Linear Regression

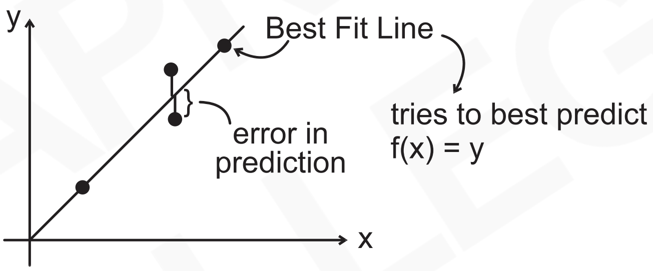
Intuition & Logic

Linear Regression is a supervised ML algorithm for regression problems.

LR models the relationship between a dependent variable (output) and one or more independent variables (inputs) by fitting the best straight line (or plane/hyperplane) to the data.

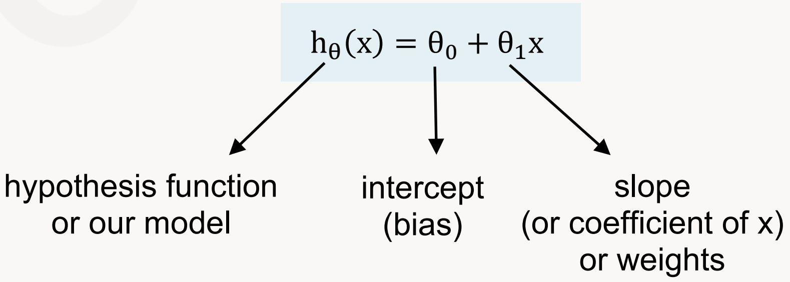
In the simplest form the linear regression can be understood by taking an example of single input feature (x) & output (y).

LR is a regression algorithm that tries to predict the relation of x & y in the form of a BEST FIT LINE.



eq^n of straight line ⇒ y = mx + c
which gives us our LR hypothesis function eq^n :-

$$y = c + mx$$



$$h_\theta(x) = \theta_0 + \theta_1 x$$

hypothesis function or our model

intercept (bias)

slope (or coefficient of x) or weights

When we have multiple independent features, it is called multiple linear regression:-

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

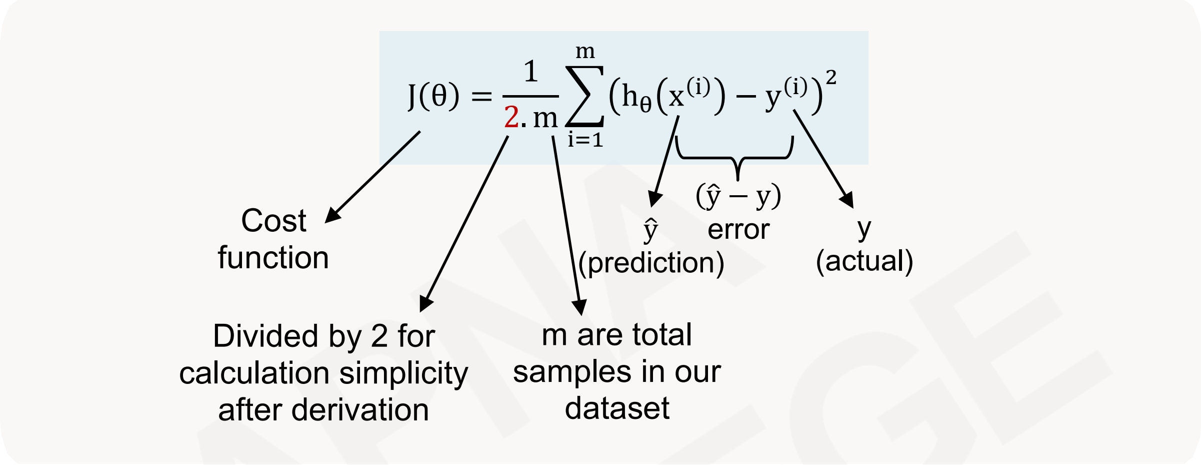
Goal: Best estimate all θ_i so that best fit line (or plane) best predicts y.

How to find this Best fit line?

We find it by trying to minimize the **Cost function**.

Cost function is a function that measures how far the predicted values $(\hat{y})$ are from actual values (y).

Most common CF used for LR is Mean Squared Error (MSE) :-



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

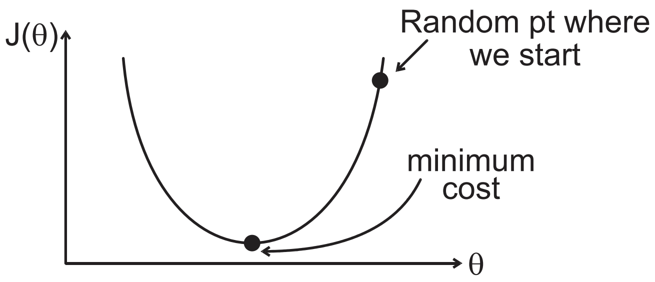
Cost function
$\hat{y}$ (prediction)
error
y (actual)
Divided by 2 for calculation simplicity after derivation
m are total samples in our dataset

Now that we know we have to minimize cost function, we do so by using a technique called **Gradient Descent**.

