XGBoost

XGBoost = Xtreme Gradient Boosting

XGBoost = Gradient Boosting with some rules and penality

→ Introduction to XGBoost

→ Key improvements over Gradient Boosting

  ↳ Regularization

  ↳ Second order optimization

→ XGBoost Training, Prediction & key Parameters

→ Implementing XGBoost on a Dataset

→ Model Evaluation, Model selection & Practical use case

Introduction to XGBoost

XGBoost = Gradient Boosting + Regularization

+ 2nd Order optimization

+ Fast engineering

$D = \{(x_i, y_i)\}$

Model prediction:

$\hat{y}_i = \sum f_t(x_i)$ $(t = 1 \text{ to } T)$

$$\hat{y}_i = \sum \underbrace{f_t(x_i)}_{\text{tree } t} \quad (t = 1 \text{ to } \underbrace{T}_{\text{number of trees}})$$

Objective of function:

$$\text{Obj} = \underbrace{\sum L(y_i, \hat{y}_i)}_{\text{Loss}} + \underbrace{\sum \Omega(f_t)}_{\text{regularization}}$$

$f_1 \; f_2 \; --- \; f_T$

regularization → reduce cheating

Regularization Term (What makes XGBoost disciplined)

$$\Omega(f) = \underbrace{\gamma}_{\text{leaf penalty}} \cdot \underbrace{K}_{\text{number of leaves}} + \underbrace{\frac{1}{2} \cdot \lambda \sum (w_j^2)}_{\text{L2 penalty}}$$

w_j → leaf weight

Here, $j = 1$ to K

Manual Step-by-Step Example (Regression, One Boosting Round)

Tiny Dataset:

i	x_i	y_i	$\hat{y}_{(0)}$	$g_i = 2(6 - y_i)$	h_i	$\hat{y}_{(1)}$
1	1	3	6	6	2	5.2
2	2	5	6	2	2	5.2
3	3	7	6	-2	2	6.8
4	4	9	6	-6	2	6.8

3	3	7	6	-2	2	6.8
→ 4	4	9	6	-6		

Loss Function:

$$L(y, \hat{y}) = (y - \hat{y})^2$$

Initial prediction:

$$\hat{y}(0) = \text{mean}(y)$$

$$(3+5+7+9)/4 = 6$$

Step 2: Gradients and Hessians

$$g_i = \partial L / \partial \hat{y}_i$$ → partial

$$h_i = \partial^2 L / \partial \hat{y}_i^2$$ → partial square

For square loss:

$g_i = 2(\hat{y}_i - y_i)$ → Direction

$h_i = 2$ → confidence

Step-3:

Make a simple tree stump

Try split: $x < 2.5$

Left leaf: $x = 1, 2$

Right ": $x = 3, 4$

Step 4: Leaf sums

$$G = \sum g_i$$

$$H = \sum h_i$$

Left Leaf: (points 1, 2)

$$G_L = 6 + 2 = 8$$

$$H_L = 2 + 2 = 4$$

Right Leaf: (points 3, 4)

$$G_R = -2 + -6 = -8$$

$$H_R = 2 + 2 = 4$$

Step-5: Compute optimal leaf weight

$$w^* = -G / (H + \lambda)$$

Compute with $\lambda = 1$

Left:

$$w_L = -8 / (4 + 1) = \frac{-8}{5} = -1.6$$

Right:

$$w_R = -(-8) / (4 + 1) = \frac{8}{5} = 1.6$$

Step-6: Update predictions using learning rate

$$\hat{y}_{(1)} = \hat{y}_{(0)} + \eta \cdot w_{leaf}$$

$\eta = 0.5$

Compute with $\eta = 0.5$

Left points (1, 2):

$$\hat{y} = 6 + 0.5(-1.6) = 6 - 0.8 = 5.2$$

...nts (3, 4):

$y = 6 + 0.5$

Right points (3, 4):

$$\hat{y} = 6 + 0.5(1.6) = 6 + 0.8 = 6.8$$