

Gradient Boost

Gradient Boost algorithm belongs to the class of boosting ensemble algorithms where base models/weak learners are trained in a sequential manner such that each new learner tries to correct the errors made by the previous ones. Gradient Boost also follows the **Stagewise Additive Method**.

It can be used ^{for} both classification and regression.

In Gradient Boost, we use decision trees as the weak learner but here the depth of tree is not limited to 1 i.e. they are not decision stumps as they were in case of AdaBoost algorithm. Here, we can decide the number of leaves our decision tree can have, which is often between 8 to 32 but can be tuned using hyper parameter tuning.

The loss function used in gradient boosting algorithm is MSE in case of regression problems and log-loss (or cross-entropy) in case of classification problems.

The final prediction is made by summing up the contributions of all the individual models (usually decision trees) that have been trained in sequence. Here's the general formula for making a prediction:

$$\hat{y} = F_M(x) = F_0(x) + \sum_{i=1}^M f_m(x)$$

where, \hat{y} is the final prediction for input x .

→ $F_M(x)$ is the final model after M iterations (i.e., after training M trees).

→ $F_0(x)$ is the initial prediction, which is often the mean of target values in case of regression or log-odds in case of classification.

→ $f_m(x)$ is the prediction from the m -th decision tree trained at the m th iteration.

tree of residuals

- α is the learning rate, a scaling factor (typically between 0 and 1) that controls the contribution of each tree to the final prediction.
- M is the total number of trees in the ensemble.

The formula allows gradient boosting to iteratively improve the model by reducing the pseudo-residuals at each step, leading to a more accurate final prediction.

Regression using Gradient Boost

Steps for doing regression using gradient boost:-

1. Initialize $f_0(x) = \operatorname{argmin}_f \sum_{i=1}^n L(y_i, f)$

2. For $m=1$ to M :
 - (a) For $i=1, 2, \dots, N$ compute

pseudo-residual $r_{im} = - \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f=f_{m-1}}$

- (b) Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm}; j=1, 2, \dots, J_m$

- (c) For $j=1, 2, \dots, J_m$ compute

$$Y_{jm} = \operatorname{argmin}_{y \in R_{jm}} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + y)$$

- (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} Y_{jm} I(x \in R_{jm})$

- (3) Output $\hat{f}(x) = f_M(x)$

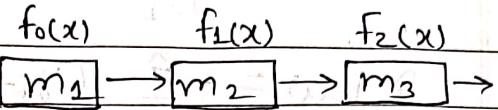
Input: training set $\{(x_i, y_i)\}_{i=1}^n$, a differentiable loss function $L(y, f(x))$, number of iterations M .

actual predicted

Let's understand this using an example taking a small dataset:

iq	cgpa	Salary
90	8	3
100	7	4
110	6	8
120	9	6
80	5	3

For the sake of simplicity and since our dataset is a toy dataset, we will be using only three weak learners.



Input: As we discussed to use gradient boost we require a training set $\{(x_i, y_i)\}_{i=1}^n$. In our context x_i is the features where x_1 denotes first row inputs, x_2 denotes second row in the dataset and so on. We have $n=5$ in our dataset.

$$x_1 = (90, 8), x_2 = (100, 7) \text{ and so on.}$$

y_i denotes the output / target where y_1 denotes output of first row, y_2 denotes output of second row - so on. Here, $y_1 = 3, y_2 = 4, y_3 = 8 \dots y_5 = 3$.

Loss \rightarrow We also require a differentiable loss function which needs to be differentiable at every point.

