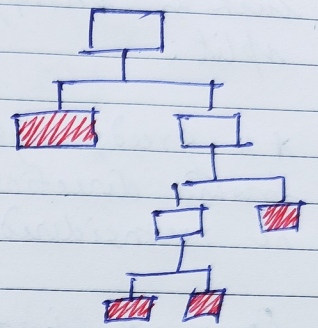
Xgboost (Extreme Gradient Boosting)

Gradient Boosting Machine (GBM).

- Decision tree creates disjoint regions (R_j). [or leaves].



- For each region (or leaf) there is a constant γ_j which is responsible for the final prediction.

Hence for any x belonging to any R_j, $j = 1, 2, 3 \ldots$ the prediction is

# of leaves = 4.
so, $J = \{1, 2, 3, 4\}$

prediction of a tree.

$$x \in R_j \Rightarrow f(x) = \gamma_j$$

So, a tree can be formally represented as a mathemat form as,

Identity function.

$$T(x; \theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j)$$

[x can belong to only one of the many R_j ~~values~~ regi
Hence $I(\cdot)$ function will return 1 only when x is belonging to a specific leaf whose corresponding γ_j is meant for the prediction.

Eg. Let x is belonging to the 2nd leaf. Hence ~~pred~~ prediction of the tree is

Work to do

$$\gamma_1 \underbrace{I(x \in R_1)}_{0} + \gamma_2 \underbrace{I(x \in R_2)}_{1} + \gamma_3 \underbrace{I(x \in R_3)}_{0} + \gamma_4 \underbrace{I(x \in R_4)}_{0}$$

θ is the parameter of the tree

$$\theta = \{R_j, \gamma_j\}$$

R_j is found out using some greedy algorithm such as entropy based split or gini impurity based split.

** greedy algorithm because it only moves in the forward direction and once a split is done, it will never be reconsidered later on for a change.

Once R_j is found, finding γ_j is simple:

$\gamma_j = \bar{x}$ for <u>regression</u> <u>model</u> where

$$\bar{x} = \frac{1}{N_j} \sum_{x_j \in R_j} x_j$$

γ_j = modal class for classification model.

θ is optimally found out by lowering some loss function $L(y_i, \gamma_j)$ where y_i is the actual value and γ_j is the predicted value.

Remember: γ_j is the prediction of the tree for a particular $x \in R_j$

$$\theta^* = \arg\min_{\theta} \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y_i, \gamma_j)$$

see the link.

A more convenient criteria is

$$\theta^* = \underbrace{\arg\min_{\theta} \sum_{i=1}^{N} L(y_i, T(x_i, \theta))}_{\text{Empirical loss minimization.}}$$

A boosted tree is the sum of such trees.

$$F_M(x) = \sum_{m=1}^{M} T(x; \theta_m).$$

[In Adaboost, each tree is weighed by its perf which is not done in GBM with trees.

In boosted trees, trees are induced in sequential manner such that each ~~subj~~ subsequent tree tries t lower the error produced by the previous set of trees.

Hence, the optimal parameter for an mth tree is found out by lowering the loss function as shown b

$$\theta_m^* = \arg\min_{\theta} \sum_{i=1}^{N} L\Big(y_i, \underbrace{F_{m-1}(x_i)}_{\text{prediction by previous set of trees.}} + \underbrace{T(x_i; \theta_m)}_{\text{Current tree.}}\Big).$$

Work to do

$$\theta_m = \{ R_{jm}, \gamma_{jm} \}.$$

When R_{jm} is found out, finding γ_{jm} is straight forward.

$$\gamma_{jm}^* = \underset{\gamma_{jm}}{\arg\min} \sum_{i=1}^{N} L\left(y_i, F_{m-1}(x_i) + \gamma_{jm}\right)$$

As per the above equation, if loss function is a continuous function of parameters, the current tree is reqd. to be generated ~~along the one~~ such that the predictions of the current tree matches the -ve gradient of the loss function.

Let g_{im} is the gradient calculated for the mth tree at the ith data point.

$$\therefore g_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x_i) = F_{m-1}(x_i)}.$$

Thus ~~f~~ ~~is~~ using gradient descent algorithm, if the parameters of the current tree is to be estimated, the eq^n would be,

Work to do

$$\theta_{im}^* = \underset{\theta_{im}}{\arg\min} \sum_{i=1}^{N} L\left(-g_{im}, T(x_i, \theta_{im})\right).$$

For regression,

L = Squared loss $= \frac{1}{2}[y_i - f(x_i)]^2$

$\therefore -g_i = [y_i - f(x_i)]$ = Residual after $m-1$ trees

$f(x_i) = F_{m-1}(x_i)$

For classification:

Build K no of trees (where K = no of classes) parallelly at the mth level. Each tree corresponding to only one ~~class~~ of the K classes.

L = deviance loss.

$K=1 \xrightarrow{f_1(x_i)} v_1 \longrightarrow \frac{e^{v_1}}{\sum_{i=1}^{K} e^{v_i}}$

$K=2 \xrightarrow{f_2(x_i)} v_2 \xrightarrow{\text{apply softmax}} \text{—do—}$

$K=3 \xrightarrow{f_3(x_i)} v_3 \xrightarrow{\text{—do—}} \text{— —}$

$K=4 \xrightarrow{f_4(x_i)} v_4 \xrightarrow{\text{—do—}} \text{— —}$

$K=5 \xrightarrow{f_5(x_i)} v_5 \xrightarrow{\text{—do—}} \text{— —}$

Softmax.

$$p_k(x) = \frac{e^{f_k(x_i)}}{\sum_{l=1}^{K} e^{f_l(x_i)}}$$

Calculate deviance loss based on these probability values.

Deviance loss $= L(y, p(x))$

$$= -\sum_{k=1}^{K} I(y = C_k) \log p_k(x)$$

$$= -\sum_{k=1}^{K} I(y = C_k) \log \frac{e^{f_k(x)}}{\sum_{l=1}^{K} e^{f_l(x)}}$$

$$= -\sum_{k=1}^{K} I(y = C_k) \left[\log e^{f_k(x)} - \log\left(\sum_{l=1}^{K} e^{f_l(x)}\right) \right]$$

Work to do

$$L = -\sum_{k=1}^{K} I(y=C_k)\left[f_k(x) - \log\left(\sum_{l=1}^{K} e^{f_l(x)}\right)\right]$$

$$= -\sum_{k=1}^{K} I(y=C_k) f_k(x) + \sum_{k=1}^{K} I(y=C_k) \log\left(\sum_{l=1}^{K} e^{f_l(x)}\right).$$

$$\therefore \frac{-\partial L}{\partial f_k(x)} = I(y=C_k) - \frac{\partial}{\partial f_k(x)} \log\left(\sum_{l=1}^{K} e^{f_l(x)}\right).$$

-ve gradient of the deviance loss func^n.

$$= I(y=C_k) - \frac{1}{\sum_{l=1}^{K} e^{f_l(x)}} \cdot \frac{\partial}{\partial f_k(x)} \cdot \left(\sum_{l=1}^{K} e^{f_l(x)}\right)$$

$$= I(y=C_k) - \frac{1}{\sum_{l=1}^{K} e^{f_l(x)}} \cdot e^{f_k(x)}$$

$$= I(y=C_k) - p_k(x).$$

Hence each of the K trees will be built to ~~lower~~ match the corresponding -ve gradient as mentioned above.