Ada Boost

1. Ada boost starts with building a very small tree(level=1) a stump from the training data.
2. The amount of say that the stump has on the final output is based on how well it compensates for those previous errors.
3. Then Ada Boost builds the next stump based on the errors that the previous stump made.
4. It keeps on building the stump on the errors that the previous stump made until it has made the number of stumps you asked for , or it has a perfect fit.

Gradient Boost for Regression

1. Loss function

$$Loss = \frac{1}{2}(observed - predicted)^2$$

1. We start with a leaf that is the average value of the variable we want to predict.
2. Then we add a tree(larger than the stump in practice leaf could be between 8 and 32) based on the residuals(difference between the Observed values and the Predicted values.)
3. Calculate the output values of the leaves of the fitted tree.The output value is the average of the value of the records present in that leaf.
4. For each record,make the final prediction by adding the initial prediction and adding the output of the tree,scaled by a Learning rate.. Unlike in ada boost where all trees are scales by the same amount.
5. Then we add another tree based on the new residuals and we keep adding new trees based on the errors made by the previous tree.

Gradient Boost for Classification

1. We start with a leaf that represents the initial prediction for every individual record using log(odds). For eg. In binary classification with 60 Yes and 40 No.

$$log(odds) = 60/40 = 1.5$$

2. We convert log(odds) to probability using Logistic function to use it for classification.

$$P(\text{Yes}) = \frac{e^{log(odds)}}{1 + e^{log(odds)}}$$

3. For each record,calculate the pseudo residual ,the difference between the Observed and the predicted.
4. Then we add a tree(larger than the stump in practice leaf could be between 8 and 32) based on the residuals(difference between the Observed values and the Predicted values.)
5. For each leaf node in the tree,calculate the output values of the leaf of the fitted tree. As the output of the leaf is represented as probability , we need to do a transformation to make it

log(odds) comparable with the previous predictions for residual calculation.

$$T = \frac{\sum Residuals_i}{\sum[\text{Previous Probability}_i * (1 - \text{Previous Probability}_i)]} \text{where i is the records in the leaf node}$$

3. For each record,make the final prediction by adding the initial prediction and adding the output of the tree(log(odds),scaled by a Learning rate.
4. For each record,convert the log(odds) prediction into predicted probability.

$$Probablity = \frac{e^{log(odds)}}{1 + e^{log(odds)}}$$

5. For each record,calculate the new residuals,the difference between the Observed and the predicted probability.
6. Then we add another tree based on the new residuals and we keep adding new trees based on the errors made by the previous tree.

XgBoost for Regression

1. Make the initial prediction,by default it is 0.5 or we can have any value as initial prediction.
2. For each record,calculate the residual(observed - initial prediction).
3. Then we add a tree,here the tree is a little different from the general Regression trees which is built in Gradient Boosting and hence lets call it a XgBoost Tree.
4. There are many ways to create Xgbosst tree,we will look at the most common way.
5. Each tree starts out as a single leaf and all the residuals go to the leaf.
6. Calculate the Quality Score or the Similarity Score for the Residuals in the leaf.

$$\text{Similarity Score} = \frac{\sum_i Residuals_i}{\text{Number of Residuals} + \lambda} \text{where } \lambda \text{ is a regularization parameter}$$

7. For each of the possible thresholds for each feature,calculate the Gain of splitting the residuals into two groups.

$$Gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$

8. Pick the split where the Gain is the maximum.
9. Continue adding the leaves until a criterion is met. Default is to allow up to 6 levels.
10. Prune the tree based on the Gain values.If the difference between the Gain and gamma is negative we will remove the branch.
11. When lambda >0 ,it is easier to prune leaves because the values for Gain are smaller and difference between Gain and gamma is more likely to be negative. It will prevent in over fitting the Training Data.
12. Note-Setting lambda to 0 ,doesn't to turn off pruning as any branch where Gain is negative would be a candidate for pruning.
13. For each leaf node in the tree,calculate the output values of the leaf as:

$$Output = \frac{\text{Sum of Residuals}}{\text{Number of Residuals} + \lambda}$$

14. Make the final prediction by adding the initial prediction and adding the output of the tree,scaled by a Learning rate called eta.
15. Continue adding new tree based on the new resdiuals and make new predictions that gives us even samller residuals.

Reference

Gradient-Boosting

XgBoost