UNIVERSITY OF MUMBAI

## DEPARTMENT OF COMPUTER SCIENCE



M.Sc. Computer Science – Semester I

Criminal Database Management System

PROJECT 2022-2023

Seat No.



UNIVERSITY OF MUMBAI

**DEPARTMENT OF COMPUTER SCIENCE**

# CERTIFICATE

This is to certify that the work entered in this Project was done in the University Department of Computer Science laboratory by

Mr./Ms. **SUYASH SURVE** Seat No. for the course of M.Sc. Computer Science - Semester I (CBCS) (Revised) during the academic year 2022- 2023 in a satisfactory manner.

**Subject In-charge Head of Department**

**External Examiner**

# Index

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr.  no. | Topic | Page No. | Date | Sign |
| 1 | INTRODUCTION | 1 |  |  |
| 2 | ALGORITHM | 4 |  |  |
| 3 | IMPLEMENTATION | 9 |  |  |
| 4 | SCREEN LAYOUT | 12 |  |  |
| 5 | CONCLUSION | 14 |  |  |
| 6 | REFERENCE | 15 |  |  |

#### ACKNOWLEDGEMENT

A project is always a result of the amalgamation of various ideas and support from countless people. I would like to acknowledge the contribution made by them. I am glad to represent my project on “**Criminal Database Management System**” This project has been a learning and challenging process for me.

I express my sincere thanks to **Prof.padmja** also express my thanks to Classmates and staff of the Computer Science Section/Department. I would like to thank my friend for her encouragement and guidance, which helped me in completing the project.

I immensely grateful to all those because without their inspiration, constant prompting and useful suggestions.

I sincerely appreciate the help provided by those in the careful preparation of the project.

Sincere thanks

from,

#### Mr. SUYASH SURVE

**DECLARATION**

I, “ **MR .SUYASH SURVE** ” hereby declare that I myself have

completed the project under the guidance of Prof.padmja, I on my own have done all the configurations and have implemented all the required settings.

--SUYASH SURVE

#### INTRODUCTION

Import the libraries

Before we start our example we will have to install the required libraries as shown below

### Installing pymongo

PyMongo is a Python distribution containing tools for working with MongoDB, and is the recommended way to work with MongoDB from Python.

pip install pymongo

### Installing tkinter

Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. It is a thin object-oriented layer on top of Tcl/Tk. Tkinter is not the only GuiProgramming toolkit for Python.

pip install tk

Understanding Gradient descent (GD) algorithm

Gradient descent (GD) is an iterative first-order optimisation algorithm used to find a local minimum/maximum of a given function. This method is commonly used in machine learning (ML) and deep learning (DL) to minimise a cost/loss function (e.g., in a linear regression). Due to its importance and ease of implementation, this algorithm is usually taught at the beginning of

almost all machine learning courses.

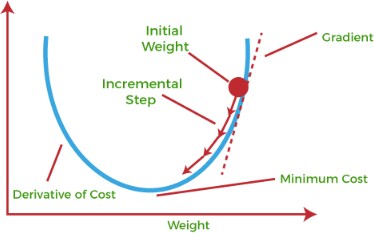
The best way to define the local minimum or local maximum of a function using gradient

descent is as follows:

If we move towards a negative gradient or away from the gradient of the function at the

current point, it will give the local minimum of that function.

Whenever we move towards a positive gradient or towards the gradient of the function at the current point, we will get the local maximum of that function. One This entire procedure is known as Gradient Ascent, which is also known as



steepest descent. The main objective of using a gradient descent algorithm

is to minimize the cost function using iteration. To achieve this goal, it performs two steps iteratively:

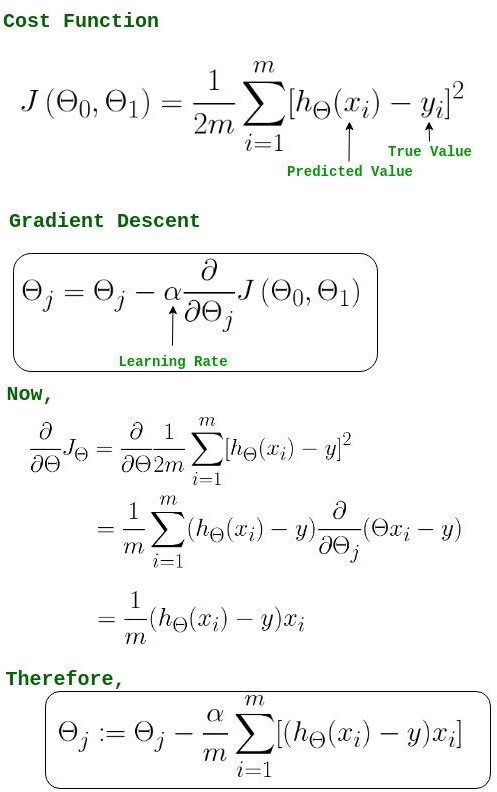
Calculates the first-order derivative of the function to compute the gradient or slope of that function.

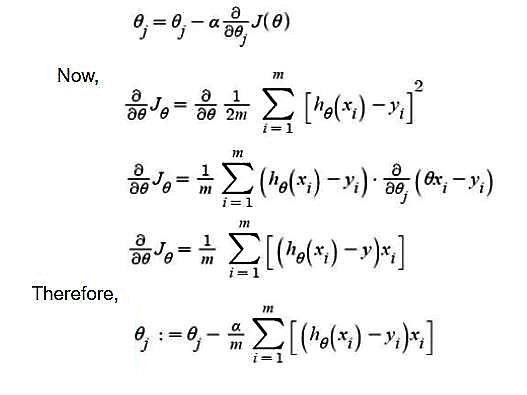
Move away from the direction of the gradient, which means slope increased

from the current point by alpha times, where Alpha is defined as Learning Rate.

It is a tuning parameter in the optimization process which helps to decide the length of the steps.

#### Gradient Descent Algorithm For Linear Regression





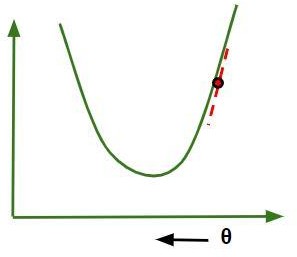
-> θj : Weights of the hypothesis.

-> 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.

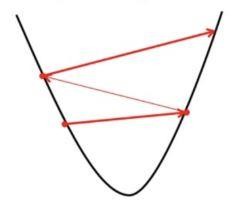
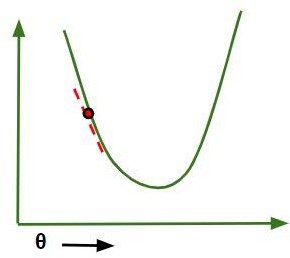
In the Gradient Descent algorithm, one can infer two points :

If slope is +ve : θj = θj – (+ve value). Hence 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.



For linear regression Cost, the Function graph is always convex shaped.

#### IMPLEMENTATION

Implementation of gradient descent in linear regression import numpy as np

import matplotlib.pyplot as plt

class Linear\_Regression:

def init (self, X, Y): self.X = X

self.Y = Y self.b = [0, 0]

def update\_coeffs(self, learning\_rate): Y\_pred = self.predict()

Y = self.Y m = len(Y)

self.b[0] = self.b[0] - (learning\_rate \* ((1/m) \*

np.sum(Y\_pred - Y)))

self.b[1] = self.b[1] - (learning\_rate \* ((1/m) \*

np.sum((Y\_pred - Y) \* self.X)))

def predict(self, X=[]): Y\_pred = np.array([]) if not X: X = self.X b = self.b

for x in X:

Y\_pred = np.append(Y\_pred, b[0] + (b[1] \* x))

return Y\_pred

def get\_current\_accuracy(self, Y\_pred): p, e = Y\_pred, self.Y

n = len(Y\_pred) return 1-sum(

[

abs(p[i]-e[i])/e[i] for i in range(n) if e[i] != 0]