The loss function, we use typically in regression problems is least squares i.e.,

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

Since, we have decided to train 3 weak learners

$$\text{our } M = 3.$$

Step 1. We have to initialize

$$f_0(x) = \operatorname{argmin}_Y \sum_{i=1}^n L(y_i, Y)$$

$$f_0(x) = \operatorname{argmin}_Y \frac{1}{2} \sum_{i=1}^n (y_i - Y)^2$$

This equation means, we want to find a value of Y such that value of loss becomes minimum. We can do it by differentiation w.r.t. Y and equating it to 0.

$$\begin{aligned}
 \frac{d f_0(x)}{d y} &= \frac{d}{d y} \frac{1}{2} \sum_{i=1}^n (y_i - y)^2 \\
 &= \frac{1}{2} \sum_{i=1}^n \frac{d}{d y} (y_i - y)^2 \\
 &= \sum_{i=1}^n (y_i - y) \frac{d}{d y} (y_i - y) \quad \{ \text{Using chain rule} \}
 \end{aligned}$$

$\frac{d f_0(x)}{d y} = - \sum_{i=1}^n (y_i - y)$, now we need to equate this to 0 to find the minimum value.

$$\Rightarrow - \sum_{i=1}^n (y_i - y) = 0$$

$\Rightarrow \sum_{i=1}^n (y - y_i) = 0$, since in our example dataset we have $n=5$, so we can write it as-

$$\Rightarrow \sum_{i=1}^5 (y - y_i) = 0$$

$$\Rightarrow (y - 3) + (y - 4) + (y - 8) + (y - 6) + (y - 3) = 0$$

$$\Rightarrow 5y = 24 \Rightarrow y = \frac{24}{5} = 4.8$$

i	gpa	salary	f ₀ (x)
90	8	3	4.8
100	7	4	4.8
110	6	8	4.8
120	9	6	4.8
80	5	3	4.8

This implies $f_0(x) = 4.8$, which is nothing but the mean of the output / target column.

* In gradient boost, the first weak learner gives the prediction as mean value of target column and it is known as leaf. It does not have any ML model involved.

Step 2: For $m=1$ to 3 (as M is 3 in our case)

(a) For $i=1$ to n , compute

$$r_{im} = - \left[\frac{\partial L(y_i, f(x))}{\partial f(x_i)} \right] \quad f = f_{m-1}$$

→ this means for every model (m), for every row (i), we need to calculate a pseudo-residual. Since we are going to use first model, so $m=1$, hence.

$$r_{i1} = - \left[\frac{\partial L(y_i, y_i)}{\partial y_i} \right] \quad [\text{In our case, } y = f(x)] \quad f = f_{1-1} = f_0$$

Also, we have 5 rows in our dataset, so we want to calculate $r_{11}, r_{21}, r_{31}, \dots, r_{51}$.

r_{11} means for first model first row pseudo-residual.

r_{51} means for first model fifth row pseudo-residual.

Replacing the value of loss function in the equation.

$$r_{i1} = - \left[\frac{\partial}{\partial y_i} \frac{1}{2} (y_i - y_e)^2 \right]_{f=f_0}$$

$$\Rightarrow r_{i1} = - \left[\frac{1}{2} \times 2 (y_i - y_e) \frac{\partial}{\partial y_i} (y_i - y_e) \right]_{f=f_0}$$

$$\Rightarrow r_{i1} = - \left[(y_i - y_e) \cdot (-1) \right]_{f=f_0}$$

$$\Rightarrow r_{i1} = [y_i - y_e]_{f=f_0}$$

$$\Rightarrow r_{i1} = [y_i - f(x_i)]_{f=f_0} \quad (\because y = f(x))$$

$$\Rightarrow r_{i1} = [y_i - f_0(x_i)]_{f=f_0}$$

$$\therefore r_{11} = y_1 - f_0(x_1), r_{21} = y_2 - f_0(x_2), \dots, r_{51} = y_5 - f_0(x_5)$$

$$\Rightarrow r_{11} = 3 - 4.8 = -1.8, r_{21} = 4 - 4.8 = -0.8, r_{31} = 8 - 4.8 = 3.2$$

$r_{41} = 6 - 4.8 = 1.2, r_{51} = -1.8$, therefore our dataset

now is -	iq	cgpa	Salary	$f_0(x)$	r_{i1}
90	8	3	4.8	-1.8	
100	7	4	4.8	-0.8	
110	6	8	4.8	3.2	
120	9	6	4.8	1.2	
80	5	3	4.8	-1.8	

