www.educba.com

Table of Contents

Boosting

→ Boosting is an ensemble technique (involves several trees) that starts from a weaker decision & keeps on building the model such that the final prediction is the weighted sum of all the weaker decision makers.

> the weights are assigned based on performance of individual tree.

→ on boosting, ensemble parameters are calculated in "stagewise way" which means that while calculating the subsequent weight, the learning from previous tree is considered as well.

> i.e) the learning of previous tree boosts the learning of next tree and goes on.. — so called Boosting

→ we mostly used decision tree as weak classifier. Any other algorithm can be used as a base, but reason for choosing trees are:-

Pros	cons
1) Robust to outliers	1) Inability to extract a linear comb of features
2) Feature scaling not required	2) High variance leading to small computational power.
3) Handles missing values	
4) can deal with irrelevant input	
5) computational scalability	

→ Boosting minimizes the variance by taking into consideration the result from various trees.

General understanding eg of boosting

① you wanna travel to different place.

→ And you know a friend who is a traveller, so you ask him about the place (Given your info) he will tell what he know & he will ask his friends.

→ He asks his friend who visited there to give some phone no or any agency.

→ And that friend asks another friend who lives there.

> First, we have to boost other person with info we have, and he will boost next person with info he have & finally get result.

② Dad wants to buy car.

→ He asks you & without any knowledge you say 'yes'. (less weightage)

→ Grandpa thinks and says what model should be bought consider grandchildren who wants car. (good weightage)

→ consider son & grandpa, mom says what to buy, new or second hand, based on financial status (Better weightage)

(6) Ada Boost (Adaptive Boosting)

→ Assume that the no. of training samples are "N", and the no. of iterations (trees created) is "M". Note that possible class outputs are $Y = \{-1, 1\}$.

① Always, total weightage should be equal "1". So Initially consider observation weights as $w_i = \frac{1}{N}$

② For $m = 1$ to M,

→ Fit a classifier $G_m(x)$ to the training data using w_i

→ compute $err_m = \sum_{i=1}^{N} w_i \cdot I(y_i \neq G_m(x_i))$ ← Prediction

→ compute $\alpha_m = \frac{1}{2} \log \frac{(1-err_m)}{err_m}$ — this is the contribution of that tree to Final result

→ calculate new weights using the formula.

$$w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$$

③ Normalize the new sample weights, so that their sum is "1".

④ construct the new tree using the new weights,

i.e; consider weights obtained in 1st tree as the weights for the second tree and, weights of 2nd tree to the 3rd and so on.

⑤ At end, compare summation of results from all trees and Final result is either the one with highest sum (for Regr) or it is the class with most weightage (for classification).

Eg:-

① understandy Gradient Adaboost by

Simple dataset for heart patient prediction

Eg:

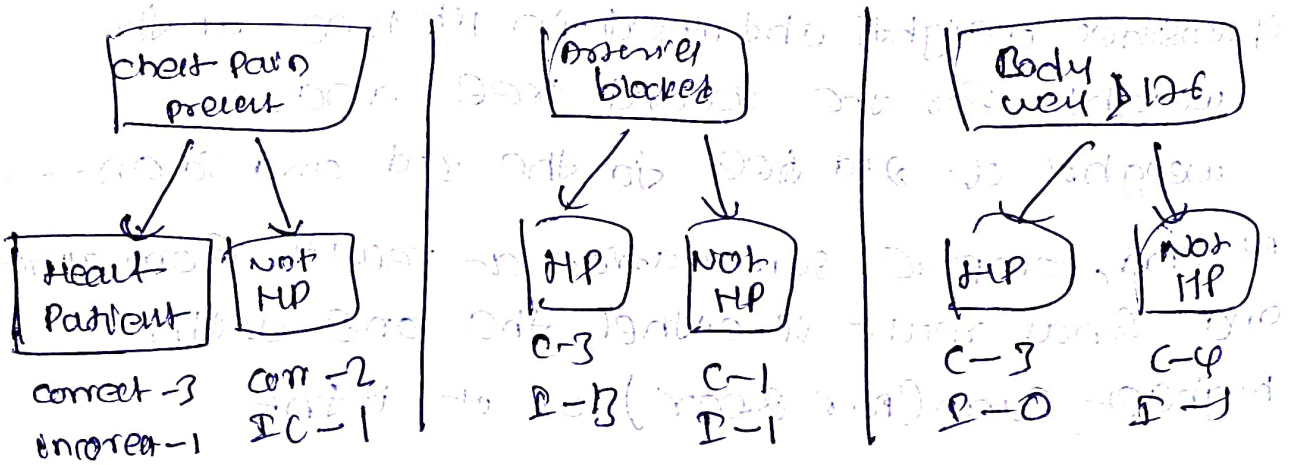
	is chest pain	Arteries blocked	weight of person	is Heart patient	new weight
0	yes	yes	205	yes	0.05
1	no	yes	180	yes	0.05
2	yes	no	210	yes	0.05
3	yes	yes	167	yes	0.33
4	no	yes	156	no	0.05
5	no	yes	125	no	0.05
6	yes	no	160	no	0.05
7	yes	yes	172	no	0.05

Step① - Initialize weights

→ there are 8 rows in our dataset.

