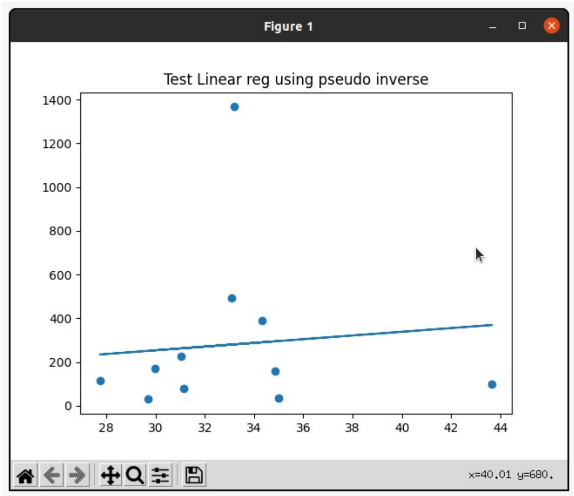


Figure 2: Plot of the linear curve and the test data

Gradient descent method

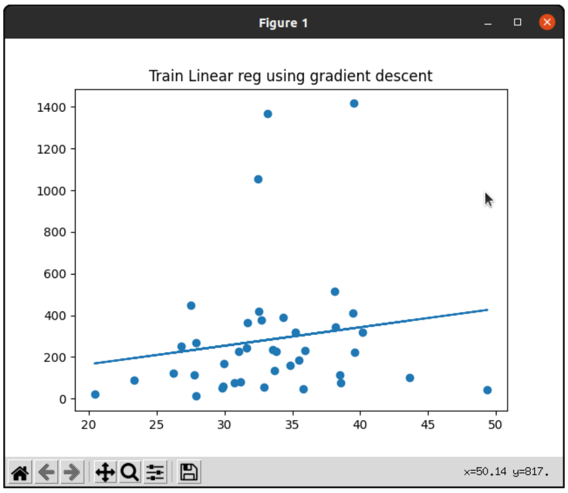


Figure 3: Plot of the linear curve and the train data

For Test Data :

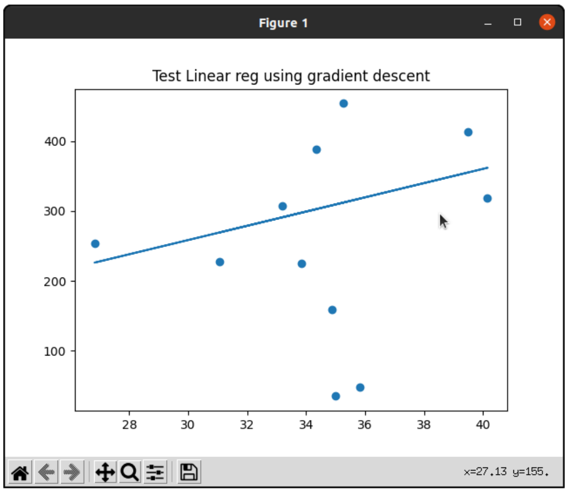
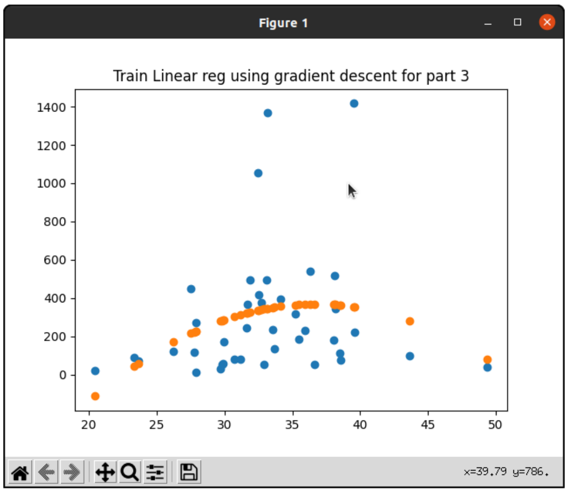


Figure 4: Plot of the linear curve and the test data

With different type of input output relationship



Observations and Discussions:

- Linear-regression models are quite straightforward and offer a simple mathematical framework for producing predictions. Several different forms of input may be utilised with linear regression.
- When dealing with small data sets, the pseudo inverse method works well. The graphs show that the linear line successfully matches the data.
- The graph for the test data validates the above assertion.
- The main motivation for using gradient descent in linear regression is the cost-effectiveness of the method in some situations due to the complexity of the computing process (faster).