(b) Now, we have to fit a regression tree using columns iq, cgpa, Salary as input and r_{i1} as the target. The decision tree will look like this

$$iq < 105$$

This decision tree divides our data into four terminal regions.

$$iq = 95$$

$$cgpa \leq 7.5$$

first and last

row fitting in

terminal region (2)

1

$$-1.8$$

$$-0.8$$

$$3.2$$

$$1.2$$

Terminal region

TR3

TR4

The end values we get

in leaf nodes is by

taking mean of all points lying in their corresponding region

(c) Now, for all terminal regions, $j = 1$ to 4, compute.

$$y_{jm} = \arg \min_y \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i + y))$$

$$\Rightarrow y_{j1} = \arg \min_y \sum_{x_i \in R_{j1}} L(y_i, f_{1-1}(x_i + y))$$

$$Y_{11} Y_{21} Y_{31} Y_{41} \Rightarrow y_{j1} = \arg \min_y \sum_{x_i \in R_{j1}} L(y_i, f_0(x_i + y))$$

for terminal

Let's calculate γ_{11} first,

$$\gamma_{11} = \operatorname{argmin}_{x_i \in R_{11}} \sum L(y_i, f_0(x_i) + \gamma)$$

We have to consider those points which belong to terminal region R_{11} (we have two such points).

$$\gamma_{11} = \operatorname{argmin}_{\gamma} \frac{1}{2} \sum_{i=1}^2 (y_i - (f_0(x_i) + \gamma))^2$$

$$\frac{\partial L}{\partial \gamma} = \sum_{i=1}^2 (y_i - (f_0(x_i) + \gamma)) \cdot (-1)$$

To get the minimum value, we replace. Equate it to 0.

$$\Rightarrow - \sum_{i=1}^2 (y_i - f_0(x_i) - \gamma) = 0$$

$$\Rightarrow y_1 - f_0(x_1) - \gamma + y_2 - f_0(x_2) - \gamma = 0$$

$$\Rightarrow 3 - 4 \cdot 8 - \gamma + 3 - 4 \cdot 8 - \gamma = 0$$

$$\Rightarrow 2\gamma = -3.6 \Rightarrow \gamma = -1.8$$

Similarly, $\gamma_{21} = \operatorname{argmin}_{x_i \in R_{21}} \sum L(y_i, f_0(x_i) + \gamma)$

$$\gamma_{21} = \operatorname{argmin}_{\gamma} \frac{1}{2} \sum_{i=1}^2 (y_i - (f_0(x_i) + \gamma))^2$$

\because since only one point in terminal region 2 we don't need Σ .

$$\frac{\partial L}{\partial \gamma} = \frac{1}{2} \times 2 [y_1 - (f_0(x_1) + \gamma)], \frac{\partial L}{\partial \gamma} = [y_1 - (f_0(x_1) + \gamma)]$$

$$\frac{\partial L}{\partial \gamma} = [y_1 - f_0(x_1) - \gamma] \cdot (-1)$$

$$\Rightarrow y_1 - f_0(x_1) - \gamma = 0$$

$$\Rightarrow \gamma = -f_0(x_1) + y_1$$

$$\Rightarrow y_2 - 4 \cdot 8 + 4 = -0.8$$

Similarly, we can calculate $\gamma_{31} = 3.2$ & $\gamma_{41} = 1.2$

- * If we notice, the values of $\gamma_{11}, \gamma_{21}, \gamma_{31}$ & γ_{41} is exactly same as the values given by our decision tree leaf nodes. This happens because loss function we are using is least squares, if it was some other loss function these values would have been different.

