

**Final Year B. Tech (EE)**

**Trimester: X**

**Subject: AIML**

**Name: Shreerang Mhatre**

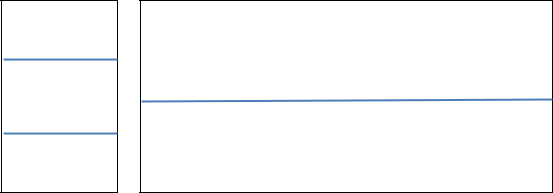
**Roll No: 52**

**Class: TY**

**Batch: A3**

**Experiment No: 05**

**Name of the Experiment**:Linear Regression Using Python



**Marks** **Teacher’s Signature with date**

**Performed on: 21/09/2023**

**Submitted on: 29/09/2023**



**Aim:** Implement Linear Regression Using Gradient Descent Algorithm.

**Prerequisite:** Knowledge of Supervised Learning method, MLP, Activation function.

**Objective:**

To create and study the Neural Network by varying parameters.

1. Define Input and Output Variable
2. Define and custom Neural Network
3. Configure the network
4. Train the network to find output

**Components and Equipment required:**

**Python software**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **AMP/WPU/AIML/2020** | | |  | **Expt. 5- 1** | |
|  |  |  |  |  |  |
|  |  |  | **WWW.MITWPU.COM** | | |





**Theory:**

**Linear Regression:**

Linear regression is an approach for modeling relationship between a scalar dependent variable y and one or more independent variables. When there is single independent variable it is called simple linear regression and when there are more than one independent variables, it is called multiple linear regression. Linear regression is a statistical procedure for predicting the value of a dependent variable from an independent variable when the relationship between the variables can be described with linear model given by y = m \* x + c.

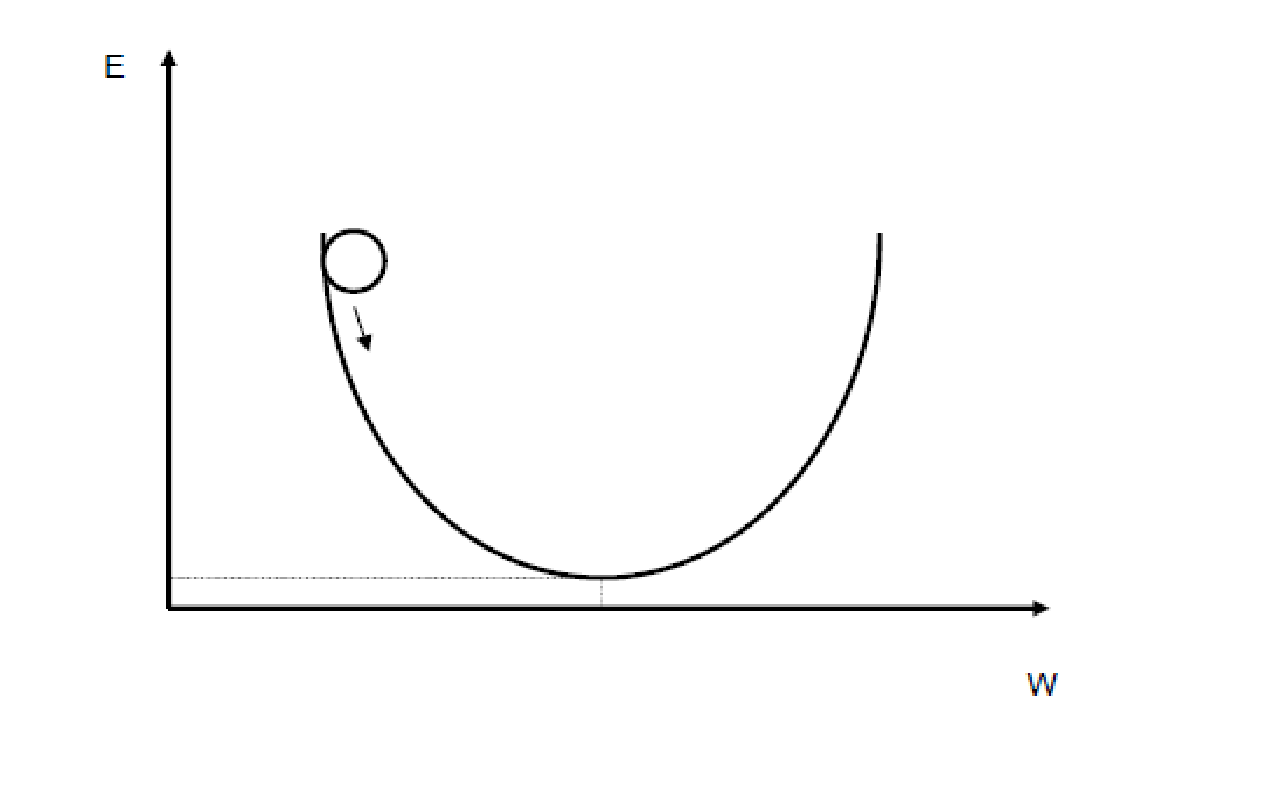
**Gradient Descent**

In this method goal is to obtain parameters such that sum of squared error between target output and actual output is minimized.

