Gradient-boosting (Analytics Vidhya blog + statquest)

Adaboost VS Gradient-boosting

Adaboost:

↓

we build "stumps"

↓

amount of say of new "stump" of how well it compensates prev. errors

↓

builds next "stump" and so on.

Gradient-boosting:

first we make a single leaf (represents initial guess)

↓

The gradient-boost builds a new "tree"

↓

Tree is built on errors made by previous trees. Tree is typically larger than a stump.

↓

Gradient boost also scales the trees but it scales them ~~on~~ by a same amount.

Gradient-boosting

GB regression

↓

when target variable is continuous

GB classifier

↓

when target variable is discrete.

"Gradient-boosting" → As objective is to minimise loss function by adding weak learners using "gradient-descent"

as scale is the same.

Steps in GB algo :→

step 1) First we build a base model. (single leaf). For simplicity we take the avg. of target column.

why avg. ?

$$\underbrace{F_0(x)}_{\text{Base model}} = \underset{\underbrace{\gamma}_{\text{we pick such a value that min. loss function.}}}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$$

step 2) Then we calculate the pseudo residuals ⇒ (observed val − pred. val)

why (observed − predicted) ?

$$\underbrace{r_{im}}_{\text{residual for ith datapoint for mth model}} = -\underbrace{\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]}_{\text{"gradient descent type algo."}}_{F(x) = \underbrace{F_{m-1}(x)}_{\text{the previous model}}} \quad \text{for } \overbrace{i = 1, 2, \ldots, n}^{n \text{ datapoints}}$$

$$\frac{\partial L}{\partial \gamma} = -(y_i - \gamma) \quad \text{hence} \quad r_{im} = (y_i - \gamma) = (\text{obs.} - \text{pred.})$$

we use above $\underbrace{\text{residuals}}_{\text{we use residuals as our goal is to decrease these residuals.}}$ to make the ~~next~~ mth model.

step 3) Let's say $\underline{\underline{h_m(x)}}$ is our decision tree made on ~~pser~~ residuals.

In this step we find output value for each leaf of ~~st~~ our dec. tree.
If there is a case ~~wt~~ where a leaf gets more than one ~~dec~~ ~~tatt~~ residual → we only take the final output of that leaf

↓

which is the avg. of all residuals in that leaf

↓

why avg?

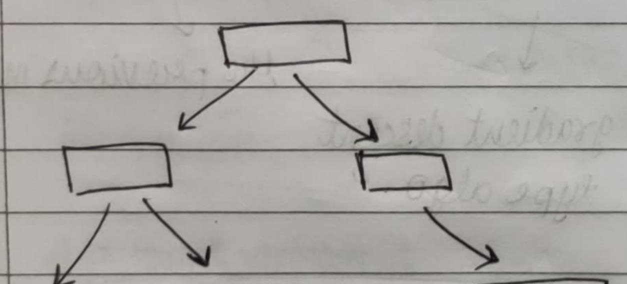
↓

$$\text{for } j = 1, 2, \dots, J_m \quad \gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma\, h_m(x_i)\right)$$

γ_{jm} → output of a particular jth leaf.

$h_m(x_i)$ → $h_m(x_i)$ is the dec. tree on passing x_i datapoint
→ 1 for the leaf it ends up in
→ 0 for all other leaves.

suppose



$R_{1,1}$ $R_{2,1}$ $R_{3,1}$ → 1st model
↓
3rd leaf

$$\gamma_{2,1} = \underbrace{\frac{\sum a_i}{n}}$$

$\gamma_{2,1}$ → output of $R_{2,1}$

↓

proof using above formula → $h_m(x_i) = 1$ for all which reach $R_{2,1}$

$h_m(x_i) = 0$ for all datapoints who don't reach $R_{2,1}$

→ derivative says

$\gamma_{2,1}$ = avg. of all residuals in $R_{2,1}$

step 4) We update the predictions of the prev. model.

$$F_m(x) = F_{m-1}(x) + \nu_m h_m(x)$$

$F_m(x)$ → new prediction

$F_{m-1}(x)$ → prev. prediction

ν_m → learning rate → if learning rate = 1 → variance is high → "overfitting of the data."

$h_m(x)$ → output of tree made on residuals.

step 5) Then we recalculate the residuals = (observed − predicted) and repeat from step 3. We do until we reach the no. of trees. limit or exactly fit the data.

more mathematically

input: Data $\{(x_i, y_i)\}_{i=1}^{n}$ and a differentiable Loss Function $L(y_i, F(x))$

step 1) Initialise model with constant value:

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$$

step 2) for $m = 1$ to $\overset{\text{no. of trees in our GB algo}}{M}$:

A) compute $r_{im} = -\left[\dfrac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x) = F_{m-1}(x)}$ for $i = 1, 2, \ldots, n$

B) Fit a regression tree to the r_{im} values to get dec. tree $h_m(x)$ and terminal region R_{jm} for $j = 1, \dots, J_m$

no. of leaves in mth tree.

C) For $j = 1, \dots, J_m$ compute

$$\gamma_{jm} = \underset{\gamma}{\text{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$$

only the datapoints that reach R_{ij}

D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{I}(x \in R_{jm})$

the dec. tree we made

summation is just in case a datapoint ends up in multiple leaves

then we sum up the associated γ's (outputs) of such leaves.

<u>Gradient boosting classifier</u> (from analytics vidhya)

This is used when target column is binary.
Same as GB regressor except log likelihood is the loss function.

$$L = -\left[\sum_{i=1}^{n} y_i \log(p_i) + (1-y_i)\log(1-p_i)\right]$$

$$L = -\left[\sum_{i=1}^{n}\left(y_i \log(odds_i) - \log\left(1 + e^{\log(odds_i)}\right)\right)\right]$$

we minm. wrt log(odds) ⇒ makes calculations simpler.

PAGE No | DATE

$$\frac{\partial L}{\partial \log(odds_i)} = -y_i + p_i$$

$$\boxed{odds_i = \frac{p_i}{1-p_i}}$$

$$r_{i,m} = -\left[\frac{\partial L}{\partial \log(odds_i)}\right] = y_i - p_i = \text{observed} - \text{predicted}$$

Now we build a decision tree. If a leaf has more than one residual we use the following formula

$$\gamma = \frac{\sum_{i=1}^{n} \text{Residual}_i}{\sum_{i=1}^{n} [\text{previous probability}_i \times (1 - \text{previous probability}_i)]}$$

output of leaf with multiple residuals.

GB classifier (from statquest)

Follow exact same mathematical steps as GB regressor with different function.