supervised Learning (Regression & classification)

week-02

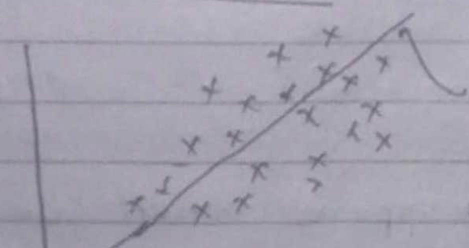
Linear Regression (simple + multiple)

- simple
  - 1 Input column and 1 output column
- multiple
  - More than 1 input column
- Polynomial

LR is a supervised learning algo. used to model the relationship between a dependent variable (target) and one or more independent variable (predictors). It assumes a linear relationship b/w the variables and aims to find a best fit line that minimize the error b/w predicted and actual values.

$$\boxed{y = mx + b}$$

How to find m and b



$y = mx + b$

Two method to find m and b

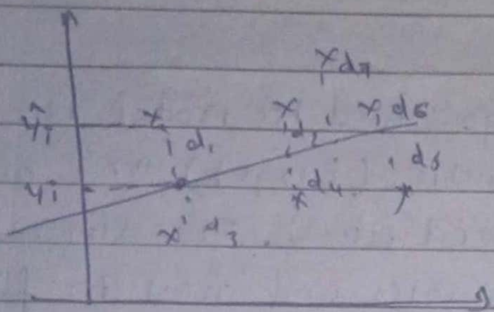
(1) closed form solution → OLS

(2) non-closed form → Gradient Descent

Through OLS

$$b = \bar{y} - m\bar{x} \qquad m = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$\bar{x}$, $\bar{y}$ → Mean



$$\text{Error} = d_1 + d_2 + d_3 \cdots + d_n$$

$$E = d_1^2 + d_2^2 + \cdots + d_n^2$$

$$\boxed{E = \sum_{i=1}^{n} d_i^2}$$ ← Error function

• Represent with J

$$\boxed{E = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

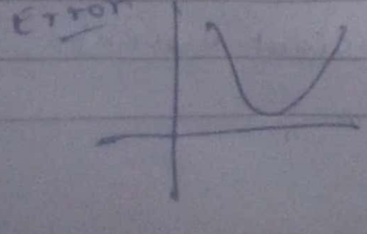
$$E = \sum_{i=1}^{n} (y_i - m x_i - b)^2$$

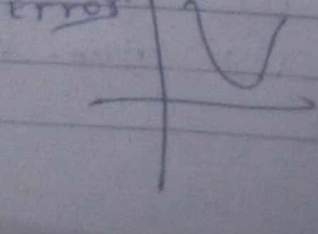
let's see what happens when we change m and b

suppose

$b = 0$ $\qquad$ $m = 1$

$$E(m) = \sum_{i=1}^{n} (y_i - m x_i)^2 \qquad E(b) = \sum_{i=1}^{n} (y_i - x_i - b)^2$$





$$\frac{\partial E}{\partial b} = \frac{\partial}{\partial b} \sum_{i=1}^{n} (y_i - mx_i - b)^2 = 0$$

$$\Rightarrow \sum \frac{\partial}{\partial b} (y_i - mx_i - b)^2 = 0$$

$$\Rightarrow \sum -2 (y_i - mx_i - b) = 0$$

$$\Rightarrow \sum y_i - \sum m x_i - \sum b = 0$$

$$\frac{\sum y_i}{n} - \frac{\sum m x_i}{n} - \frac{\sum b}{n} = 0$$

$$\bar{y} - m\bar{x} - \frac{\cancel{n} b}{\cancel{n}} = 0$$

$$\bar{y} - m\bar{x} - b = 0$$

$$\boxed{b = \bar{y} - m\bar{x}}$$

$$E = \sum (y_i - mx_i - \bar{y} + m\bar{x})^2$$

$$\frac{\partial E}{\partial m} = \sum \frac{\partial}{\partial m} (y_i - mx_i - \bar{y} + m\bar{x})^2 = 0$$

$$\Rightarrow \sum 2 (y_i - mx_i - \bar{y} + m\bar{x})(-x_i + \bar{x}) = 0$$

$$\Rightarrow \sum -2 (y_i - mx_i - \bar{y} + m\bar{x})(x_i - \bar{x}) = 0$$

$$\Rightarrow \sum (y_i - mx_i - \bar{y} + m\bar{x})(x_i - \bar{x}) = 0$$

$$\Rightarrow \sum [(y_i - \bar{y}) - m(x_i - \bar{x})] (x_i - \bar{x})] = 0$$

$$\Rightarrow \sum [(y_i - \bar{y})(x_i - \bar{x}) - m(x_i - \bar{x})^2] = 0$$

$$\Rightarrow \sum (y_i - \bar{y})(x_i - \bar{x}) = m \sum (x_i - \bar{x})^2$$

$$m = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Multiple Linear Regression

More than one column in input

$x_1 | x_2 | x_3 | y$

- 3D → plane
  - $y = mx_1 + nx_2 + b$
  - $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$
- 4D → hyperplane
  - $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$
- nd → -11-
  - $y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i$

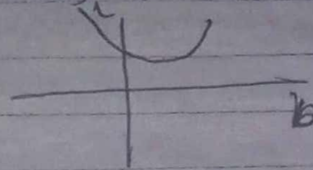
Gradient Descent

Gradient descent is a first order iterative optimization algorithm for finding a local minimum of a differentiable fxn.

