Introduction

XGboost is the most widely used algorithm in machine learning, whether the problem is a classification or a regression problem. It is known for its good performance as compared to all other machine learning algorithms.

Even when it comes to machine learning competitions and hackathon, XGBoost is one of the excellent algorithms that is picked initially for structured data. It has proved its determination in terms of speed and performance.

> "It is the execution of gradient boosted decision trees that is designed for high speed and performance."

In this blog, I will introduce you to *the XGBoost algorithm. What is it all about? What is the working behind the algorithm, parameters used by XGBoost and many different features about XGBoost?*

What is XGBoost? Why is it so powerful?

Its is also known as *"Extreme Gradient Boosting"*

> The name XGboost, though, actually refers to the engineering goal to push the limit of computations resources for boosted tree algorithms. Which is the reason why many people use XGboost? — Tianqi Chen

It is called "an enhanced gradient boosting library" that makes use of a gradient boosting framework. Neural Network performs well when it comes to prediction problems that involve unstructured data like images and text.
But, decision tree-based algorithms are considered to be good performers when it comes to small to medium structured data or tabular data. XGboost is commonly

used for supervised learning in machine learning. It was created by *PhD student Tianqi Chen, University of Washington.*

Let's us understand the reason behind the good performance of XGboost -

1. Regularization:

   This is considered to be as a dominant factor of the algorithm. Regularization is a technique that is used to get rid of overfitting of the model.
2. Cross-Validation:

   We use cross-validation by importing the function from sklearn but XGboost is enabled with inbuilt CV function.
3. Missing Value:

   It is designed in such a way that it can handle missing values. It finds out the trends in the missing values and apprehends them.
4. Flexibility:

   It gives the support to objective functions. They are the function used to evaluate the performance of the model and also it can handle the user-defined validation metrics.
5. Save and load:

   It gives the power to save the data matrix and reload afterwards that saves the resources and time.

What is the working behind XGBoost?

It carries out the gradient boosting decision tree algorithm. It has several different names like gradient boosting, gradient boosting machine, etc.

Boosting is nothing but ensemble techniques where previous model errors are resolved in the new models. These models are added straight until no other improvement is seen. One of the best examples of such an algorithm is the AdaBoost algorithm.

Gradient boosting is a method where the new models are created that computes the error in the previous model and then leftover is added to make the final prediction.

It uses a gradient descent algorithm that is the reason it is called a "Gradient Boosting Algorithm". Weather classification or regression methods are supported for both types of predictive modelling problems.

Check here the talk shared by the creator of the algorithm to the LA Data Science Group that was titled as "XGBoost: A Scalable Tree Boosting System."

XGBoost Parameters Tuning

When it comes to model performance, each parameter plays a vital role. Let us quickly understand what these parameters are and why they are important.

Since there are many different parameters that are present in the documentation we will only see the most commonly used parameters. You can check the documentation to go through different parameters.

XGBoost, as per the creator, parameters are widely divided into three different classifications that are stated below -

1. General Parameter:

The parameter that takes care of the overall functioning of the model.

- Booster[default=gbtree]

  Assign the booster type like gbtree, gblinear or dart to use. Use gbtree or dart for classification problems and for regression, you can use any of them.

- nthread[default=maximum cores available]

  The role of nthread is to activate parallel computation. It is set as maximum only as it leads to fast computation.

- silent[default=0]

  It is better not to change it if you set it to 1, your console will get running messages.

2. Booster Parameter:

A parameter that powers the selected booster performance.

Parameters for Tree Booster

- nrounds[default=100]

  It controls the maximum number of iterations. For classification, it is similar to the number of trees to grow. Should be tuned using CV

- eta[default=0.3][range: (0,1)]

It commands the learning rate i.e the rate at which the model learns from the data. The computation will be slow if the value of eta is small. Its value is between 0.01-0.03.

- gamma[default=0][range: (0,Inf)]
  Its function is to take care of the overfitting. Its value is dependent on the data. The regularization will be high if the value of gamma is high.

- max_depth[default=6][range: (0,Inf)]
  Its function is to control the depth of the tree, if the value is high, the model would be more complex. There is no fixed value of max_depth. The value depends upon the size of data. It should be tuned using CV.

- subsample[default=1][range: (0,1)]
  Its values lie between (0.5-0.8) and it controls the samples given to the tree.

- colsample_bytree[default=1][range: (0,1)]
  It checks about the features supplied to the tree.

- lambda[default=0]
  It is used to avoid overfitting and controls L2 regularisation.

- alpha[default=1]
  Enabling alpha, it results in feature selection by which it is more useful for high dimension dataset. It controls L1 regularization on weights.

<u>Parameters for Linear Booster</u>

Its computation is high as it has relatively fewer parameters to tune.

- nrounds[default=100]

  It powers the iteration that is required by gradient descent to converge. It should be tuned using CV.

- lambda[default=0]

  Its function is to permit Ridge Regression.

