Experiment No. 2 - Implement a simple linear regressor with a single neuron model

Linear Regression

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.

Loss Function

Mean Squared Error Equation

$$E = 1/n \sum_{i=0}^{n} (y_i - \bar{y_i})^2$$

$$E = 1/n \sum_{i=0}^{n} (y_i - (mx_i + c))^2$$

Gradient Descent

Derivative of loss function with respect to m

$$D_m = 1/n \sum_{i=0}^{n} 2(y_i - (mx_i + c))(-x_i)$$

$$D_m = -2/n \sum_{i=0}^{n} (x_i)(y_i - \bar{y_i})$$

Derivative of loss function with respect to c

$$D_c = -2/n \sum_{i=0}^{n} (y_i - \bar{y_i})$$

Update the current value of m and c

$$m = m - L * D_m$$

$$c = c - L * D_c$$

- **Data** - data.csv x,y points
- **Task**
- **Mathematical Model** - Linear Regression
- **Loss Function** - Mean Squared Error
- **Learning Algorithm** - Gradient Descent
- **Model Evaluation**

Algorithm

1. Import Library
2. Load and plot the data
3. Model using Gradient Descent
4. Print the correct values of slope, intercept
5. Plot the linear fit on the data plot

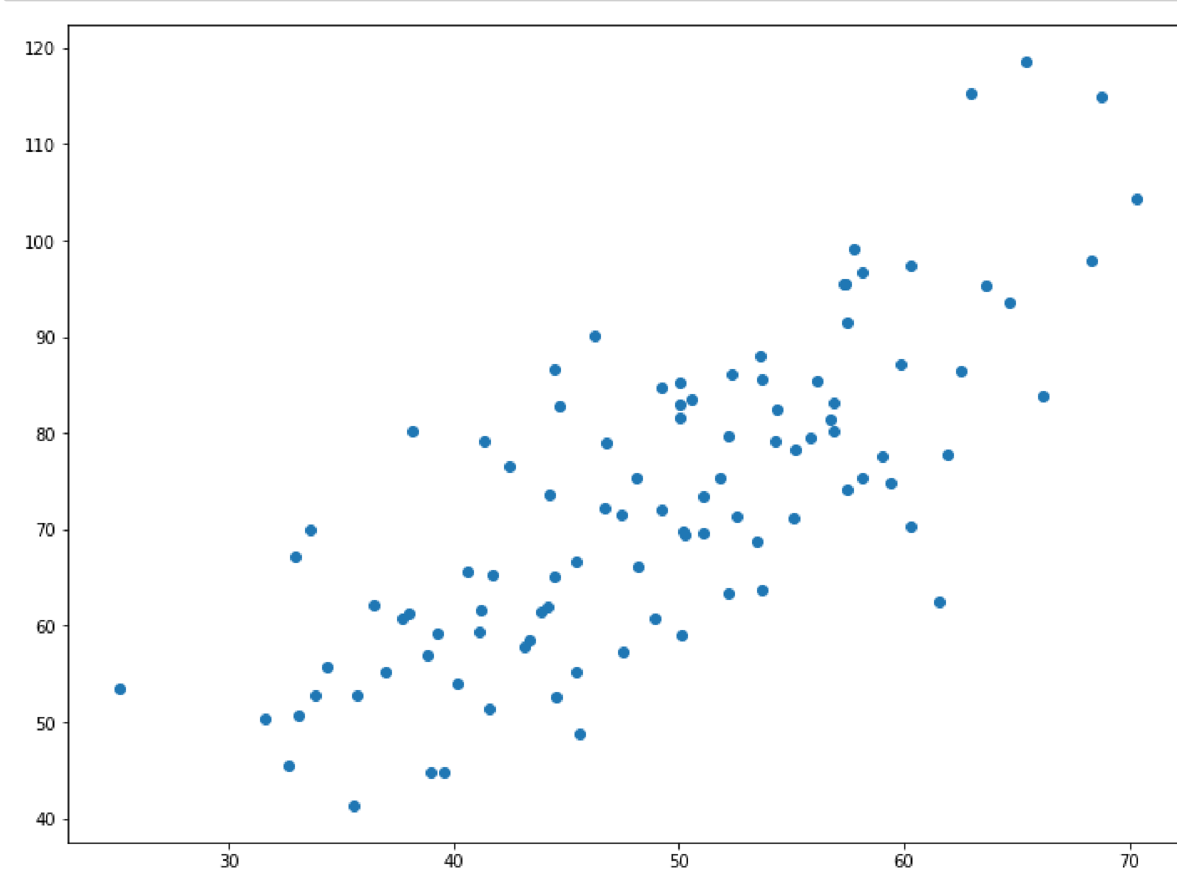
1. Import Library

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