)/n

#def predict(self, b, yi):

def compute\_cost(self, Y\_pred): m = len(self.Y)

J = (1 / 2\*m) \* (np.sum(Y\_pred - self.Y)\*\*2) return J

def plot\_best\_fit(self, Y\_pred, fig):

f = plt.figure(fig) plt.scatter(self.X, self.Y, color='b') plt.plot(self.X, Y\_pred, color='g') f.show()

def main():

X = np.array([i for i in range(11)]) Y = np.array([2\*i for i in range(11)])

regressor = Linear\_Regression(X, Y)

iterations = 0

steps = 100

learning\_rate = 0.01 costs = []

#original best-fit line Y\_pred = regressor.predict()

regressor.plot\_best\_fit(Y\_pred, 'Initial Best Fit Line')

while 1:

Y\_pred = regressor.predict()

cost = regressor.compute\_cost(Y\_pred) costs.append(cost) regressor.update\_coeffs(learning\_rate)

iterations += 1

if iterations % steps == 0: print(iterations, "epochs elapsed") print("Current accuracy is :",

regressor.get\_current\_accuracy(Y\_pred))

stop = input("Do you want to stop (y/\*)??") if stop == "y":

break

#final best-fit line

regressor.plot\_best\_fit(Y\_pred, 'Final Best Fit Line')

#plot to verify cost function decreases h = plt.figure('Verification')

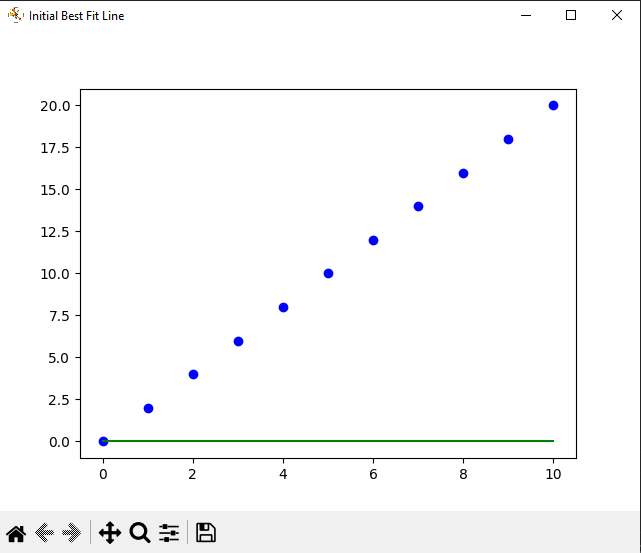
plt.plot(range(iterations), costs, color='b') h.show()

