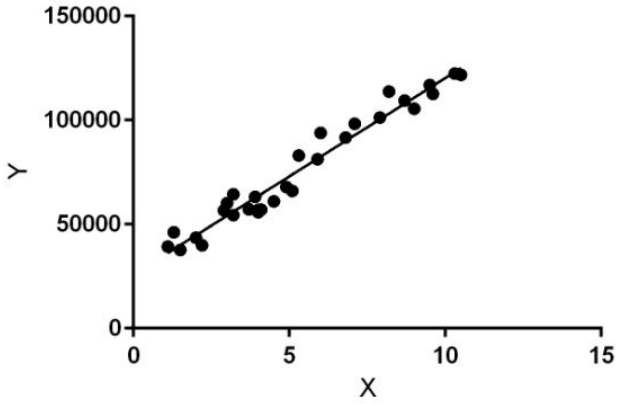
Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

JUNAID GIRKAR
60004190057
TE COMPS A4

EXPERIMENT - 1

AIM: Implement Linear Regression algorithm with gradient descent.

THEORY:
Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.



Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.
In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Hypothesis function for Linear Regression :

$$y = \theta_1 + \theta_2.x$$

While training the model we are given :
x: input training data (univariate – one input variable(parameter))
y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.
θ1: intercept
θ2: coefficient of x

Once we find the best θ1 and θ2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

How to update θ1 and θ2 values to get the best fit line ?

Cost Function (J):
By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ1 and θ2 values, to reach the best value that minimize the error between predicted y value (pred) and true y value (y).

$$minimize \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

$$J = \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

Cost function(J) of Linear Regression is the Root Mean Squared Error (RMSE) between predicted y value (pred) and true y value (y).

GRADIENT DESCENT

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Now,

$$\frac{\partial}{\partial \theta} J_\theta = \frac{\partial}{\partial \theta} \frac{1}{2m} \sum_{i=1}^{m} \left[h_\theta(x_i) - y_i\right]^2$$

$$\frac{\partial}{\partial \theta} J_\theta = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i\right) \cdot \frac{\partial}{\partial \theta_j} \left(\theta x_i - y_i\right)$$

$$\frac{\partial}{\partial \theta} J_\theta = \frac{1}{m} \sum_{i=1}^{m} \left[\left(h_\theta(x_i) - y\right) x_i\right]$$

Therefore,

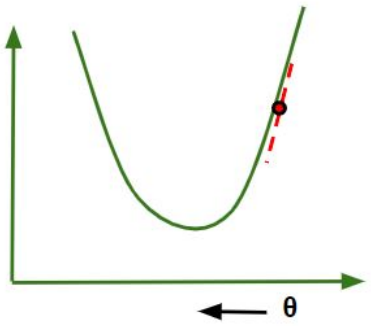
$$\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} \left[\left(h_\theta(x_i) - y_i\right) x_i\right]$$

-> **θj** : Weights of the hypothesis.
-> **hθ(xi)** : predicted y value for ith input.

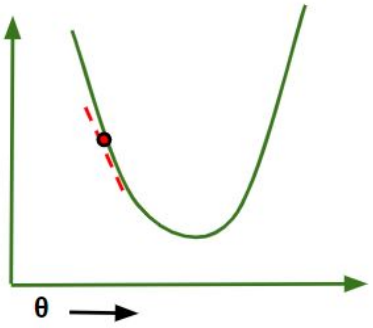
-> **j** : Feature index number (can be 0, 1, 2, ..., n).
-> **α** : Learning Rate of Gradient Descent.

We graph cost function as a function of parameter estimates i.e. parameter range of our hypothesis function and the cost resulting from selecting a particular set of parameters. We move downward towards pits in the graph, to find the minimum value. The way to do this is taking derivative of cost function as explained in the above figure. Gradient Descent step-downs the cost function in the direction of the steepest descent. The size of each step is determined by parameter α known as Learning Rate.
In the Gradient Descent algorithm, one can infer two points :

- **If slope is +ve** : θj = θj – (+ve value). Hence the value of θj decreases.

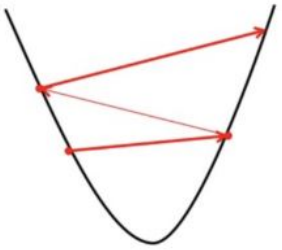


- If slope is -ve : θj = θj – (-ve value). Hence value of θj increases.



The choice of correct learning rate is very important as it ensures that Gradient Descent converges in a reasonable time. :

- If we choose **α to be very large**, Gradient Descent can overshoot the minimum. It may fail to converge or even diverge.



