Program - 5

Program to learn Linear Regression using Gradient Descent method

# Linear Regression with Gradient Descent

# Making the imports

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

plt.rcParams['figure.figsize'] = (12.0, 9.0)

# Preprocessing Input data

data = pd.read\_csv('data.csv')

X = data.iloc[:, 0]

Y = data.iloc[:, 1]

plt.scatter(X, Y)

plt.show()

# Building the model

w1\_old = 0.1

w0\_old= 0.1

w1\_new = 0

w0\_new= 0

alpha = 0.0001  # The learning Rate

epochs = 1000  # The number of iterations to perform gradient descent

n = float(len(X)) # Number of elements in X

# Performing Gradient Descent

while(abs(w1\_new – w1\_old) + abs(w0\_new – w0\_old))>0.01

    w1\_old=w1\_new

w0\_old = w0\_new

Y\_pred = w1\_old\*X + w0\_old  # The current predicted value of Y

    D\_w1 = (-2/n) \* sum(X \* (Y - Y\_pred))  # Derivative wrt w1

    D\_w0 = (-2/n) \* sum(Y - Y\_pred)  # Derivative wrt w0

    w1\_new = w1\_old - alpha \* D\_w1  # Update w1

    w0\_new= w0\_old - alpha \* D\_w0  # Update c

print (w1, w0)

# Making predictions

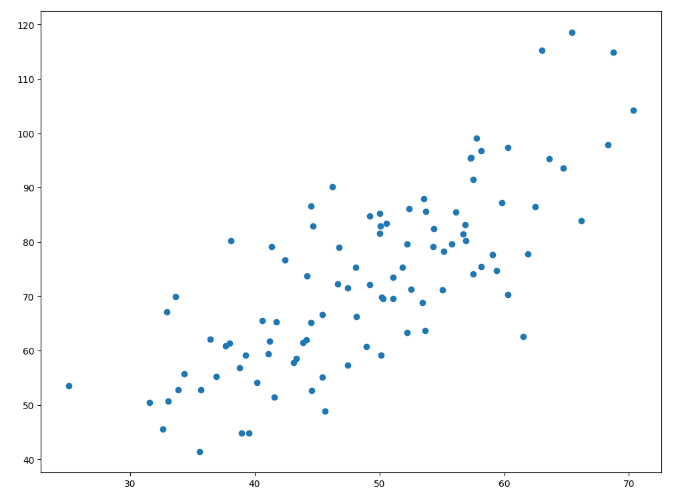
Y\_pred = w1\*X + w0

plt.scatter(X, Y)

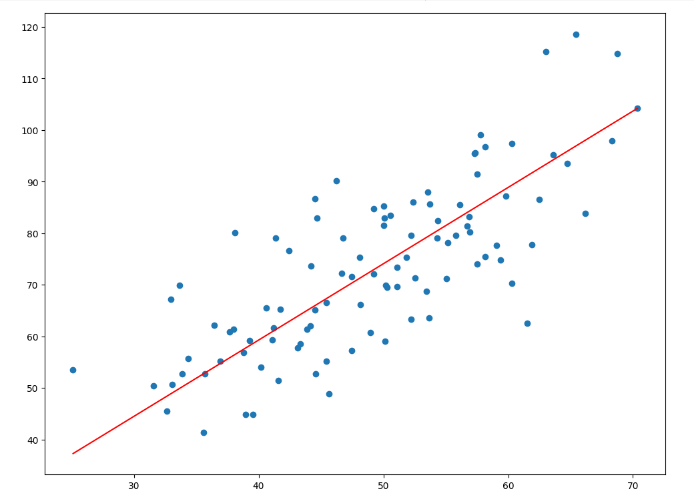
plt.plot([min(X), max(X)], [min(Y\_pred), max(Y\_pred)], color='red')  # regression line

plt.show()

OUTPUT



1.4796491688889395 0.10148121494753734



Assignment:

Write the inference for the linear regression model by varying ‘m’ and ‘c’ values (with reference to the regression line obtained)

Theory behind Linear Regression using Gradient Descent method:

**Linear Regression**

In statistics, linear regression is a linear approach to modelling the relationship between a dependent variable and one or more independent variables. Let **X** be the independent variable and **Y** be the dependent variable. Let us define a linear relationship between these two variables as follows:

where **m** is the slope of the line and **c** is the y intercept.

This equation is used to train our model with a given dataset and predict the value of **Y** for any given value of **X**. The challenge is to determine the value of **m** and **c**, such that the line corresponding to those values is the best fitting line or gives the minimum error.

**Loss Function**

The loss is the error in predicted value of **m** and **c**. Goal is to minimize this error to obtain the most accurate value of **m** and **c**.  
Loss is calculated using the Mean Squared Error function.

There are three steps in this function:

1. Find the difference between the actual Y and predicted Y value () , for a given X.
2. Square this difference.
3. Find the mean of the squares for every value in X.

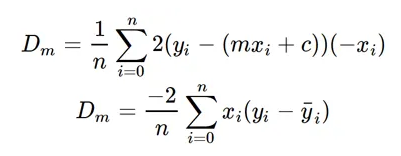
**The Gradient Descent Algorithm**

Gradient descent is an iterative optimization algorithm to find the minimum of a function. In this program, GD function is used as Loss Function.

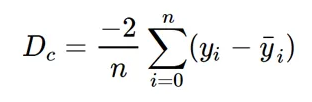
To understand the concept of Gradient Descent, imagine a valley and a person with no sense of direction who wants to get to the bottom of the valley. He goes down the slope and takes large steps when the slope is steep and small steps when the slope is less steep. He decides his next position based on his current position and stops when he gets to the bottom of the valley which was his goal.

Let’s try applying gradient descent to **m** and **c** and approach it step by step:

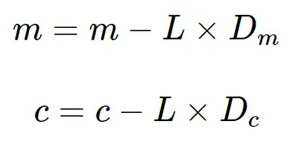
1. Initially let m = 0 and c = 0. Let L be the learning rate. This controls how much the value of **m** changes with each step. L could be a small value like 0.0001 for good accuracy.
2. Calculate the partial derivative of the loss function with respect to m, and plug in the current values of x, y, m and c in it to obtain the derivative value **D**.



Dₘ is the value of the partial derivative with respect to **m**. Similarly, find the partial derivative with respect to **c**, :



1. Now we update the current value of **m** and **c** using the following equation:



1. Repeat this process until the loss function is a very small value or ideally 0 (which means 0 error or 100% accuracy). The value of **m** and **c** that are left with are the optimum values.

With these, **m** can be considered the current position of the person. **D** is equivalent to the steepness of the slope and **L** can be the speed with which he moves. Now the new value of **m** that is used to calculate using the above equation will be his next position, and **L×D** will be the size of the steps he will take. When the slope is steeper (**D** is more) he takes longer steps and when it is less steep (**D** is less), he takes smaller steps. Finally, he arrives at the bottom of the valley which corresponds to loss = 0.  
Now with the optimum value of **m** and **c**, model is ready to make predictions.

**Reference:**

1. [Linear Regression using Gradient Descent | by Adarsh Menon | Towards Data Science](https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931)