

BOOSTING

Boosting
Steps in
general

BOOSTING — This is another ensemble method that aims to reduce the bias and variance by sequentially training B number of models. Each new model tries to correct the errors made by the previous model.

STEP 1: Initialization - Start with an initial prediction

$$\hat{y}^{(0)} = \text{Mean}(y)$$

STEP 2: Iterative training - For each iteration 1 to B .

↳ Compute residuals

$$g_r^{(b,i)} = y^{(i)} - \hat{y}^{(b-1,i)}$$

↳ Fit a model

Train a new model $h^{(b)}$ to predict the residuals $g_r^{(b)}$

↳ Update the predictions

$$\hat{y}^{(b,i)} = \hat{y}^{(b-1,i)} + \alpha h^{(b)}(x^{(i)})$$

where, α is the learning rate

STEP 3: Final Prediction

$$\hat{y}^{(M,i)} = \hat{y}^{(0)} + \alpha \sum_{b=1}^B h^{(b)}(x^{(i)})$$

GRADIENT BOOSTING ALGORITHM (one type of boosting)

The gradient boosting algorithm is an ensemble machine learning algorithm that builds a strong predictive model sequentially using several weak learners.

PRELIMINARIES:

(2)

Gradient Descent Optimization : The gradient descent optimization is an optimization algorithm to find the minimum of a loss function by moving in the opposite direction of gradient at the current point of evaluation.

$$\Theta_j := \Theta_j - \alpha \frac{\partial J}{\partial \Theta_j}$$

$$\text{OR } \Theta_{j+1} = \Theta_j - \alpha \frac{\partial J}{\partial \Theta_j}$$

In the above equation Θ_{j+1} is the value of the parameters at the $(j+1)^{\text{th}}$ iteration, Θ_j is the value of the parameters set at the j^{th} iteration. α is the learning rate. J is the loss function. $\frac{\partial J}{\partial \Theta_j}$ is the gradient of the loss function w.r.t the parameter Θ_j .

* This equation states that the value of Θ will improve over its previous values after a given number of iterations.

INTUITION: What if instead of Θ we have the errors or residuals made by our machine learning algorithm. The residuals will improve over given number of iterations.

Where is the gradient and what are we boosting?

Expressing the residual as the gradient of the loss function helps to incorporate other loss functions as well. Also, because we build the ensemble model sequentially, therefore essentially we are boosting.

② Consider the loss function in a regression case. One of the widely used loss function in a regression setting is the mean squared loss or the MSE.

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

Understanding the Gradient Boosting Algorithm

→ Take the derivative of the loss function w.r.t. \hat{y} . \hat{y} is nothing but the predictions made by the ML model

$$\begin{aligned} \therefore \frac{\partial}{\partial \hat{y}} L(y, \hat{y}) &= \frac{\partial}{\partial \hat{y}} \left[\frac{1}{2} (y - \hat{y})^2 \right] \\ &= \frac{1}{2} \cdot -2(y - \hat{y}) \cdot (0 - 1) \\ &= (\hat{y} - y) = \text{or} \leftarrow \text{Residual} \end{aligned}$$

From this, we see that the gradient of the loss function w.r.t. the predictions is equal to the residual.

→ Plugging in the expression for the gradient of the loss function and the predictions in the gradient descent optimization update equation. We get:

$$\begin{aligned} \hat{y}_{(j+1)} &= \hat{y}_j + \alpha \frac{\partial L}{\partial \hat{y}_j} \\ \Rightarrow \hat{y}_{(j+1)} &= \hat{y}_{(j)} + \alpha \text{or} \end{aligned}$$

Now, imagine if we have a model that is trained to predict the residuals, then instead of or (residual) we can simply put the model.

Mathematically, we can say,

$$y_{(j+1)} = y_{(j)} + \alpha h_{(j)}(x)$$

where, $h_{(j)}$ is j th model trained to predict the residuals from the feature x .

ILLUSTRATION:

Consider a dataset D.

X	y
—	—
—	—
—	—
:	:
:	:
—	—

\hat{y}	$y - \hat{y} = e_1$
—	—
—	—
—	—
:	:
:	:
—	—

NOTE:

* You do not fit a model between X and y rather between X and e_1

You train a model to predict the residuals, e_1 , from the features, X . Let us call that model $h(x)$. But, to calculate the first residual you need the first \hat{y} . This initial \hat{y} can be taken as the mean of the target values; we can write those as

$$\hat{y}_{(0)} = \frac{1}{M} \sum_{m=1}^M y_m$$

Once you have this $\hat{y}_{(0)}$, you can use this calculate the first residual $e_{(0)}$. Once you have this $e_{(0)}$, you can train your first ML model such that

$$h_{(0)}(x) = e_{(0)}$$

Also, model predictions after the first iteration will be

$$\hat{y}_{(1)} = \hat{y}_{(0)} + \alpha h_{(0)}(x)$$

In the next iteration you train your next model that aims to further correct the errors made by your first model.

Mathematically we can write this as:

$$h_{(1)}(x) = g_{(1)}$$

Model prediction after second model training will be:

$$\hat{y}_{(2)} = \hat{y}_{(1)} - \alpha h_{(1)}(x)$$

$$\text{OR } \hat{y}_{(2)} = \hat{y}_{(0)} + \alpha h_{(0)}(x) + \alpha h_{(1)}(x)$$

$$\text{OR} \quad \hat{y}_{(2)} = \hat{y}_{(0)} + \alpha (h_{(0)}(x) + h_{(1)}(x))$$

$$\text{OR } \hat{y}^{(n)} = \hat{y}^{(0)} + \alpha(h_{(0)}(x) + h_1(x) + \dots + h_{(n-1)}(x))$$

Writing compactly.

$$\hat{y}(B) = \hat{y}_0 + \sum_{b=1}^B h_{1b}(x)$$

Learning Rate

NOTE: $0 < \alpha < 1$; α value near to zero brings gradual updates/improvements.
 α value near to one brings aggressive improvements.
 α value greater than one is impractical
 α value less than zero is counter-productive

B is a parameter you can play with to improve your boosting algorithm.