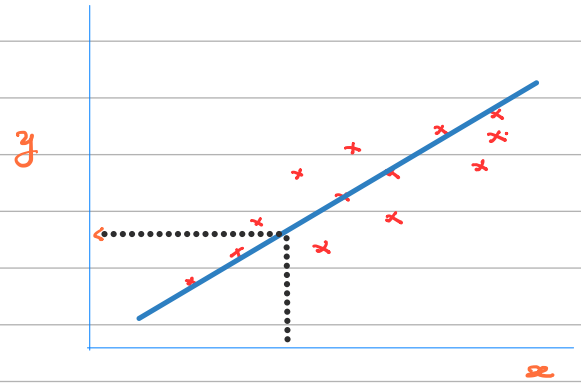
Simple Linear Regression

- Supervised Learning

- predict continuous numerical value
- finds best fitting straight line relation btw input and output

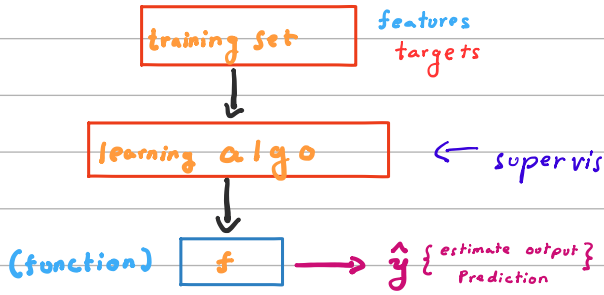
- used for a simple way to understand relationships and make prediction



x = "input" value
y = "output" value

$(x^{(i)}, y^{(i)})$ = i^{th} training example.

$(x^{(i)}, y^{(i)}) = (812, 120)$



← supervised Learning Algorithm

$f_{w,b}(x) = wx + b$

$f(x) = wx + b$

input feature

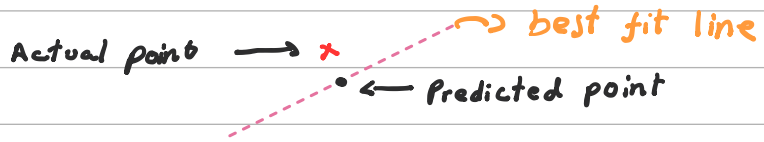
$f(x) = wx + b$ ← bias or intercept

weight or coefficient

Inputs represents the features
Output represents the target

The linear regression algo learns from the data to generate best fit line

Residual Error: Diffrence btw actual point and predicted point

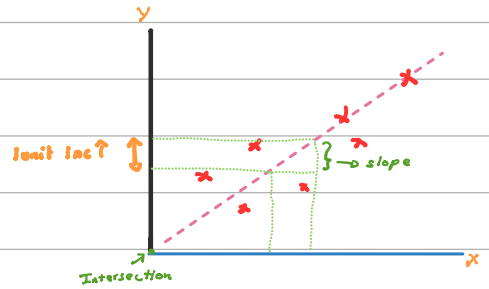


Aim: To find best fit line with minimal error

$$\hat{y} = mx + c$$

m = slope or coiffficient

c = Interception



Interception When the value of x is zero where is the best fit line intersecting y

$$h_\theta(x) = \theta_0 + \theta_1 x \longrightarrow \text{coefficient}$$

θ_0 ↓ intecept

Cost function formula

$$\sum_{i=1}^{m} \frac{1}{2m} \left(h_\theta(x)^{(i)} - y^{(i)} \right)^2$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x)^{(i)} - y^{(i)} \right)^2 \downarrow\downarrow\downarrow$$

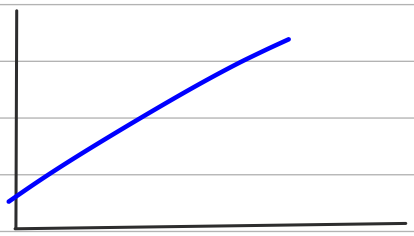
↳ squared Error Function

model :

$$f_{w,b}(x) = wx + b$$

parameters :

w, b



cost function :

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2$$

$f_{w,b}(x^{(i)})$: Predicted Value

$y^{(i)}$: actual value

goal :

minimize $J(w,b)$
w,b

Types of Linear Regression

LR

- Simple LR
- Multiple LR
- Polynomial LR

x_1	x_2	x_3	y_1
CGPA	gen	grade	lpa

$y = mx + b$

$y = m x_1 + n x_2 + b$

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

$y = \beta_0 + \beta_1 x_1 \dots + \beta_n x_n$

$$\boxed{y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i}$$

Linear Regression

- OSI — $\beta = (x^T x)^{-1} x^T y$
- Gradient decent

Linear regression can be found in two diff ways

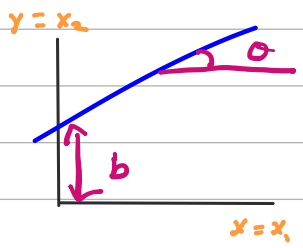
We use gradient decent most of the time because calculating inverse computationally cost alot. And the time complexity of a inverse $O(n^3)$ is more time consuming when large sets of column data present

