Data Science Bootcamp

Hyperiondev

Gradient Descent

WELCOME

Your Lecturer for This Session

Alfred Ndlovu

Lecture – Housekeeping

- ❑ The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- ❑ No question is daft or silly - **ask them!**
- ❑ There are Q/A sessions midway and at the end of the session, should you wish to ask any follow-up questions.
- ❑ You can also submit questions here: hyperiondev.com/sbc4-ds-questions
- ❑ For all non-academic questions, please submit a query: hyperiondev.com/support
- ❑ Report a safeguarding incident: hyperiondev.com/safeguardreporting
- ❑ We would love your feedback on lectures: https://hyperionde.wufoo.com/forms/zsgv4m40ui4i0g/

Lecture – Code Repo

Go to: github.com/HyperionDevBootcamps

Then click on the “**C4_DS_lecture_examples**” repository, do view or download the code.

Objectives

1. Understand gradient descent
2. How it works in linear regression

Gradient Descent?

- **Gradient descent is an optimization algorithm used in machine learning to find the minimum (or maximum) of a function.**

- **It is used to train models by minimizing the error between predicted and actual values.**
  - Predicted - what the model returns
  - Actual Values - what was meant to be returned

So what happens during training?

- The model takes the input data "x_train" and makes predictions.

- The predictions are compared with the corresponding true values in "y_train" to calculate the error or loss.

- The model adjusts its parameters using an optimization algorithm such as gradient descent to minimize the loss.

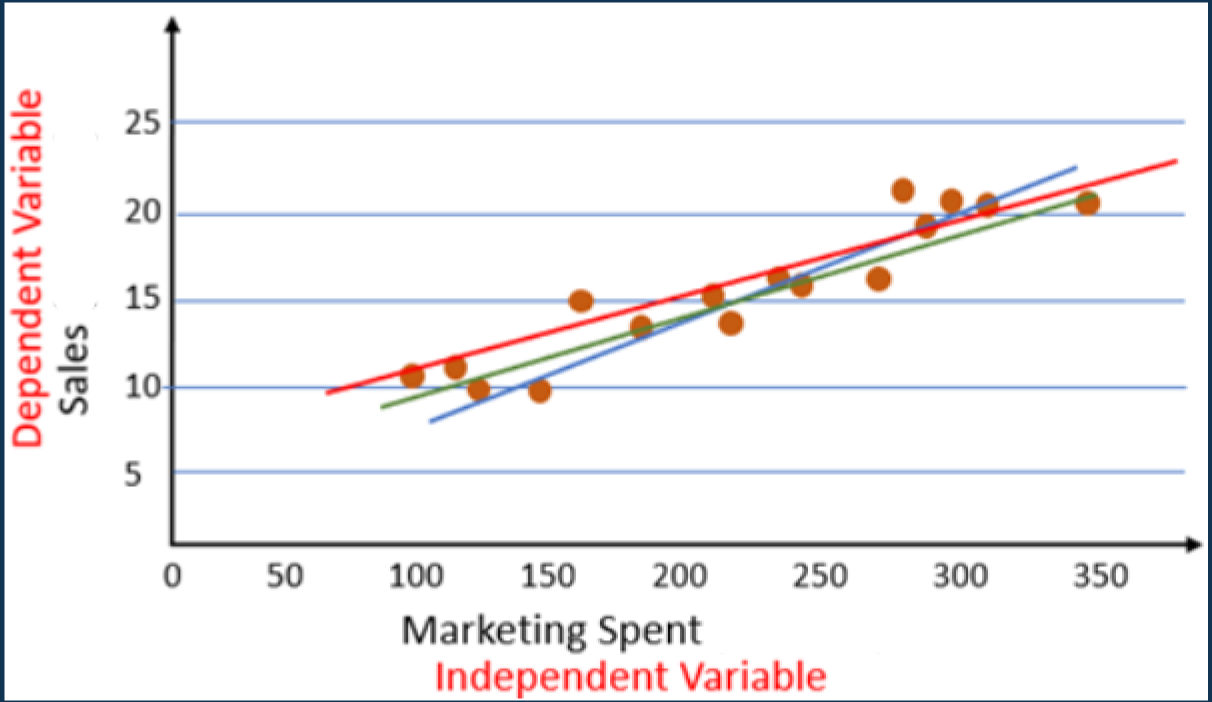
Example

- **Linear Regression** - we assume that there's a linear relationship between the independent variables and the dependent variable