(d) Now, $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} Y_{jm} I(x \in R_{jm})$
for us $m=1$, so

$$f_1(x) = f_0(x) + dt_1$$

this term is output of our decision tree.

$$\hat{f}(x) = f_1(x)$$

i.e., output of first model = $f_0(x) + dt_1$.

but we also use a learning rate, let it be α or $\gamma = 0.1$.

$$f_{\text{pred}} = f_1(x) = f_0(x) + \alpha dt_1 = m_1 + \alpha x m_2$$

The $f_1(x)$ is output of model m_2	iq	cgpa	salary	$f_0(x)$	r_{11}	$f_1(x)$	
	90	8	3	4.8	-1.8	4.62	$= 4.8 + 0.1 * (-1.8)$
	100	7	4	4.8	-0.8	4.72	$= 4.8 - 0.18$
	110	6	8	4.8	3.2	4.48	$= 4.62$
	120	9	6	4.8	1.2	4.68	Similarly, $4.8 + 0.1 * (-0.8)$
	80	5	3	4.8	-1.8	4.62	$= 4.8 - 0.08$

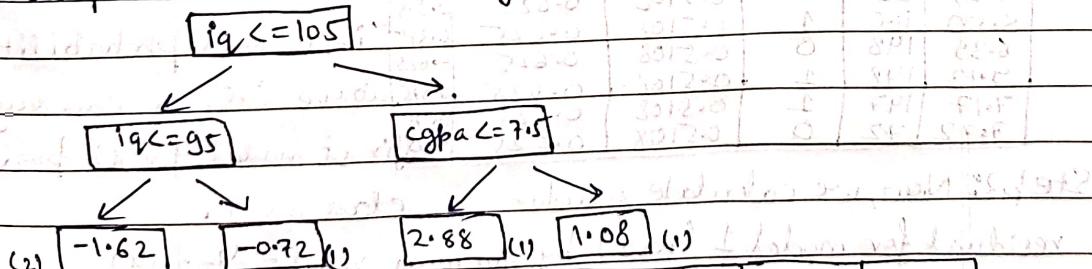
Now, we calculate the residual for model 2, which will be

	iq	cgpa	salary	$f_0(x)$	r_{11}	$f_1(x)$	r_{12}	
	90	8	3	4.8	-1.8	4.62	-1.62	
	100	7	4	4.8	-0.8	4.72	-0.72	
	110	6	8	4.8	3.2	5.12	2.88	
	120	9	6	4.8	1.2	4.92	1.08	
	80	5	3	4.8	-1.8	4.62	-1.62	

We can see for the second model, loss got reduced in each row.

Now, we need to repeat the entire process for model (m_3).

When we train the decision tree using inputs and output as r_{12} (residual of model 2) it will be like —



	iq	cgpa	salary	$f_0(x)$	r_{11}	$f_1(x)$	r_{12}	$f_2(x)$	r_{13}
	90	8	3	4.8	-1.8	4.62	-1.62	4.458	-1.458
	100	7	4	4.8	-0.8	4.72	-0.72	4.648	-0.648
	110	6	8	4.8	3.2	5.12	2.88	5.408	2.592
	120	9	6	4.8	1.2	4.92	1.08	5.028	0.972
	80	5	3	4.8	-1.8	4.62	-1.62	4.458	-1.458

Eventually, loss will decrease as we add more models.

Gradient Boosting for classification

Input: training set $\{(x_i, y_i)\}^n$, a differentiable loss function $L(y, F(x))$, number of iterations.

The loss function we use for classification problem is log-loss.
We will use the same steps we used for regression, one change is we are using different loss function.

Let's start with our dummy dataset -

Cgpa	iq	is placed	pre1(log-odds)
6.82	118	0	0.5108
6.36	125	1	0.5108
5.39	99	1	0.5108
5.50	106	1	0.5108
6.39	148	0	0.5108
9.13	148	1	0.5108
7.17	147	1	0.5108
7.72	72	0	0.5108