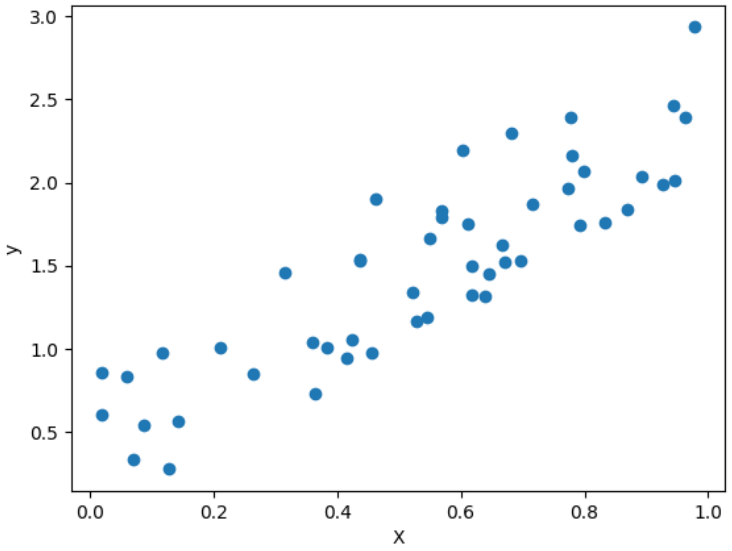
- If we choose α to be very small, Gradient Descent will take small steps to reach local minima and will take a longer time to reach minima.

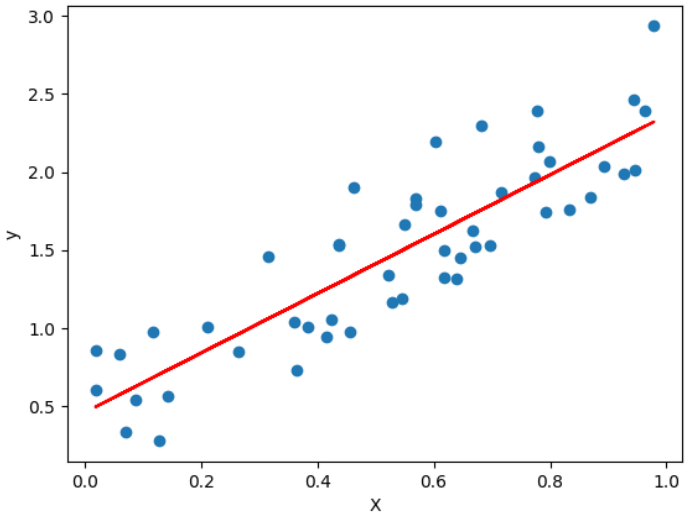
CODE:

```
# Importing Libraries
import numpy as np
import matplotlib.pyplot as plt
# Data for Regression
np.random.seed(0)
X = np.random.rand(50,1)
y = 2*X + np.random.rand(50,1)
# Visualizing Data
plt.scatter(X,y)
plt.xlabel("X")
plt.ylabel("y")
plt.show()
# Initializing Variables
m = X.shape[0]
epochs = 10000
theta = np.random.rand(2,1)
alpha = 0.01
print('Initial theta values', theta)
# Adding Bias
X = np.concatenate((np.ones(X.shape),X),axis=1)
# Cost Function
def computeCost(X,y,theta):
    m = X.shape[0]
    h_x = np.matmul(X,theta)
    J = (1/(2*m))*(sum(np.square(h_x-y)))
    return J
print('Initial Cost', computeCost(X,y,theta))
# Gradient Descent
X_transpose = np.transpose(X)
for i in range(epochs):
    h_x = np.matmul(X,theta)
    theta = theta - ((alpha/m) * np.matmul(X_transpose,h_x-y))
print('Theta from Gradient Descent', theta)
# Plotting the Regression Line
h_x = np.matmul(X,theta)
plt.scatter(X[:,1],y)
plt.plot(X[:,1], h_x,color='r')
plt.xlabel("X")
plt.ylabel("y")
plt.show()
```

OUTPUT:

```
Initial theta values [[0.67781654]
 [0.27000797]]
Initial Cost [0.35883128]
Theta from Gradient Descent [[0.46065518]
 [1.90169445]]
```





CONCLUSION: We learnt about supervised machine learning and implemented one of the popular algorithms - Linear Regression in Python. We learnt about cost functions and implemented Gradient Descent which is one of the methods of minimising cost for plotting the most accurate regression line.