Hyperplane in N Dimentions

The line formed in 3D, 4D or ND is known as hyperplane

Equation of Hyperplane

2D

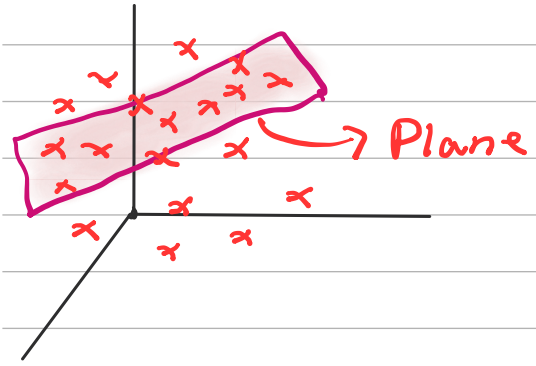


$Y = mx + b$

$ax + by + c = 0$

$a x_1 + b x_2 + c = 0$

$w_1 x_1 + w_2 x_2 + w_0 = 0$



Equation of Plane

$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + w_0 = 0$

multiple Linear Regression (mathematical formulation)

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots \beta_n x_n$$

$$\hat{Y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \\ \hat{y}_{100} \end{bmatrix} = \begin{bmatrix} \beta_0 & \beta_1 x_{11} & \beta_2 x_{12} & \beta_3 x_{13} \\ \beta_0 & \beta_1 x_{21} & \beta_2 x_{22} & \beta_3 x_{23} \\ & \vdots & & \\ \beta_0 & \beta_1 x_{1001} & \beta_2 x_{1002} & \beta_3 x_{1003} \end{bmatrix} \quad (100, 3)$$

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} \beta_0 & \beta_1 x_{11} & \beta_2 x_{12} & \beta_3 x_{13} \dots & \beta_m x_{1m} \\ \beta_0 & \beta_1 x_{21} & \beta_2 x_{22} & \beta_3 x_{23} \dots & \beta_m x_{2m} \\ \beta_0 & \beta_1 x_{n1} & \beta_2 x_{n2} & \beta_3 x_{n3} \dots & \beta x_{nm} \end{bmatrix}$$

$$\hat{y} = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} \dots & x_{1m} \\ 1 & x_{21} & x_{22} & x_{23} \dots & x_{2m} \\ \vdots & & & & \\ 1 & x_{n1} & x_{n2} & x_{n3} \dots & x_{nm} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}$$

$$\boxed{\hat{y} = X\beta} \quad \text{— (1)}$$

$$e = y - \hat{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \hat{y} \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix}$$

Regression Metria

To find the effeciency of Regression algorithm.

1). MAE - mean absolute error
2. MSE - mean Score Error
3. RMSE - Root mean score error

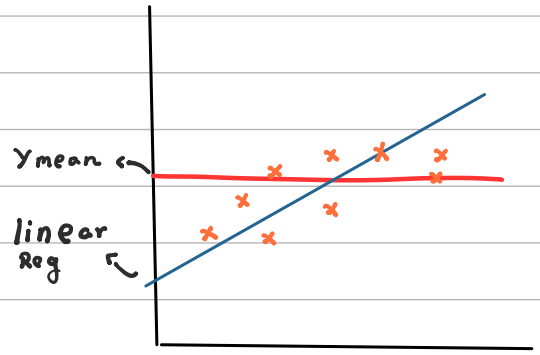
← loss function / error function

4. R2 Score - Coefficient of determination
5. Adjusted R2 score

↳ goodness of fit (literature)

} ← checking the accuracy

R 2 Score



$$R2 = 1 - \frac{SS_R}{SS_m}$$ - Sum of Squared Error in Regression / Sum of Squared Error in mean line

$$R2 = 1 - \frac{\left[\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right]_{Reg}}{\left[\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right]_{m}}$$

(for linear Reg)

(for mean line)

$R_2 = 0$ ← If the score is moving towards $R_2 = 0$ then its worst Reg line

$R_2 = 1$ ← If the score is moving towards $R_2 = 1$ then its perfect lin line

$R_2 = -(neg)$ ← It means that the lin line performs worst than the mean line

$R2 = 0.80$ ↷ 80% of Variance

Adjusted R2 score

In case of irrevelent column is added the R_2 score will either increase or remain constant, to avoid this we use Adj R_2 score

