Simple Linear Regression

It is a supervised machine learning algorithm used to predict a continuous target variable based on the values of independent variable. The main goal is to find the best fit line that represents the relationship between the input and the output data.

Best fit line

↳ Best captures the overall trend of the data.
↳ Minimizes the error between actual and predicted value.
↳ helps to predict future outcomes for unseen value of x.

The general form of the line is:

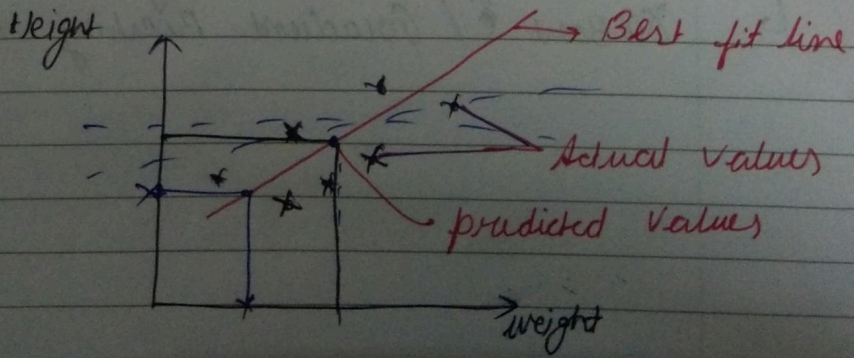
$$\hat{y} = w_0 + w_1 x$$

where,

$\hat{y}$: Predicted value
w_0: Intercept (bias)
w_1: slope (weight)
x: independent variable

Actual value (y): The real output from the dataset.

Predicted value ($\hat{y}$): The value predicted by the regression model.



As we know the equation of simple linear regression model is

$$\hat{y} = w_0 + w_1 x$$

we have to calculate w_0 and w_1.

Eg:- Suppose we have a dataset and we want to predict a person height (in cm) based on their weights (in kg).

Person	weight x	Height y
A	50	160
B	60	163
C	70	175
D	80	170
E	90	180

Suppose after training the model, we get

$$\hat{y} = 145 + 0.4x$$

$\therefore w_0 = 145$ (Intercept)

$w_1 = 0.4$ (slope)

weight(x)	Actual height (y)	Predicted Height ($\hat{y}$)	Residual ($y - \hat{y}$)
50	160	165.0	-5.0
60	163	169.0	-6.0
70	175	173.0	2.0
80	170	177.0	-7.0
90	180	181.0	-1.0

Negative Residual → Prediction too high

Positive Residual → Prediction too low.

* Mean Squared Error (MSE)

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = 23.0$$

so, the average squared error made by the model on the dataset is $23.0 cm^2$

Now, there are two ways to find the value of w_0 and w_1

1) 2)

Now, we want the best fit-line $\hat{y} = w_0 + w_1 x$ that minimizes the prediction error b/w actual value y_i and predicted value $\hat{y}_i$.

∴ we use Mean Squared Error (MSE) as the cost function:

$$J(w_0, w_1) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

cost function

Now, there are two ways to minimize the cost function.

1) Gradient Descent (iterative)
2) Normal equation (closed form)

1) Gradient descent (convergence algorithm)

Gradient Descent is an optimization algorithm used to minimize the cost function in machine learning and deep learning models.

Intuition behind Gradient descent

Imagine you are on a mountain and it's dark. You want to reach the lowest point (valley) using just your sense of slope under your feet.

→ Each step you take is in the opp. direction of the slope (gradient)

→ The steepness of the slope tells you how far to move

→ you repeat until you reach a flat point (minimum)

Mathematical Formulation

Let's say we are trying to minimize a cost-function $J(\theta)$, where θ are the parameters (like weights in linear regression.

we updat the parameters using:

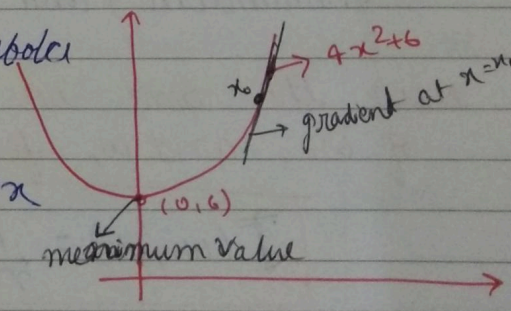
$$\theta : \theta - \alpha \nabla J(\theta)$$ → updat formula

where;
→ θ : Parameters (like weights $w_0, w_1, \ldots$)
→ α : Learning rate (step size)
→ $\nabla J(\theta)$: Gradient (partial derivative of cost fn w.r.t θ)

Proof of the formula using example

We are going to use gradient descent to minimize the function $f(x) = 4x^2 + 6$.