Gradient descent is an iterative method. Algorithm starts with some set of values for our model parameters (weights and biases) and improves them slowly. Change in weight for each iteration is proportional to derivative of cost function w.r.t. the current weight value.

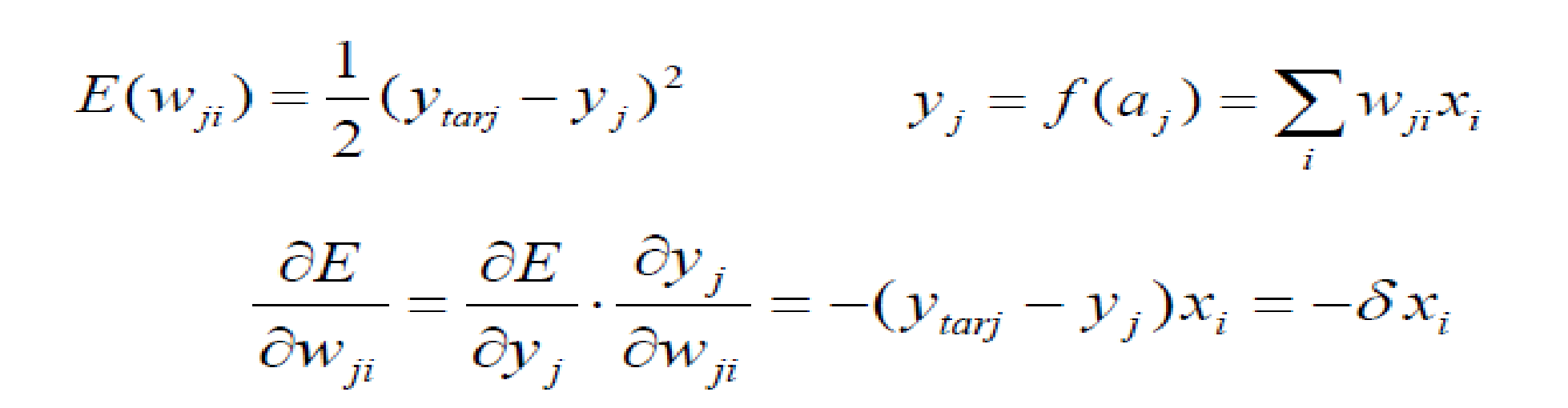
Thus algorithm moves in the direction of minima of cost function.

Gradient descentis an optimization algorithm that approaches a local minimum of a function by taking steps proportional to the negative of the gradient of the function as the current point.



**Figure 1: Error and global minima**

To run gradient descent on error function, we first need to compute its gradient. The gradient will act like a compass and always point us downhill. To compute it, we will need to differentiate our error function. Since our function is defined by two parameters (w1 and w2), we will need to compute a partial derivative for each. These derivatives work out to be:



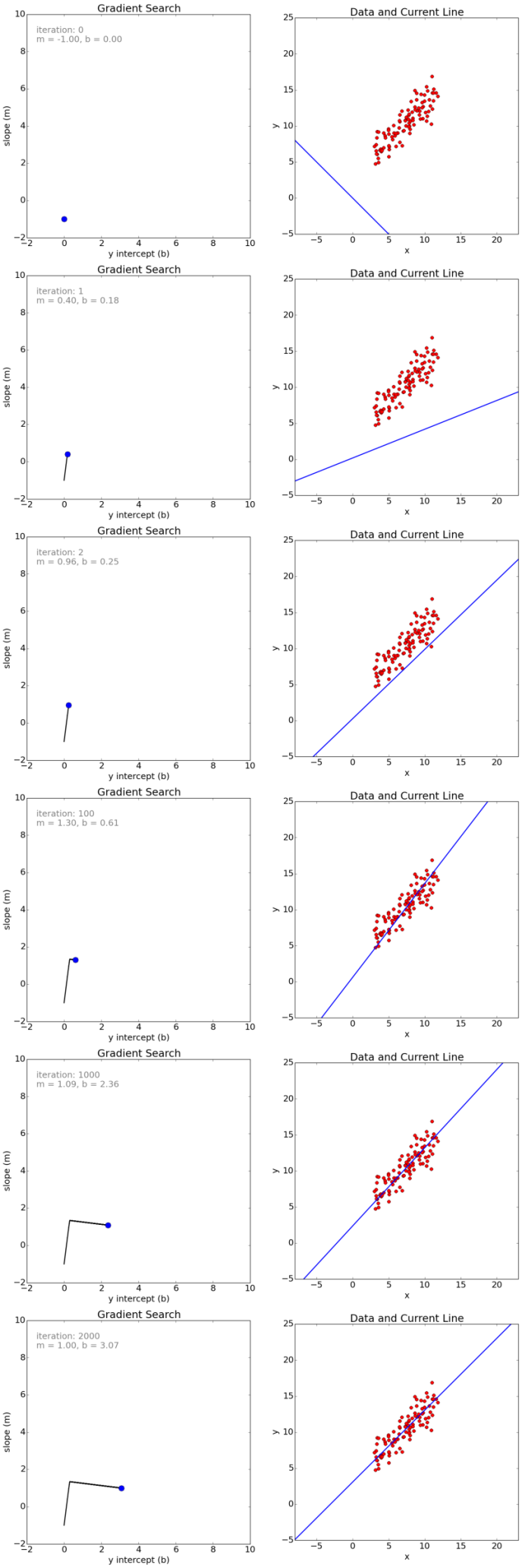
We now have all the tools needed to run gradient descent. We can initialize our search to start at any pair of w1 and w2 values (i.e., any line) and let the gradient descent algorithm march 