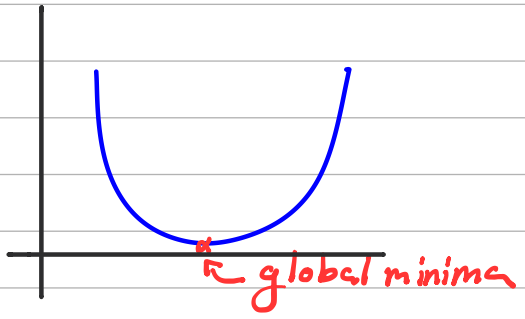
$$R^2_{adj} = 1 - \left[\frac{(1-R_2)(n-1)}{(n-1-k)}\right]$$

$n \rightarrow$ no of Rows
$k \rightarrow$ Indipendent columns
$R_2 \rightarrow R_2$ score

when relatable column is added then R2 increases

Mean Square Error

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



① mean Squared Error :

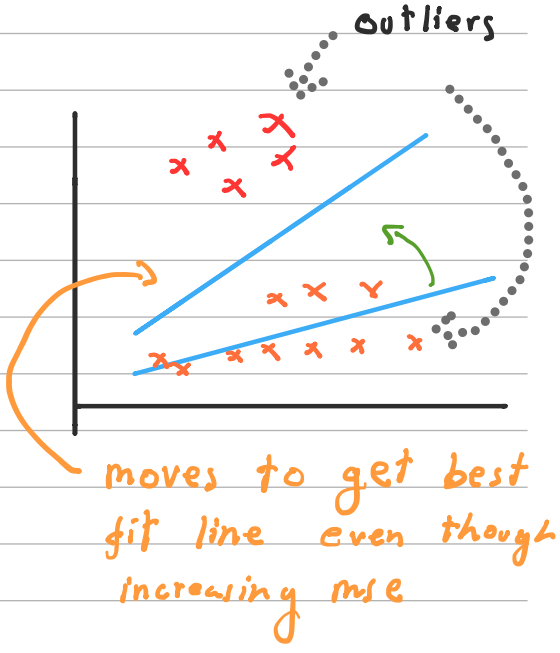
Advantage :

① Diffrenciable. (To find global minima)
② It has one global minima and local minima

Disadvantage :

① Non Robust to outliers

② It changes its unit



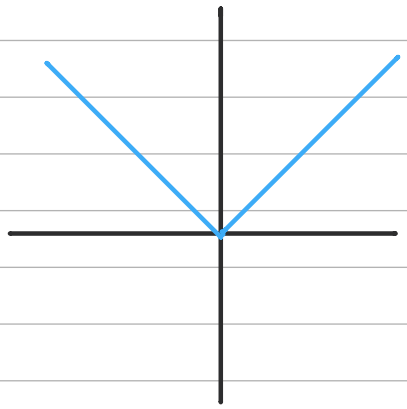
Mean absolote Error (MAE)

cost function MAE = $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$

Advantage

1) It is robust to outliers
(The Best fit line if changed due to outliers might be minimal because its not squared)

2) It will be in same unit



no gradient decent is formed in MAE

Disadvantage

① Convergence usually takes more time

② optimisation is a complex process

RMSE

RMSE = $\sqrt{MSE}$

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

Advantage:

1) It will have same units
2) Differentiable

Note: The results by MAE & RMSE have diffrent value and are not same.

Disadvantage

Not Robust to outliers

Polynomial Regression

$y = \beta_0 + \beta_1 x$ → Simple Linear Regression

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \dots \beta_n x_n$

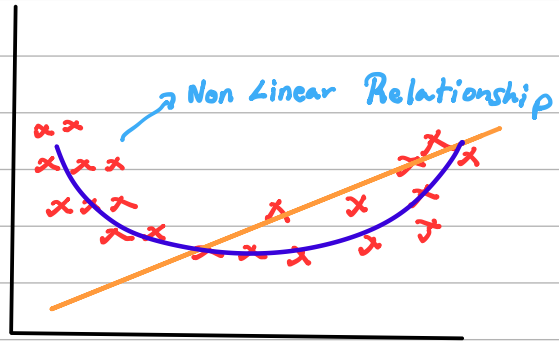
→ multiple Linear Regression

degree = 2 → hyperparameter.

Simple Polynomial Regression : $y = \beta_0 + \beta_1 x + \beta_2 x^2$

(To extract non linear relation)

multiple Polynomial Regression : $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x_2 + \beta_4 x_2^2$



When we cannot use Linear Regression (If causes to much error) we use polynomial Regression.