2. Load and plot the data

In [2]:
```
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()
```



3. Model using Gradient Descent

In [3]:
```
def linear_regression_model(X, Y):
    # Building the model
    m = 0
    c = 0

    L = 0.0001  # The learning Rate
    epochs = 1000  # The number of iterations to perform gradient desc
ent

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

    #plt.show()

    # Performing Gradient Descent
    for i in range(epochs):
        Y_pred = m*X + c  # The current predicted value of Y
        D_m = (-2/n) * sum(X * (Y - Y_pred))  # Derivative wrt m
        D_c = (-2/n) * sum(Y - Y_pred)  # Derivative wrt c
        m = m - L * D_m  # Update m
        c = c - L * D_c  # Update c
        #print(i)

        #Below two lines are optional
    #     plt.scatter(X, Y)
    #     plt.plot([min(X), max(X)], [min(Y_pred), max(Y_pred)], color
='red') # predicted

    return m, c
```

4. Print the correct values of slope, intercept

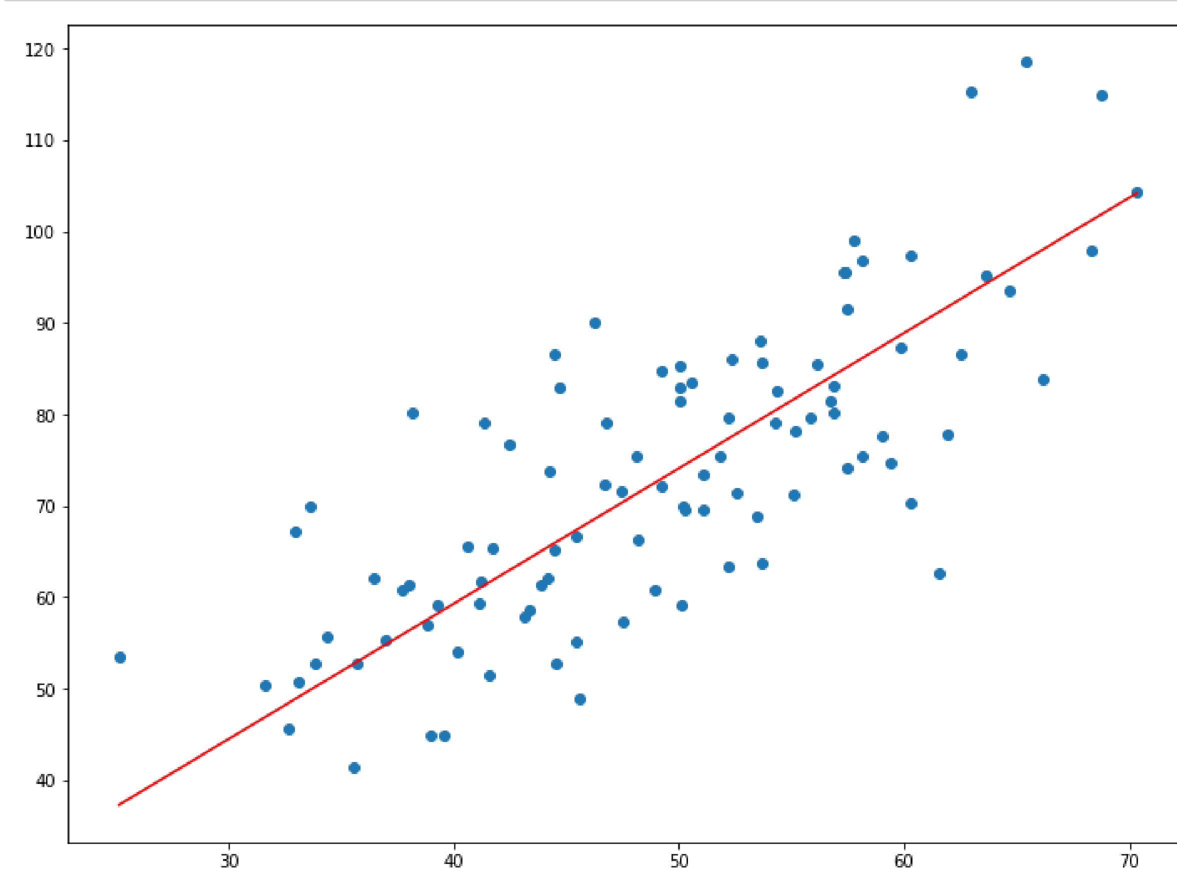
In [4]:
```
m, c = linear_regression_model(X, Y)
print('m =', m, 'c =', c)
```

```
m = 1.4796491688889395 c = 0.10148121494753726
```