downhill on our error function towards the best line. Each iteration will update w1 and w2 to a line that yields slightly lower error than the previous iteration.

The learning Rate variable controls how large of a step we take downhill during each iteration. If we take too large of a step, we may step over the minimum. However, if we take small steps, it will require much iteration to arrive at the minimum.

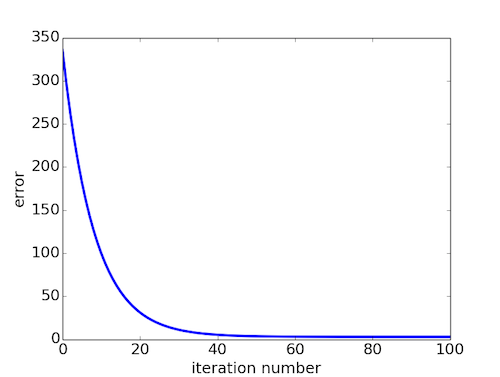
Below are some snapshots of gradient descent running for 2000 iterations for the example problem? We start out at point w1 = -1 w2 = 0. Each iteration w1 and w2 are updated to values

that yield slightly lower error than the previous iteration. The left plot displays the current location of the gradient descent search (blue dot) and the path taken to get there (black line). The right plot displays the corresponding line for the current search location. Eventually we ended up with a pretty accurate fit.

[](http://d1u2s20mo6at4b.cloudfront.net/wp-content/uploads/gradient_descent_search1.png)

**Figure 2: Different iterations of gradient descent**

We can also observe how the error changes as we move toward the minimum. A good way to ensure that gradient descent is working correctly is to make sure that the error decreases for each iteration. Below is a plot of error values for the first 100 iterations of the above gradient search.

[](http://d1u2s20mo6at4b.cloudfront.net/wp-content/uploads/gradient_descent_error_by_iteration.png)

**Figure 3: MSE Vs Epochs**

While the model in above example was a line, the concept of minimizing a cost function to tune parameters also applies to regression problems that use higher order polynomials and other problems found around the machine learning world.

Procedure

**Algorithm:**

**Step 0:**  Initialize weights w0,w1.(Set to small random values) and α (learning rate).

**Step 1:** While stopping condition is false, do steps 2-8.

**Step 2:** For each training pair with (xi,yi), do steps 3-6.

**Step 3:** Each input unit (Xi, i = 1…….n) receives input signal xi.

**Step 4:** Estimate E(w), where E(w)=1/(2M)\* ∑ ((w0+w1\*xi)-yi)2

**Step 5:** Estimate new values of w0(new), w1(new) using the following equations and store it in a temporary variable.

**Step 6:** Perform simultaneous update of w0(new)=Temp0

w1(new)=Temp1

**Step 7:** Update Mean square error, until the Error function E(w) is minimized 

**Step 8:** Stop.

**Python Code**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

data = pd.read\_csv('age1.csv')

data.head(5)

data.describe()

plt.scatter(data['AgeGroup'],data['TotalCases'])

plt.xlabel("Age Group of people")

plt.ylabel("Total Confirmed cases")

plt.title("Corona with age Prediction")

data\_n=data.values

m=data\_n[:,0].size

x=data\_n[:,0].reshape(m,1)

y=data\_n[:,2].reshape(m,1)

xTrain,xTest,yTrain,yTest=train\_test\_split(x,y,test\_size=0.1,random\_state=0)

yTrain

linearRegressor=LinearRegression()

linearRegressor.fit(xTrain,yTrain)

yprediction=linearRegressor.predict(xTest)

xTest,yprediction

import matplotlib.pyplot as plot

plot.scatter(xTest,yTest,color='red')

plot.plot(xTrain,linearRegressor.predict(xTrain))

plot.title('Age vs Corona Confirmed cases')

plot.xlabel('Age Group No')

plot.ylabel('Confirmed cases')

plot.show()

**Conclusions:**

**Post Lab Questions:**

1. 1. Explain the Gradient Descent algorithm.
2. Explain significance of learning rate in case of gradient descent algorithm?
3. What is global minima and local minima?