Polynomial Regression → Simple Polynomial Regression

Polynomial Regression → Multiple Polynomial Regression

Simple Polynomial Regression

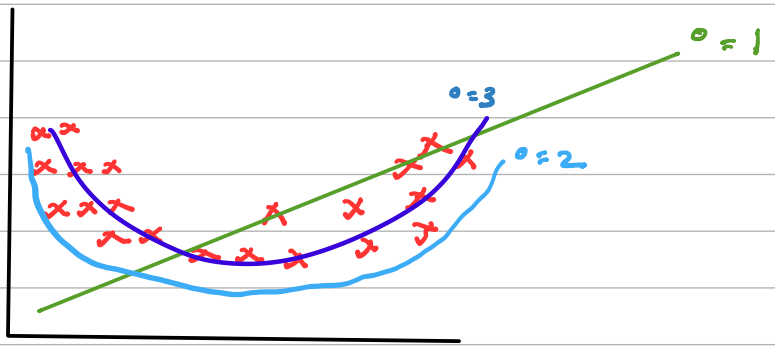
Polynomial degree = 0 $h\theta(x) = \theta_0 * 1 = \theta_0 x_1^0$ ⇒ constant

Polynomial degree = 1 $h\theta(x) = \theta_0 x_1^0 + \theta_1 x_1^1$ ⇒ Simple Linear Reg

polynomial degree = 2 $h_\theta(x) = \theta_0 x_1^0 + \theta_1 x_1^1 + \theta_2 x_1^2$

$\vdots$

polynomial degree = n $h_\theta(x) = \theta_0 x_1^0 + \theta_1 x_1^1 + \theta_2 x_1^2 \dots \theta_n x_1^n$



multiple Polynomial Regression

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$ { multiple Linear Regression }

Polynomial degree = 2

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_1^2 + \theta_5 x_2^2 + \theta_6 x_3^2$

Regularized Linear Regression Techniques.

① Ridge Regression → Reduces over fitting

Generalized Model

Low Bias
Low Varience

Aim to make generalized model

To prevent overfitting in Linear Regression we use Ridge regression

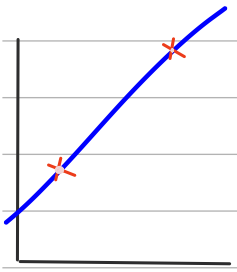
Linear Regression
cost function → Residual Error

$$= \frac{1}{m} \sum_{i=1}^{m} \left(h\theta(x)^{(i)} - y^{(i)} \right)^2$$

↓↓↓ (should be low)

$$\left(h\theta(x)^{(i)} - y^{(i)} \right)^2$$

$= 0$ ← If its zero then its overfitting



Ridge Regression (L_2 Regularization)

$$= \left(h\theta(x)^{(i)} - y^{(i)} \right) + \lambda (\text{slope})^2$$

$\lambda = 1$ $= 0 + 1(2)^2$

$= 4$

λ ↑↑ slope ↓↓

Even though line passes through the points dosent be 0 and dosent cause overfitting

∴ Overfitting is removed

② Lasso Regression (L_1 Regularization / norm) → feature Selection

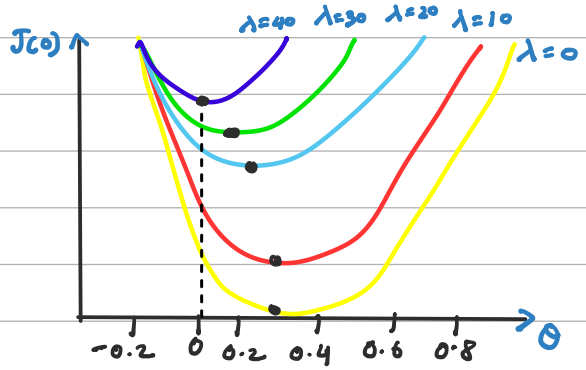
Cost fn : $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{n} |\text{slope}|$

$$(h_\theta(x)^{(i)} - y^{(i)})^2 + \lambda\,|\text{slope}|$$

1} Prevents overfitting

2} Feature selection {feature Removal}

It uses feature selection process in which if the coefficent is less by using feature selection we eleminate the value by adjusting λ (hyperparameter) value.



Note

$\lambda \uparrow\uparrow$ $\theta \downarrow\downarrow$

Till it becomes 0

③ Elastic Net Regression

- Reducing Overfitting → Ridge
- Feature Selection → lasso

$$\text{Cost function} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda_1 \sum_{i=1}^{n}(\text{slope})^2 + \lambda_2 \sum_{i=1}^{n}|\text{slope}|$$

Mean square Error + Reducing overfitting + Feature Selection

λ_1, λ_2 = hyperparameter Tuning