- The goal is to find the best-fit line that minimizes the difference between the predicted values and the actual values in the dataset(cost function)

- Example : Marketing vs Sales

Example



How do we know which of these lines is the best fit line?

COST FUNCTION ?

- Its a function used in machine learning and optimization algorithms to measure how well a model or algorithm performs on a given dataset

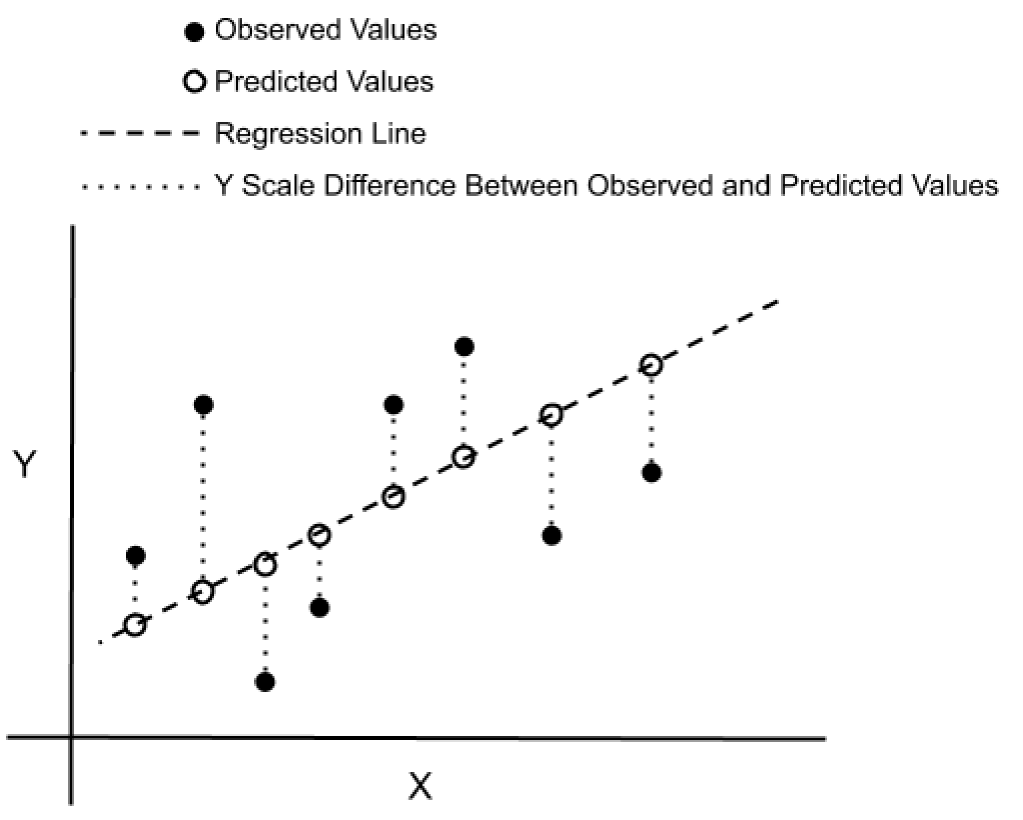
- It quantifies the discrepancy between the predicted outputs of the model and the actual observed values.

Linear Regression COST FUNCTION

Mean Squared Error

- measure the average squared difference between the predicted and actual values
- Quantifies the average "error" or "deviation" of the model's predictions from the true values.