Conclusion:

1. As a result, we were able to build and evaluate the linear regression using the gradient descent technique and the pseudo inverse approach.
2. We also examined linear regression for various input-output configurations.

Appendices: Code :

```

import pandas as pd
import numpy as np
from scipy.io import loadmat
import os
import matplotlib.pyplot as plt
from numpy.random.mtrand import shuffle
from sklearn.linear_model import LinearRegression

def grad_desc(Xs, Ys, rate = 0.001, iterations = 100):
    w = np.random.rand(Xs.shape[1],1)
    for _ in range(iterations):
        errors =  Ys - np.dot(Xs,w)
        grad = (Xs.T).dot(errors)
        w = w - rate*grad
    return w

def BT19ECE084_linreg(filename):

    # filename = r"/content/Matlab_accidents.mat"
    dff,head = suffle_divide(filename)
    df = dff.drop([head[i] for i in ...
        [0,1,2,3,4,5,6,7,8,9,10,13,14,15,16] ],axis=1)

    ratio = 0.8
    data_numpy = df.to_numpy()
    np.random.shuffle(data_numpy)

    train_data = data_numpy[:int(len(data_numpy) * ratio), :]
    test_data = data_numpy[int(len(data_numpy) * ratio):, :]
# print(train_data.shape , test_data.shape)

    Y_train = train_data[:,0]
    X_train = train_data[:,1]
    Y_train = Y_train.reshape( len(Y_train) , 1)
    X_train = X_train.reshape( len(X_train) , 1)
    # pseudo inverse
    W = np.dot( np.dot( np.linalg.inv( np.dot(X_train.T, ...
        X_train)) , X_train.T ) , Y_train )

    Y_pre = W*X_train

    plt.scatter(list(X_train.T[0]) ,list(Y_train.T[0]) )
    plt.plot(list(X_train.T[0]) ,list(Y_pre.T[0]) )
    plt.title("Train Linear reg using pseudo inverse")
    plt.show()
```

```
    # testing lin reg using pseudo inverse
    # testing
    Y_test = test_data[:,0]
    X_test = test_data[:,1]
    Y_test = Y_test.reshape( len(Y_test) , 1)
    X_test = X_test.reshape( len(X_test) , 1)

    Y_pre = W*X_test

    plt.scatter(list(X_test.T[0]) ,list(Y_test.T[0]) )
    plt.plot(list(X_test.T[0]) ,list(Y_pre.T[0]) )
    plt.title("Test Linear reg using pseudo inverse")
    plt.show()

    # using gradient descent
    reg = LinearRegression()
    reg.fit(X_train, Y_train)

    Y_pre = reg.predict(X_train)

    plt.scatter(list(X_train.T[0]) ,list(Y_train.T[0]) )
    plt.plot(list(X_train.T[0]) ,list(Y_pre.T[0]) )
    plt.title("Train Linear reg using gradient descent")
    plt.show()

    # testing lin reg using gradient descent

    Y_pre = reg.predict(X_test)

    plt.scatter(list(X_test.T[0]) ,list(Y_test.T[0]) )
    plt.plot(list(X_test.T[0]) ,list(Y_pre.T[0]) )
    plt.title("Test Linear reg using gradient descent")
    plt.show()

    # using another type of input output relationship

    X_sq = X_test**2
    X_new = np.concatenate((X_train,X_train**2),axis=1)
    reg.fit(X_new, Y_train)
    Y_pre = reg.predict(X_new)

    plt.scatter(list(X_new.T[0]) ,list(Y_train.T[0]) )
    plt.plot(list(X_new.T[0]) ,list(Y_pre.T[0]) )
    plt.title("Train Linear reg using gradient descent for part 3")
    plt.show()
```