∴ $w = 1/N = 1/8$ (samples are equally important)

Step② - consider column to create a weak decision maker and then try to figure out what are correct & incorrect predictions based on that column



→ we will calculat "Gini index" of the column.

> And we select tree with the lowest GI. & this will be the first decision maker for our model.

now, we

step ③ – Now, we calculate the contribution of tree (columns) to our final decision

→ we got weight as first decision maker & we have one incorrectly predicted by it

error = 1/8   contribut = $\frac{1}{2} \log\left(\frac{1-err}{err}\right) = 0.97$.

→ step ④ – modify the weights

→ Increase the sample weight for incorrectly predicted, new weight = old wed·e^σm

$w_{i} = 1/8 \times e^{0.97} = 0.33$

→ Decrease the sample weight for correctly predicted, new weight = old wei × e^(−σm)

$w_{i} = \frac{1}{8} \times e^{(-0.97)} = 0.05$

And add weights & divide by summation to normalize it.

> there, new normalized weights will act as sample weights for the next iteration.

Step ⑤

→ And we create new tree which consider dataset prepared with new sample weight.

→ Suppose, m tree classify a person as HP & n tree classify a person as Not HP.

then contribution of all m tree & all n-tree are added.

Weighted is taken

⑦

* Gradient Boosting tree

→ Gradient Boosted tree use decision tree as estimator. It can work with different loss function, evaluate its gradient & approximate with simple tree

Adaboost is a special case of Gradient boosting where it use exponential loss function.

Algorithm

① calculate the avg. of the label column as initially this avg shall minimize the total error

② calculate pseudo residuals.

pseudo residual = actual label − predicted result (avg)

Mathematically,

(*) Derivative or Pseudo residual = $\frac{\partial L(y_i, F(x_i))}{\partial (F(x_i))}$

→ Here, gradient or error term is getting calculated as the goal is to minimize the error. Hence, name is gradient boosted tree.

(3) create a tree to predict the pseudo residuals instead of a tree to predict for actual col. value.

new result = (previous result) + (learning rate × residual)

$F_1(x) = F_0(x) + \nu \cdot \gamma$

ν - learning rate
γ - residual.

(4) Repeat these steps until residual stops decreasing.

Eg: to understand GBM.

Q) we need to develop a model to predict an apartment rent based on Sq. footage data.

Sq feet	Rent
750	1160
800	1200
850	1280
900	1450
950	2000

Sol: Step① - set initial model

Lets set avg rent prices as our model : $F_0(x) =$ 1410

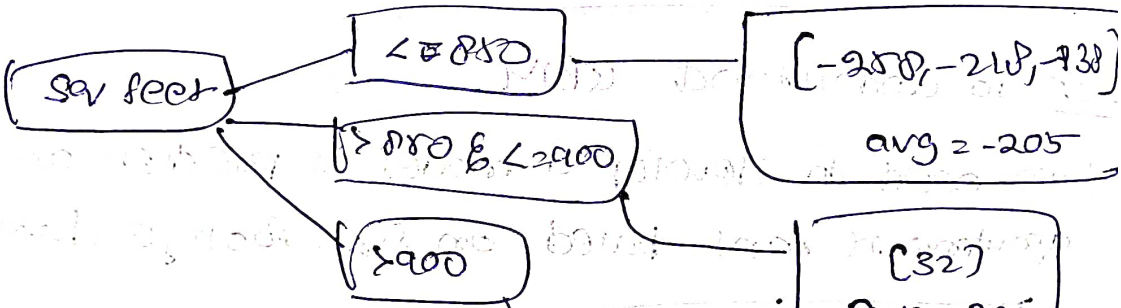
Step ① — calculate the residuals

→ for every instance, we compute the residuals as diff b/w actual & predicted val (avg)

Sq feet	Actual rent	Predicted rent (avg)	Residual
250	1160	1418	-258
800	1200	1418	-218
850	1280	1418	-138
900	1450	1418	32
950	2000	1418	582

Step ③ — Build a model on residuals, instead on actual col. values

we build a decision tree to predict residual



Step ④ — updating predicted value by combing all trees

F(m) updated Predicted val = Fo(x) + (learning rate × Residual predicted)

Fo(x)	Residual Predicted	Learning rate	F(m)
1418	-205	1	1213
1418	-205	1	1213
1418	-205	1	1213
1418	32	1	1450
1418	582	1	2000

step 5 — Repeat step 2-4, for the predefined no. of iterations.

step 6 — Finally, the composite model sums together all of the weak models into one strong predictor.

GBM vs AdaBoost

→ Essentially both Adaboost & GBM have similar intuition, i.e. to convert a set of weak learner into a single strong learner.

(*) However, the only point of difference is, "How they create a weak learner during process?"

→ Adaboost, changes the weights assigned to each of the instance, such that every subsequent learner focuses more on the difficult ones and therefore, contributes to creating a strong learner.

→ Gradient boosting, trains the weak learner on errors. Each iteration computes the residuals & fits the next learner. Finally, using an optimization process, the algorithm computes the contribution of each weak learner which minimizes the overall error of the model.