Shape ⇒ U shaped parabola
minimum ⇒ at $x = 0$
value at minimum ⇒ 6
derivative slope : $f'(x) \Rightarrow 8x$



Now, we use the formula

$$x_{new} = x - \alpha f'(x)$$

Iteration 1:

Let's take $x = 3$

$\therefore$ slope $= 8 \times 3 = 24$

$\therefore x_1 = 3 - 0.2 \times 24 = 0.6$

$\therefore$ New value $= f(0.6) = 7.44$

Iteration 2

→ $x = 0.6$

→ slope $= 0.6 \times 8 = 4.8$

→ $x_2 = 0.6 - 0.1 \times 4.8 = 0.12$

$f(x_2) = f(0.12) = 6.0576$

Iteration 3

→ $x = 0.12$

→ slope $= 0.12 \times 8 = 0.96$

→ $x_3 = 0.12 - 0.1 \times 0.96 \cong 6.002304$

NOTE

1) α is too small ⇒ Slow convergence (takes forever)

2) α is too large ⇒ overshoots minimum → diverge

3) α is just right ⇒ smooth convergence ⇒ find minimum

Result

After step 3

→ x changed : $3 \rightarrow 0.6 \rightarrow 0.12 \rightarrow 0.024$

→ $f(x)$ changed : $42 \rightarrow 7.44 \rightarrow 6.0576 \rightarrow 6.002304$

hence as $x \rightarrow 0$ $f(x) \rightarrow 6$

∴ the cost function $f(x)$ will be 6 which is minimum and it occurs at $x = 0$.

Now, the same we will find the ~~minima~~ value of w_0 and w_1 for the cost fn

$$J(w_0, w_1) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where $\hat{y}_i = w_0 + w_1 x_i$

Now, Gradient w.r.t w_0

$$\frac{\partial J}{\partial w_0} = -\frac{2}{n} \Sigma (y_i - \hat{y}_i)$$

Gradient w.r.t w_1

$$\frac{\partial J}{\partial w_0} = -\frac{2}{n} \Sigma (y_i - \hat{y}_i) \cdot x_i$$

Instead of keeping the 2 in the derivative, we absorb it into the learning rate α. This simplifies the gradients to:

$$\frac{\partial J}{\partial w_0} = \frac{-1}{n} \Sigma (y_i - \hat{y}_i)$$

$$\frac{\partial J}{\partial w_1} = \frac{-1}{n} \Sigma (y_i - \hat{y}_i) \cdot x_i$$

* Update rule

$$w_0 = w_0 + \alpha \cdot \frac{1}{n} \Sigma (y_i - \hat{y}_i)$$

$$w_1 = w_1 + \alpha \cdot \frac{1}{n} \Sigma (y_i - \hat{y}_i) \cdot x_i$$

Repeat until convergence like we did in example.

#2> Normal Equation (closed Form)

A closed form solution means a direct mathematical formula to compute the model parameter (weights) in one-step - no need for loop or iteration.

* Setup of Linear Regression in vector / Matrix form.

Suppose we are given

- Input matrix $X \in R^{m \times n}$
  Each row = one sample
  Each column = one feature
  (usually, a column of 1s is added to include the bias / Intercept)
- output vector $y \in R^m$
- weights vector $w \in R^n$

* Model Hypothesis

$$\hat{y} = Xw$$

The cost function will be:

$$J(w) = \frac{1}{2m} \left(\|y - Xw\| \right)^2$$

As we know

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

Now, Let say $a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \Rightarrow \|a\|^2 = a^T a = a_1^2 + a_2^2 + a_3^2$

So, when

$$y - Xw = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{bmatrix} \Rightarrow \|y - Xw\|^2 = \sum_{i=1}^{m} e_i^2$$

This exactly mean

$$(y - Xw)^T (y - Xw) = e_1^2 + e_2^2 + e_3^2 + \dots + e_m^2$$

$\therefore$ J_w becomes $\frac{1}{2m} (y - Xw)^T (y - Xw)$

$$\Rightarrow J_w = \frac{1}{2m} \left[y^T y - 2w^T X^T y + w^T X^T Xw \right]$$

taking partial derivative of $J(w)$ (∇) w.r.t w

$$\nabla_w J(w) \text{ or } \frac{\partial J(w)}{\partial w} = \frac{1}{2m} \left[-2X^T y + 2X^T Xw \right]$$

$$= \frac{1}{m} \left[X^T Xw - X^T y \right]$$

Set derivative to zero

$$\frac{1}{m}(X^T X w - X^T y) = 0 \Rightarrow$$

$$\Rightarrow X^T X w = X^T y$$

$$\therefore \boxed{w = (X^T X)^{-1} X^T y}$$

This gives optimal weight (w) in one shot (provided $X^T X$ is invertible)