XGBoost : A Scalable Tree Boosting System

IES-601 Seminar

Name : Shivam Tiwari

Roll Number : 23N0463

Contents:

Introduction:

- Machine learning and data-driven approaches are becoming very important in many areas. Smart spam classifiers protect our email by learning from massive amounts of spam data and user feedback; advertising systems learn to match the right ads with the right context; fraud detection systems protect banks from malicious attackers
- Tree boosting is a highly effective and widely used machine learning method based on ensembling methods, One method which we widely uses is Gradient boosting (GBM) and this is one technique that shines in many applications but we faces challenges while using this.
- To resolve many problem we will discuss our approach of machine learning method XGBoost ,this is a scalable machine learning system for tree boosting and power of this method we can predict by the kaggle competitions, In 29 competition the 15 winners used this method.

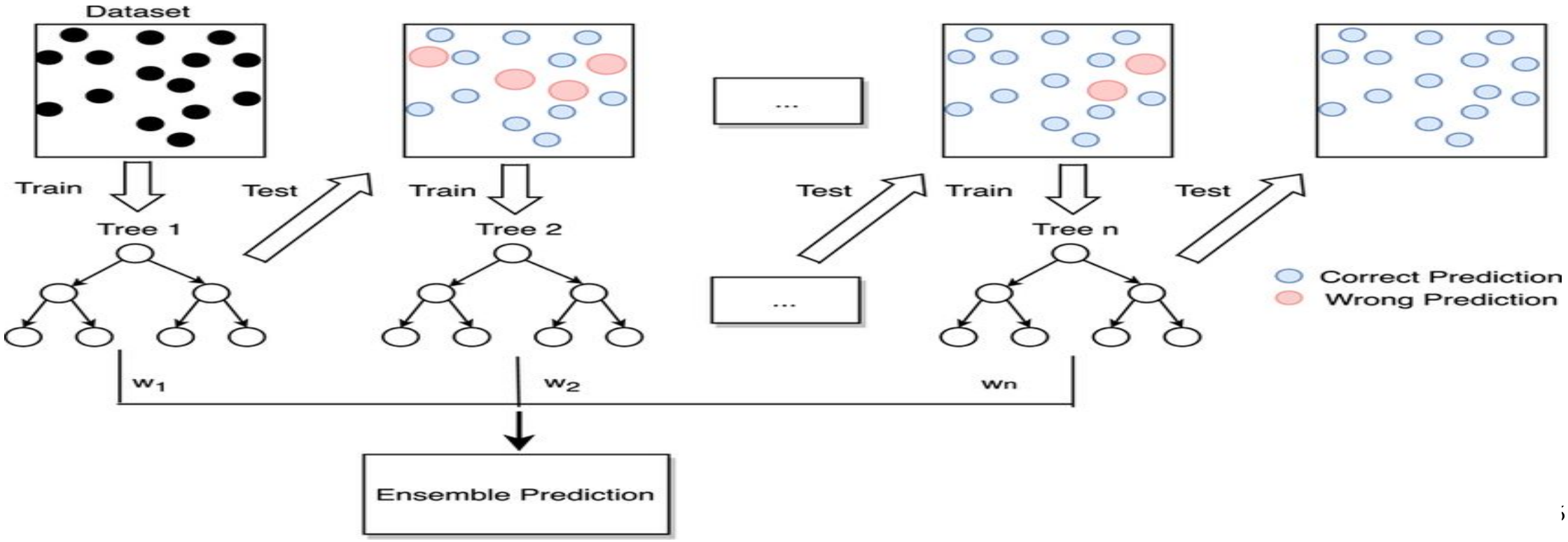
History :

- Machine learning :
  - In late 1980 linear regression and Naive Bayes algorithm developed
  - In 1990 major changes occurred , Many new algorithms are evolved like:
    - Random Forest
    - SVM
    - Gradient Boosting

- Development of XGBoost:
  - XGBoost was first introduced by Tianqi Chen in 2014 as part of the Distributed (Deep) Machine Learning Community (DMLC) group.
  - The name *XGBoost* is short for ***Ext**reme **G**radient **Boost**ing*.
  - It quickly gained popularity in the data science community due to its exceptional performance compared to other tree-based models.

Boosting:

- **Boosting** is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data.
- The second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.



Why XGBoost ?

- XGBoost is a powerful machine learning algorithm known for its efficiency and effectiveness in handling structured/tabular data.
- Some Reasons Why we need XGBoost:
  - High Accuracy: XGBoost often delivers more precise predictions.
  - Speedy Processing: It quickly handles large datasets
  - Versatility: Suitable for various prediction tasks.
  - Prevents Overfitting: Avoids getting too detailed to make more reliable predictions.
  - Identifies Important Factors: Highlights what factors matter most for predictions.
  - Works for Different Problems: Whether you're trying to predict categories, numbers, or rankings, XGBoost can handle lots of different types of problems.