5. Plot the linear fit on the data plot

In [5]:
```
# Making predictions
Y_pred = m*X + c

plt.scatter(X, Y)
plt.plot([min(X), max(X)], [min(Y_pred), max(Y_pred)], color='red') #
 predicted
plt.show()
```



Questions

1. Explain concept of linear regression. What are types of Regression?

Ans:- Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things:
(a) does a set of predictor variables do a good job in predicting an outcome (dependent) variable?
(b) Which variables in particular are significant predictors of the outcome variable, and in what way do they–indicated by the magnitude and sign of the beta estimates–impact the outcome variable?

These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. The simplest form of the regression equation with one dependent and one independent variable is defined by the formula

$$y = x * b + c$$

where y = estimated dependent variable score,
c = constant,
b = regression coefficient,
and x = score on the independent variable.

Types of Linear Regression
Simple linear regression
1 dependent variable (interval or ratio), 1 independent variable (interval or ratio or dichotomous)

Multiple linear regression
1 dependent variable (interval or ratio) , 2+ independent variables (interval or ratio or dichotomous)

Logistic regression
1 dependent variable (dichotomous), 2+ independent variable(s) (interval or ratio or dichotomous)

Ordinal regression
1 dependent variable (ordinal), 1+ independent variable(s) (nominal or dichotomous)

Multinomial regression
1 dependent variable (nominal), 1+ independent variable(s) (interval or ratio or dichotomous)

Discriminant analysis
1 dependent variable (nominal), 1+ independent variable(s) (interval or ratio)

2. Compare linear & non – linear regression

Ans:- Linear regression requires a linear model. No surprise, right? But what does that really mean?

A model is linear when each term is either a constant or the product of a parameter and a predictor variable. A linear equation is constructed by adding the results for each term.

While a linear equation has one basic form, nonlinear equations can take many different forms. The easiest way to determine whether an equation is nonlinear is to focus on the term “nonlinear” itself. Literally, it’s not linear. If the equation doesn’t meet the criteria above for a linear equation, it’s nonlinear.

That covers many different forms, which is why nonlinear regression provides the most flexible curve-fitting functionality.

3. What is linear regression? Describe with suitable example, how regression helps to predict the output for test sample.

Ans:- 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.

We will use below question as an example

4. Using regression analysis fit the linear model to the given data
X= [17,13,12,15,16,14,16,17,18,19]
Y= [94,70,59,80,92,65,87,95,99,105]
Predict the output for X = 12.5.

Ans:- Let's predict the output using Linear Regression Model

In [6]:
```
# Load the data
X = [17,13,12,15,16,14,16,17,18,19]
Y = [94,70,59,80,92,65,87,95,99,105]
X_data = pd.Series(X)
Y_data = pd.Series(Y)
```

In [7]:
```
# Train the model using given data
m, c = linear_regression_model(X_data, Y_data)
print(m, c)
```

```
5.397014194955672 0.2588921363544246
```

In [8]:
```
# prediction
X_pred = 12.5
Y_pred = m*X_pred + c

print(Y_pred)
```

```
67.72156957330031
```

5. Explain over fitting. What is the reason for over fitting?

Ans:- Overfitting refers to a model that models the training data too well. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.

References

[1] - https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931

Author Name:- Hemant Ghuge
LinkedIn:- https://www.linkedin.com/in/hemantghuge/
GitHub:- https://github.com/HemantGorakshGhuge