GD is an iterative technique that iteratively updates θ_0 & θ_1 until the MSE (or $J(\theta)$) reaches its lowest value.

How does Gradient Descent for LR work?

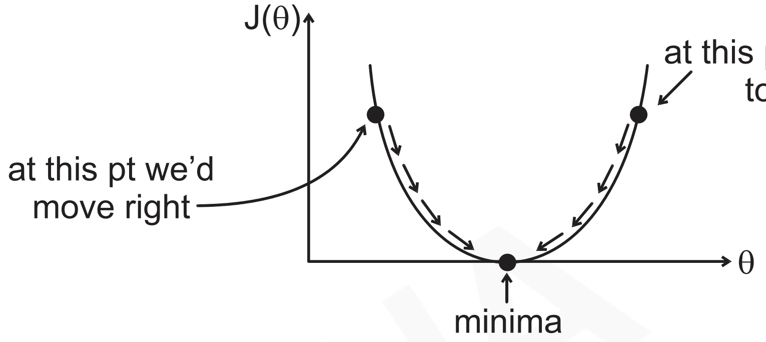
GD is an optimization technique used to train our LR model by minimizing prediction error.
To understand GD, let's plot θ & $J(\theta)$



we get such a curve. Minimum cost is at the Global Minima. We iteratively try to converge to this value using GD.

How? Using these steps:-

1. Start with random value of θ_0 & θ_1.
2. Calculate the error between j & y using MSE i.e. $J(\theta)$
3. Compute Gradient i.e. derivative of cost function. Why? Because it is essentially the slope which will point in the direction of the steepest increase


J(θ)
at this point we'd need to move left
at this pt we'd move right
θ
minima

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

$$\text{we find } \frac{\partial J(\theta)}{\partial \theta_0} \& \frac{\partial J(\theta)}{\partial \theta_1}$$

Extra

$$\text{if } J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left((\theta_0 + \theta_1 x_i) - y_i\right)^2$$

$$\text{so } \frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\cdot\sum_{i=1}^{m}(\underbrace{(\theta_0 + \theta_1 x_i)}_{\hat{y}_i} - y_i)$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{m}\sum_{i=1}^{m} x_i(\underbrace{(\theta_0 + \theta_1 x_i)}_{\hat{y}_i} - y_i)$$

4. Update Parameters θ_0 & θ_1 to reduce the error

learning rate

$$\theta_k = \theta_k - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta_k}$$

convergence theorem to update θ_0 & θ_1

Interpretation

$$\theta_1 = \theta_1 - \alpha \cdot (\text{slope})$$

α i.e. Learning rate controls size of each update.

$$\theta_1 = \theta_1 - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta_1}$$

$$\theta_0 = \theta_0 - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta_0}$$

5. Keep repeating this process (Steps 2 to 4) until error stops decreasing significantly.

Special Note - For simple linear regression we can use formulas like Normal Equation i.e.

$$\theta = (X^T X)^{-1}.(X^T y)$$

to find parameters directly (without Gradient Descent). So sklearn directly estimates the coefficients using the Ordinary least squares (OLS) method.

However for large datasets or high-dimensional data these methods become computationally expensive that's why we need Gradient Descent.

Also, in polynomial regression, the cost function becomes highly complex and non-linear, so analytical solutions are not available. That's where gradient descent plays an important role.

Linear Regression Assumptions

Every ML model has some assumptions which are foundational conditions that must hold true for the model's results to be reliable, accurate & generalizable.

Linear regression relies on several key assumptions, often remembered by the acronym **LINE** (with some extensions):

- **L - Linearity:** The relationship between the features and target.
- **I - Independence:** Observations are independent of each other.
- **N - Normality:** The error(residuals) follows a normal distribution.
- **E - Equal Variance (Homoscedasticity):** The error term has a constant variance.
- **Multicollinearity:** There is no multicollinearity between the features.

We can breakdown these into different categories:

Assumptions about the **residuals**:

- **Normality assumption:** The error terms, ε(i), are normally distributed.
- **Zero mean assumption:** The residuals have a mean value of zero.
- **Constant variance assumption:** The residual terms have the same (but unknown) value of variance, σ^2. This assumption is also called the assumption of homogeneity or homoscedasticity.
- **Independent error assumption:** The residual terms are independent of each other, i.e. their pair-wise covariance value is zero.

Assumptions about the **estimators**:

- The independent variables are measured without error.
- There does not exist a linear dependency between the independent variables, i.e. there is no multicollinearity in the data.

> *Keep Learning & Keep Exploring!*