- alpha[default=1]

  Its function is to permit Lasso Regression.

3. Learning Tak Parameters:

A parameter that validates the learning process of the booster.

- Objective[default=reg:linear]
  - Reg: linear - It is used for linear regression.
  - Binary: logistic - It is used for logistic regression for binary classification that returns the class probabilities.
  - Multi: softmax - It is used for multi-classification using softmax that returns predicted class labels.
  - Multi: softprob - It is used for multi-classification using softmax that returns predicted class probabilities.

  (To learn more about similar activation functions, read here)

- eval_metric [no default, depends on objective selected]

  These metrics are used to validate a model's capability to generalize. For the classification type of problem, the default is an error and for regression, the default metric is RMSE.

Error functions are listed below:

- mae - Used in regression.
- Log loss - Negative log-likelihood that is used in classification.
- AUC - Area under curve used in classification
- RMSE - Root mean square error used in regression
- Error - Binary classification error rate.
- mlogloss - multiclass log loss used for classification again.

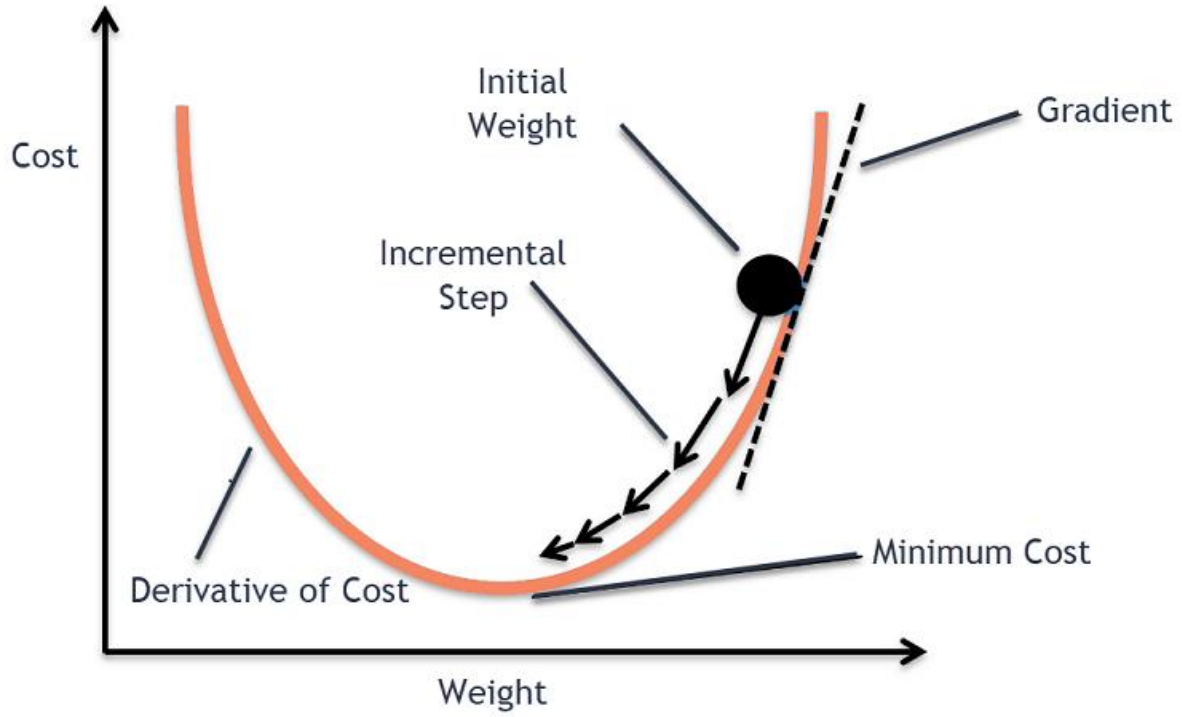
Conclusion

Machine Learning is a vigorous research area and there are already different feasible choices to XGBoost. Microsoft Research Area has released a LightGBM framework for gradient boosting that again shows good performance.

Gradient descent is an iterative optimization algorithm for finding the local minimum of a function.

To find the local minimum of a function using gradient descent, we must take steps proportional to the negative of the gradient (move away from the gradient) of the function at the current point. If we take steps proportional to the positive of the gradient (moving towards the gradient), we will approach a local maximum of the function, and the procedure is called Gradient Ascent.

Gradient descent was originally proposed by CAUCHY in 1847. It is also known as steepest descent.



Source: Clairvoyant

The goal of the gradient descent algorithm is to minimize the given function (say cost function). To achieve this goal, it performs two steps iteratively:

1. Compute the gradient (slope), the first order derivative of the function at that point
2. Make a step (move) in the direction opposite to the gradient, opposite direction of slope increase from the current point by alpha times the gradient at that point

Gradient descent algorithm

$$\text{repeat until convergence } \{$$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$(\text{for } j = 1 \text{ and } j = 0)$$

$$\}$$