For this example, we will again be using combination of three models -

$$f_0(x) = f_1(x) + f_2(x)$$



In regression, we saw the first model we use is a simple

Step 1: model which predicted the mean of target column, but in case of classification the first model will simply predict log-odds i.e., $\log\left(\frac{p}{1-p}\right)$. Therefore, $\log\text{odds} = \sum \log_e\left(\frac{S_i}{3}\right)$

$$\Rightarrow \log_e\left(\frac{5/8}{1-5/8}\right) \Rightarrow \log_e\left(\frac{5/8}{3/8}\right) \Rightarrow \log_e\left(\frac{5}{3}\right) = 0.5108$$

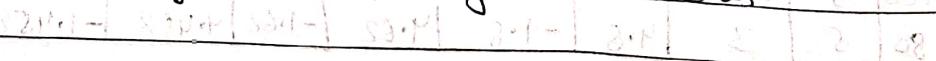
Step 2: Now, we need to get probability (p) for each log-odds, which is calculated using the formula $\Rightarrow p = \frac{1}{1 + e^{-\text{log-odds}}}$

Cgpa	iq	is placed	pre1(log-odds)	prob1 (probability)	res1
6.82	118	0	0.5108	0.625	-0.625
6.36	125	1	0.5108	0.625	0.375
5.39	99	1	0.5108	0.625	0.375
5.50	106	1	0.5108	0.625	0.375
6.39	148	0	0.5108	0.625	-0.625
9.13	148	1	0.5108	0.625	0.375
7.17	147	1	0.5108	0.625	0.375
7.72	72	0	0.5108	0.625	-0.625

Step 2: Now, we calculate pseudo-residual for model 1 (res1) by formula $\text{res1} = y - f_0(x)$

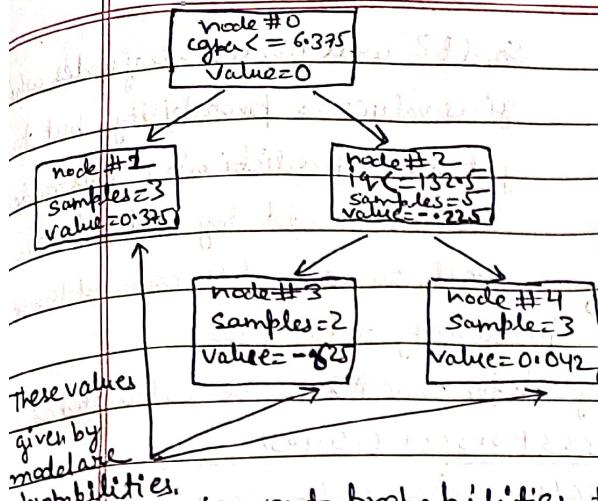
We will now train a regression decision tree using input columns and res1 as the output column as we were doing in case of regression also.

When we train a regression tree (weak learner having max-leaf-nodes = 3), we get the tree as -



Labels are not clearly visible in the sketch.

dt 1



In case of regression, we were directly taking these values as the prediction, but prediction for model 2, $f_1(x) = f_0(x) + \alpha dt_1$. Here $f_0(x)$ is log-odds and dt_1 produces probability, so they both can't be added directly, hence we need to convert probabilities to log-odds again. This is done using

Σ . Residual

$$\Sigma [Prev.\text{Prob} * (1 - Prev.\text{Prob})]$$

For using this formula, we need to know which sample was distributed in which terminal region/node.

cgbeta	ig	is placed	pred1 (log-odds)	pred1 probability	res1	node/ leaf	log-odds dt1	pred2 log-odds	pred2 prob.
6.82	118	0	0.5108	0.625	-0.625	3	-2.66	-2.159	0.103
6.36	125	1	0.5108	0.625	0.375	1	1.6	2.110	0.891
5.39	99	1	0.5108	0.625	0.375	1	1.6	2.110	0.891
5.50	106	1	0.5108	0.625	0.375	1	1.6	2.110	0.891
6.29	148	0	0.5108	0.625	-0.625	4	0.18	0.690	0.666
9.13	148	1	0.5108	0.625	0.375	4	0.18	0.690	0.666
7.17	147	1	0.5108	0.625	0.375	4	0.18	0.690	0.666
7.72	72	0	0.5108	0.625	-0.625	3	-2.66	-2.159	0.103

For node 3, let's calculate log-odds.

$$\frac{-0.625 - 0.625}{0.625(1-0.625) + 0.625(1-0.625)} = \frac{-2 \times 0.625}{2 \times 0.625 \times 0.375} = -2.658$$

Similarly, it can be calculated for node 1 & 4 as well.