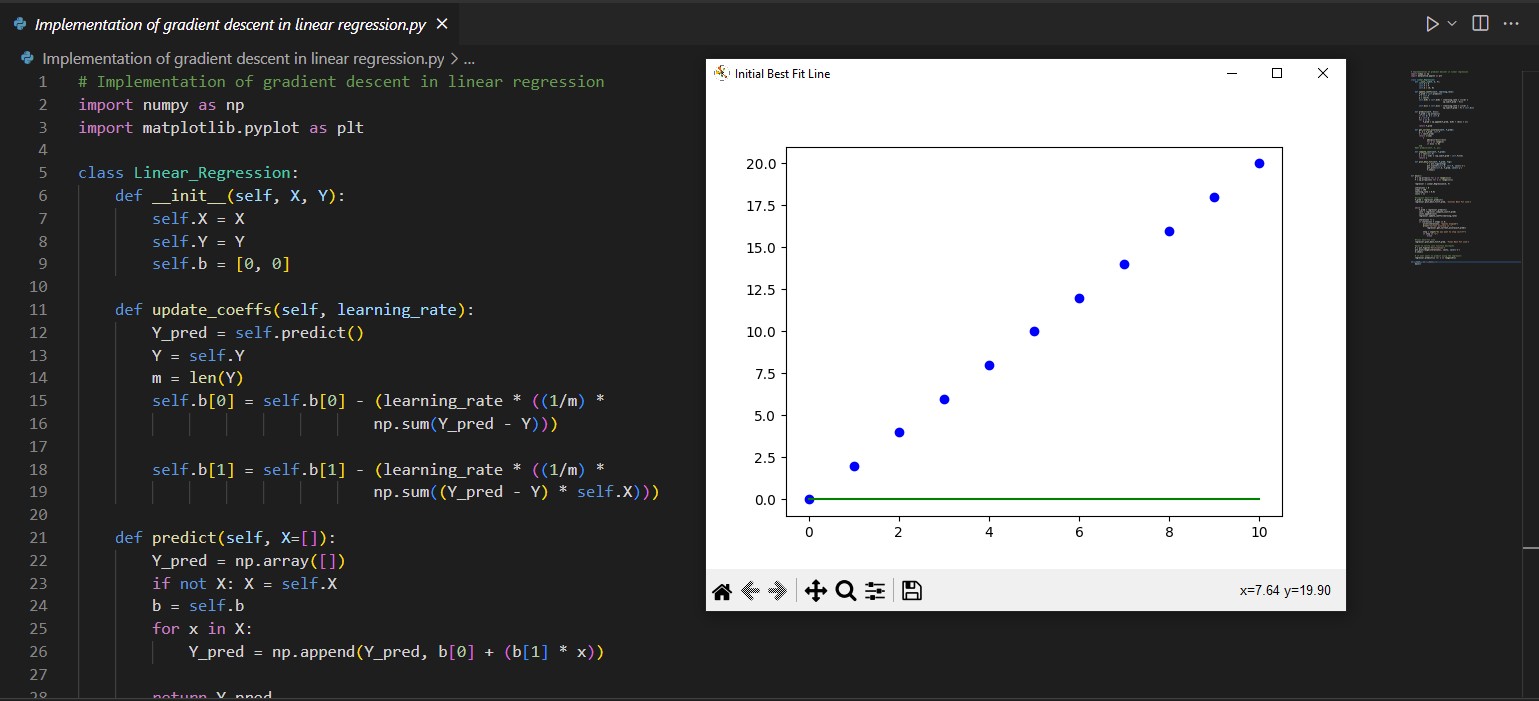
# if user wants to predict using the regressor: regressor.predict([i for i in range(10)])

if name == ' main ': main()

**SCREEN LAYOUT**

**Output Image:-**





1

#### Conclusion

In this mini project, we have discussed the gradient descent algorithm works and implement it from scratch in python. First we look at what linear regression is, then we define the loss function. We learn how the gradient descent algorithm works and finally we will implement it on a given data set and make predictions.

Gradient Descent can be used to optimize parameters for every algorithm whose loss function can be formulated and has at least one minimum. Also, we cleared up our understanding of the infamous gradient descent loss function equation.

We learned how to use a visual method to estimate the number of iterations required. Finally, we got to know the actual workings behind the scenes of the famous gradient descent algorithm.

M.Sc. Computer Science – Semester I Algorithm for Optimization Project-2022-2023

15

#### References

1. <https://www.geeksforgeeks.org/gradient-descent-in-linear-regression/>
2. [https://towardsdatascience.com/implementing-gradient-descent-in-python-from-scratch-](https://towardsdatascience.com/implementing-gradient-descent-in-python-from-scratch-760a8556c31f) [760a8556c31f](https://towardsdatascience.com/implementing-gradient-descent-in-python-from-scratch-760a8556c31f)
3. <https://github.com/kanuarj/GettingStartedWithML/tree/master/10_gradient_descent>
4. <https://www.youtube.com/watch?v=KtBxft9wDC0>
5. <https://www.javatpoint.com/gradient-descent-in-machine-learning>

M.Sc. Computer Science – Semester I Algorithm for Optimization Project-2022-2023