XGboost

July 31, 2025 1:09 PM

Extreme Gradient Boosting

Can be used for classification & regression ; takes in multiple input features (x) to predict a target (y)
Let's discuss the case of XGB Regressor.

e.g

Day (i)	x1 (Day of week)	x2 (Price)	x3 (Promotion)	y (Demand)
1	1	10	0	100
2	2	9	1	120
3	3	11	0	90
4	4	10	1	130
5	5	12	0	85

(1) Initialize Prediction

Usually called base prediction

Mean of targets

$\hat{y}_i^{(0)} = \bar{y} = \frac{100+120+90+130+85}{5} = 105$

($\hat{y}_i^{(0)}$: Base; $\hat{y}_i$: Prediction for i^{th} row; $\hat{y}_i$: 105)

(2) Squared Error Loss

$L = \frac{1}{2}(y_i - \hat{y}_i)^2$

(L: Loss function; y_i: Actual; $\hat{y}_i$: Prediction)

Gradient

$g_i = -\frac{\partial L}{\partial \hat{y}_i} = y_i - \hat{y}_i$

e.g

i	yi (Actual)	gi (gradient $(\hat{y}_i - y_i)$ $(105 - y_i)$)
1	100	5
2	120	-15
3	90	15
4	130	-25
5	85	20

$g_i = \hat{y}_i - y_i$

($\hat{y}_i$: Prediction; y_i: Actual)

$h_i = \frac{\partial^2 L}{\partial \hat{y}_i^2} = \frac{\partial}{\partial \hat{y}_i}\left(\frac{\partial L}{\partial \hat{y}_i}\right) = \frac{\partial}{\partial \hat{y}_i}(\hat{y}_i - y_i) = 1$

The goal is to improve prediction $\hat{y}_i$ at every step by adding a corrective factor

$\hat{y}^{(t)} = \hat{y}^{(t-1)} + f_t(x)$

($\hat{y}^{(t)}$: Prediction at step t; $\hat{y}^{(t-1)}$: Prediction at last step; $f_t(x)$: Corrective term)

$\hat{y}_i^t = \hat{y}_i^{t-1} + \eta w_j$

That is why we use gradient - to see where we going (steepest descent like gradient descent)

③ Tree

Left
Leaf 1
Day 1-2
$G_L = \sum_L g_i = -10$
$H_L = \sum_L h_i = 2$

Right
Leaf 2
Day 3-5
$G_R = \sum_R g_i = 10$
$H_R = \sum_R h_i = 3$

We measure how good this split is, for which we compute 'Gain'

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma$$

λ → Regularization

γ → Penalty term for split
If set to low - Splits are favored & can lead to complex trees
" high - simple trees

Split with most Gain is preferred

$$\omega_j = \frac{\sum_{I_j} g}{\sum_{I_j} h + \lambda}$$

If >0 underpredicting
<0 overpredicting

λ → Regularization

I_j → Set of samples in leaf

$\sum h$ → Total curvature (∵ 2nd derivative)
Confidence of g

Going to the right leaf

Right
Leaf 2
Day 3-5
$G_R = \sum_R g_i = 10$
$H_R = \sum_R h_i = 3$

$\omega_j = \frac{10}{4} = 2.5$

∴ all samples in this leaf would have prediction decreased by 2.25 ($\lambda = 0$)

Left
Leaf 1
Day 1-2
$G_L = \sum_L g_i = -10$
$H_L = \sum_L h_i = 2$

$\omega_j = \frac{-10}{3} = -3.33$

④ Update Predictions

$$\hat{y}_i^{(1)} = \hat{y}_i^{(0)} + \eta \omega_j$$

η → Learning Rate
ω_j → Weight
$\hat{y}_i^{(1)}$ → Prediction in next iteration (First)
$\hat{y}_i^{(0)}$ → Prediction previous iteration (Base)

Right

$\hat{y}_i^{(1)} = \hat{y}_i^{(0)} + \eta \omega_j$
$y_i^{(1)} = 105 + (0.1)(2.5) = 105.25$

Left

$y_i^{(1)} = 105 + (0.1)(-3.33)$
$= 104.667$

Repeat

$$\hat{y}_i = \hat{y}_i^{(0)} + \sum_{t=1}^{T} \eta f_t(x_i)$$

Actual eq. with Regularization

Objective Function

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \right] + \Omega(f_t)$$

- $f_t(x_i)$: Output of new tree
- ω_j
- g_i: Gradient of loss w.r.t $\hat{y}_{t-1}$
- h_i: Hessian
- $\Omega(f_t)$: Regularization $\gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2$

Taylor Approx :

$$g_i = \frac{\partial L}{\partial \hat{y}_i^{t-1}}, \quad h_i = \frac{\partial^2 L}{(\partial \hat{y}_i^{t-1})^2}$$

So, its in the form of taylor series

The advantage of using this taylor series approx. is you don't need to define exact loss function