Now, we calculate log-odds pred2 as $f_0(x) + \text{log-odds}(dt_1)$

$$\Rightarrow 0.5108 + (-2.658) = -2.15$$

$$\Rightarrow 0.5108 + 1.6 = 2.11 \text{ and so on. Now, we need to get probability}$$

again as shown above using pred2 log-odds.

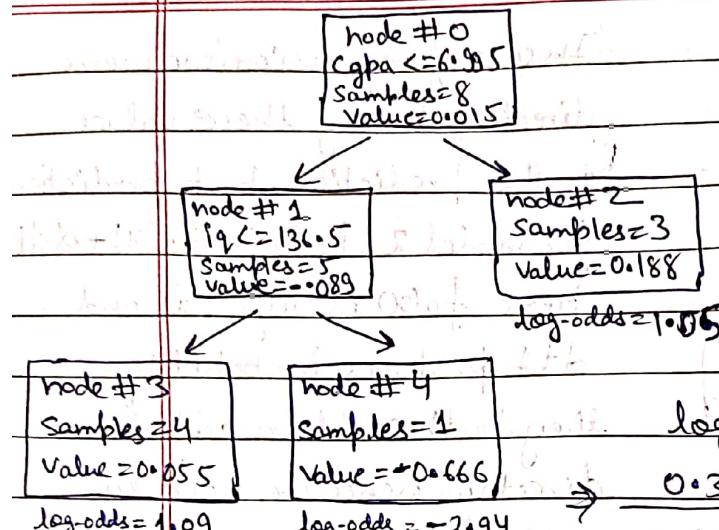
$$\Rightarrow \frac{1}{1+e^{-2.115}} = 0.103, \frac{1}{1+e^{-2.11}} = 0.891 \text{ and so on.}$$

This pred2 is the prediction of our model 2 i.e., $f_1(x)$.

Step3: Now, we again calculate residual for $f_1(x)$ and then fit a decision tree using inputs and res2 as target.

cgbeta	ig	is placed	pred1 log-odds	pred1 prob	res1	log-odds dt1	pred2 log-odds	pred2 prob	res2
6.82	118	0	0.5108	0.625	-0.625	-2.66	-2.159	0.103	-0.103
6.36	125	1	0.5108	0.625	0.375	1.6	2.110	0.891	0.891
5.39	99	1	0.5108	0.625	0.375	1.6	2.110	0.891	0.891
5.50	106	1	0.5108	0.625	0.375	1.6	2.110	0.891	0.891
6.29	148	0	0.5108	0.625	-0.625	0.18	0.690	0.666	-0.666
9.13	148	1	0.5108	0.625	0.375	0.18	0.690	0.666	0.333
7.17	147	1	0.5108	0.625	0.375	0.18	0.690	0.666	0.333
7.72	72	0	0.5108	0.625	-0.625	-2.66	-2.159	0.103	-0.103

dt 2



So, dt 2 will give us 3 leaf nodes which gives values as probability, but to get final prediction by this model we need to add log-odds of $f_1(x)$ as well, so we need to convert these probability to log-odds.

log-odds for node #9 can be calculated as -

$$0.333 + 0.333 + (-0.103)$$

$$0.666(1-0.666) + 0.666(1-0.666) + 0.103*(1-0.103)$$

$$\Rightarrow \frac{0.563}{0.444 + 0.092} \Rightarrow \frac{0.563}{0.536} \Rightarrow 1.05$$

After getting log-odds for each node, we add $f_1(x) + dt 2$ to get predicted log-odds for $f_2(x)$. Finally, we convert that log-odd to probability again giving us the final output.