Mean Squared Error



T'S ALL ABOUT MINIMIZING THE COST FUNCTION

Linear Regression - minimize the cost function **MSE** as much as possible in order to find the best fit line

EQUATION OF A STRAIGHT LINE

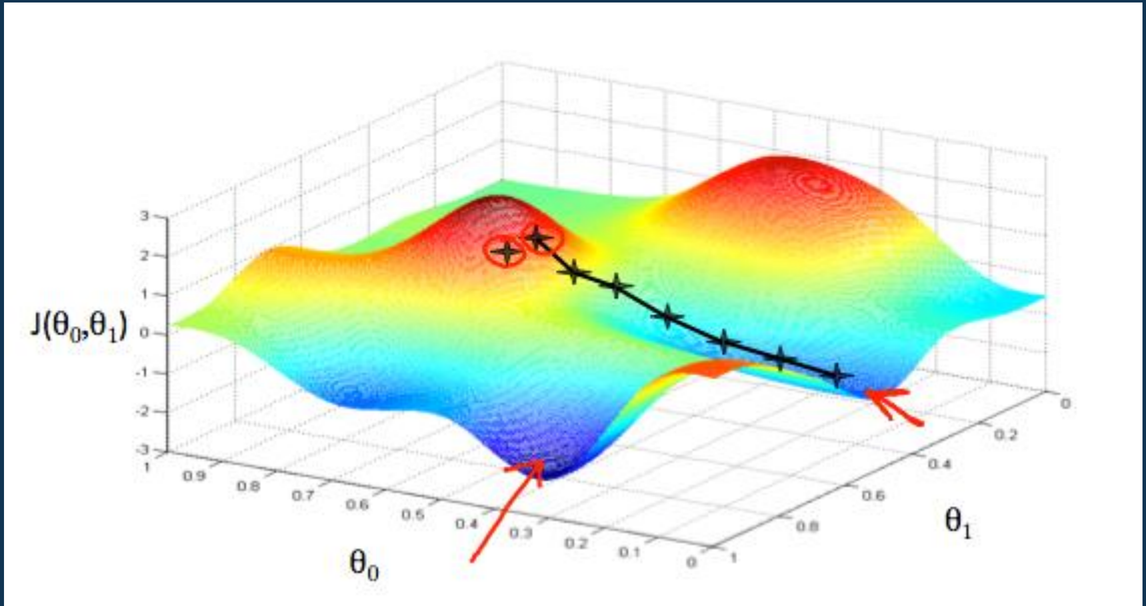
$$y = mx + c$$

m is the gradient of the line (how steep the line is)

c is the y -intercept (the point in which the line crosses the y -axis).

X represents the input or independent variable

DEEP DIVE



- Plot ‘m’ and ‘c’ against MEAN SQUARED ERROR
- For some combination of m and c, we will get the least Error (MSE)

HOW DOES GRADIENT DESCENT WORK ?

1. Let **m** = 0 and **c** = 0

1. Calculate the partial derivative of the Cost function with respect to **m** and **c**

1. Update the current values of **m** and **c**

1. Repeat this process until our **Cost function** is very small (ideally 0).

Derivative with respect to m and c?

- A derivative represents the slope or steepness of a function at any point on its graph.

- It tells us how quickly the function is increasing or decreasing.

- To find the derivative with respect to m (slope), we differentiate the MSE equation with respect to m
  d(MSE)/dm
- To find the derivative with respect to c (intercept), we differentiate the MSE equation with respect to c
  d(MSE)/dc

UPDATING M AND C !!

Learning Rate :

- It controls how quickly or slowly the model learns from the data.
- If the learning rate is too large, let's say 0.1, the parameter updates will be significant, you risk diverging
- If the learning rate is too small, let's say 0.0001, the parameter updates will be tiny, and the model will take longer to converge to the optimal values

- range of 0.1 to 0.001 are commonly used as a starting point. This range provides a reasonable balance between convergence speed and stability.

STILL UPDATING !

- $m = m - \alpha * d(MSE)/dm$
- $c = c - \alpha * d(MSE)/dc$
- α is our learning rate
- After updating, we just repeat till the **COST function** has been minimized

CODING TIME !!!!

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic explained, should you have any

Hyperiondev

Thank you for joining us