Development of XGBoost:

Modification which leads to make XGBoost different from others:

- Design and build a highly scalable **end-to-end tree boosting** system.
- A theoretically justified **weighted quantile** sketch for efficient proposal calculation.
- Introduce a novel sparsity-aware algorithm for **parallel tree learning.**
- An effective **cache-aware block** structure for out-of-core tree learning.

Performance:

★ **Regularized Learning Objective:**

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

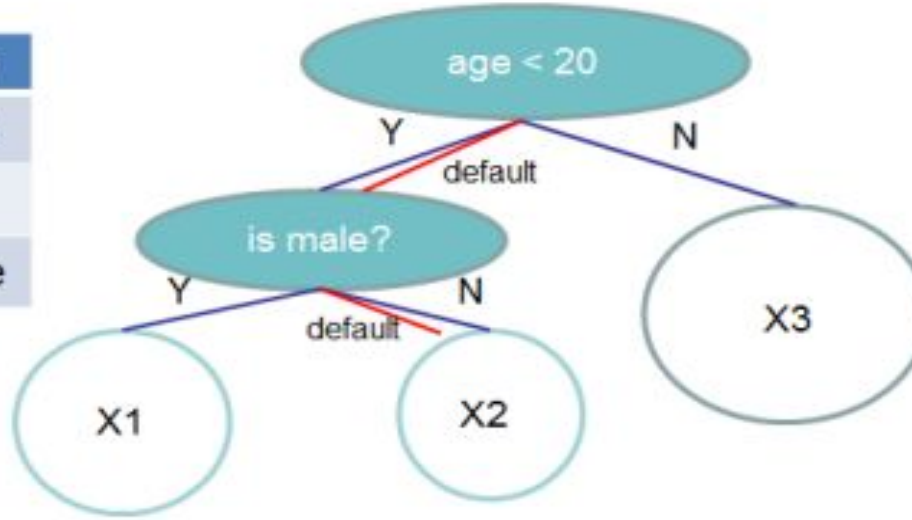
- Adding regularised term in objective function such that these w (weight of leaf nodes) and T (number of leaf nodes) we get in such a way to reduce the value of objective function.
- **Sparsity Aware Split Finding:** Having missing values or zeroes in data set then other other algorithm did not work but xg boost works smartly to handle the**se missing values.**
- Use different types of permutation to handle them and fix this issue , making different trees for values which are missing and try to fill out.

Performance:

- Filling of missing values

Data

Example	Age	Gender
X1	?	male
X2	15	?
X3	25	female



- Approximate Tree learning:

  Making quantile of a column and creating bins on the basis of quantiles as quantile are larger then, small bin width and if quantiles are small then bin's width will be large , by this better splits will be find out due to getting more close to distribution of data set.

Speed:

- **Parallel Processing:**
  - Boosting is essentially a sequential process so parallel is not in sense of model building but we use to build trees parallel to built models which we use in sequentially process.
- **Optimized Data Structures:**
  - **Column Block:** Xg boost uses column blocks to store the data that is instead of storing data in row wise we store it in column wise manner by which it increase the speed of splitting by taking these column blocks.
- **Cache Awareness:**
  - XgBoost uses CPU cache memory to make decision trees effectively for large datasets , Store the node value of trees on which our tree is distributed we store these values in form of cache.

Speed:

- **Out of Core Computing:**
  - Making the Chunks of data sets and then we handle the data set and training the models, this factor uses cache memory

- **Distributed Computing:**
  - First we take partition of data set and then we locally train them and finding all the node , from all nodes we take a master node on that basis we decide our splitting criteria

- **GPU Support:**
  - Due to more parallel working the CPU get slow down therefore they uses GPU to increase the speed for these all parallel processing.

Comparison :

System	exact greedy	approximate global	approximate local	out-of-core	sparsity aware	parallel
XGBoost	yes	yes	yes	yes	yes	yes
pGBRT	no	no	yes	no	no	yes
Spark MLLib	no	yes	no	no	partially	yes
H2O	no	yes	no	no	partially	yes
scikit-learn	yes	no	no	no	no	no
R GBM	yes	no	no	no	partially	no

- By doing updates we can easily compare our Xgboost is now available to do all of the major task in which other algorithms fails

Experiment and Results:

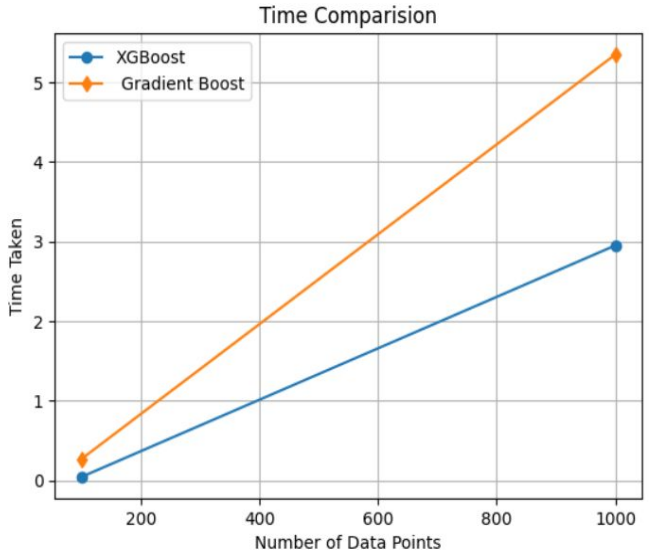
We are comparing XGBoost with Gradient Boosting.

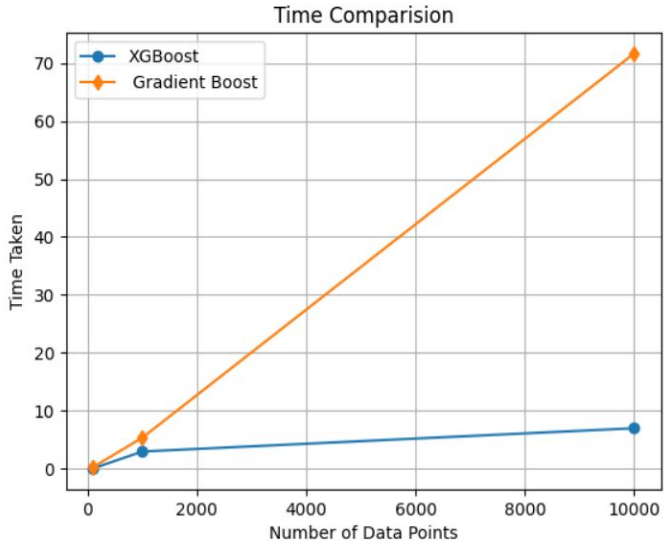
1. Speed :

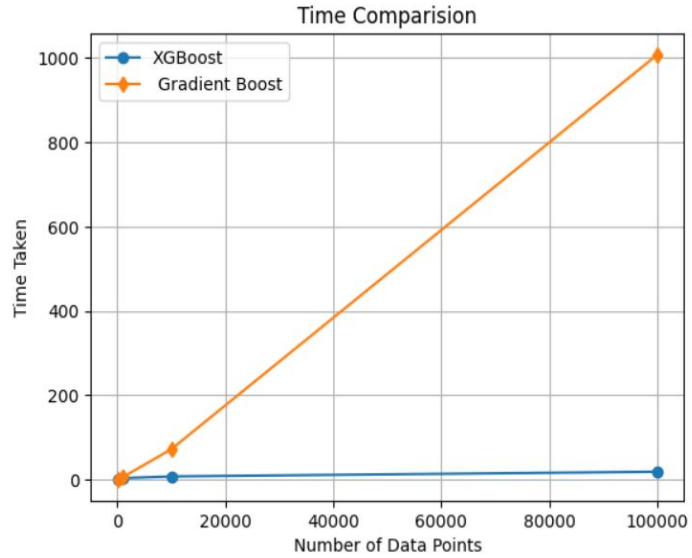
Sr No	Data Points	Time XGBoost	Time Gradient Boost
1	100	0.04190325736999512	0.2668020725250244
2	1000	2.948838949203491	5.3467936515808105
3	10000	6.971016883850098	71.6635754108429
4	100000	18.13682198524475	1007.5712835788727

- Clearly from table we see that as number of Data points increases gradient boosting time increases exponentially whereas XGBoost work very efficiently with great speed.

Results:







➢ From graph we clearly observe that for large data gradient boosting which is one of the best algorithm of machine learning takes so much time but this XGBoost system able to reduce the time very effectively.

Experiment and Results:

2. Performance:

Sr No	Data Points	Accuracy XGBoost	Accuracy Gradient Boost
1	100	0.95	0.8
2	1000	0.875	0.85
3	10000	0.935	0.9325
4	100000	0.8668	0.8735

As we see speed of Xgboost is brilliant but its performance is also better than the Gradient Boosting as we can see most of the time XGBoost is better.

References:

1.Paper Link: 1603.02754 (arxiv.org)

- Authors: Tianqi Chen, University of Washington
  Carlos Guestrin, University of Washington

2. XGBoost Documentation:

XGBoost Documentation — xgboost 2.0.3 documentation

3.Wikipedia:

XGBoost - Wikipedia