Types
- Batch GD
- SGD — SGD Regressor
- MBGD

Loss fxn $J(\theta)$

opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$ with respect to the parameters



loss(w,b)

$\eta \frac{\partial L}{\partial w}$

Learning rate η

determine the size of the steps we take to reach a (local) minimum.

Back propogation Algo

gradient descent

```
epochs = 5
for i in range(epochs):
    for j in range(x.shape[0]):
        -> select 1 row (random)
        -> predict (using forward prop)
        -> calculate loss (using loss function -> mse)
        -> update weights and bias using GD
```

$$w_n = w_o - \eta \frac{\partial L}{\partial w}$$

-> calculate avg loss for the epoch

* $\left\{ \frac{\partial L}{\partial w} \right\}$ derivative How much data we use to compute 3 types

accuracy -> time — tradeoff

Batch GD (vanilla GD)

↳ we take the entire dataset and then we make update

Ex:

50 points ⟶ predict ⟶ current weigh

↑ dot product

y_hat = np.dot(x, w) + b

└ 50 predict

y = 50 actual outcome

- using this y and y_hat we calculate Loss $\sum_{i=1}^{50}$
- then basis of that loss we update single time weigh and bias

Ex:

epoch = 10

```
for i in range(10):
    y_hat = np.dot(x,w)+b
```

total 10 times repeat

50 values

y = 50

y_hat, y ⟶ loss

w, b update $w_n = w_0 - \eta \frac{\partial L}{\partial w}$

⟶ Loss

→ weight update frequency high

Stochastic GD

designed to handle large datasets efficiently by introducing randomness in the gradient computation

epoch → 10 (50 rows)

for i in range(10): → shuffle
    for i in range (x.shape[0])
    ↳ ~~shuffle~~
    ↳ ~~1 point~~
        ↳ ~~shuf~~ random point
        ↳ y-hat → forward
        ↳ loss
        ↳ w, b update → $w_n = w_0 - \eta \frac{\partial L}{\partial w}$

avg loss print → for the epoch

☆ 50 rows then 50 times weight update. (SGD)
no. of epoch times update (BGD)

Difference between both

	BGD	SGD
update if 10 epochs	10 time	10 (n) → no. of rows
which is the faster to converge (given same # epochs)		SGD

Mini Batch GD

combines the advantages of both SGD and BGD
by dividing training dataset into smaller
called mini-batches.

for i in epochs →
  for j in num of batch
    1 batch
      ↳ y-pred (vector)
      ↳ loss
      ↳ update

update of epoch speed: fast (bgd) > mbgd > sgd slow

convergence: slow bgd < mbgd < sgd fast

→ why batch size is provided in multiple of (2)?

actually RAM ka effective use krne ke liye
diya jata hain } design to handle binary value

→ what if batch_size doesn't divide # rows properly

e.g. # of rows n = 400

batch-size = 150

$\# \text{ of batch} = \frac{400}{150} = 2.66$

150, 150, left 100

Batch 1, b2, b3 ← left include in b_3

Regression Metrics

(1) Mean Absolute error

used to evaluate the accuracy of prediction

Measures the avg absolute diff. b/w the actual values and the predicted values.

$$MAE = \frac{1}{n} * \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

predicted

actual

total no. of observation.

Advantage ① same unit

② Robust to outliers

Disad. Modulus fxn graph is not diff. at zero.

(2) Mean squared error

avg squared diff. b/w predicted values and the actual values in a dataset.

A lower MSE indicates better model accuracy, higher suggests poor performance.

$$MSE = \frac{1}{n} * \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Advantage ① use as a loss fxn becoz differentiable

Dis ①. Not Robust to outliers.

RMSE

Root of MSE

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{n}}$$

Benefit

- RMSE output comes in same unit

Disad - Not Robust to outliers.

R2 score

Goodness of fit

value of R^2 ranges b/w 0 and 1,

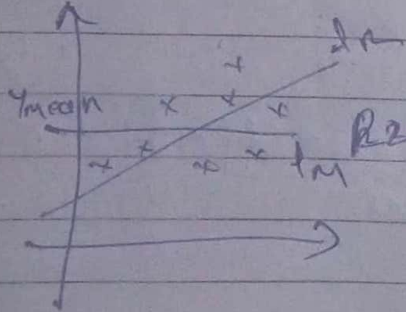
→ $R^2=1$ indicates a perfect fit, meaning the model explains all the variability in the dependent variable

→ $R^2=0$ means the model does not explain any variability and fails to capture the relationship b/w the variables.

$$R^2 = 1 - (SSres / SStot)$$

$\downarrow$ residual sum of squares

total sum of squares



$$R2 = SSR / SStot$$

sum of squares of Regression

$$R_2 = 1 - \frac{\left[\sum_{i=1}^{n}(y_i-\hat{y}_i)^2\right]_{Reg}}{\left[\sum_{i=1}^{n}(y_i-\hat{y}_i)^2\right]_{M}}$$

Adjusted R_2 score

To address the limitations of R^2, Adjusted R^2 is used.

if we add more input R_2 score ↑

$$R_2 adj = 1 - \left[\frac{(1 - R_2)(n-1)}{(n-1-K)}\right]$$

n - no. of rows

K - indepent value.