Gradient Boosting XGBoost

Francesco Pugliese, PhD

neural1977@gmail.com

We will discuss about…

- ✓ Gradient Boosting
- ✓ XGBoost

Boosting

- ✓ In Machine Learning, the **Boosting Technique** is an **ensemble meta-algorithm** for primarily reducing **bias** and **variance** in Supervised Learning.

- ✓ **Bias** in Machine Learning: it is a systematic error occurring due to incorrect assumptions in the Machine Learning process (i.e. biased training set, etc.). Basically, the bias represents the error between average model prediction and the ground truth and describes how well the model matches the training data set. Higher biases correspond to these situations: a) Failure to capture proper data trends; b) Underfitting; c) High error rate.

- ✓ **Variance** refers to the huge variations of the model when using different portions of the training data set. In other words, variance is the variability of the model prediction. Generally, variance arise in highly complex models with a large number of features:

    1. Models with high bias will have low variance.
    2. Models with high variance will have a low bias.

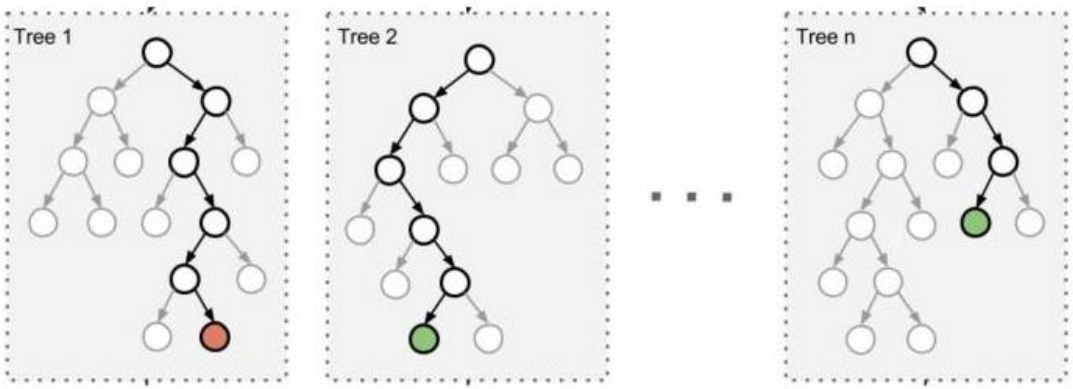
- ✓ All these contribute to the **flexibility** of the model. For instance, a model that does not match a data set with a high bias will create an inflexible model with a low variance that results in a suboptimal machine learning model. Characteristics of high variance: a) Noise in the data set, Overfitting, Complex models

Gradient boosting

- ✓ **Meta-learning** refers to machine learning algorithms that learn from the output of other machine learning algorithms. Meta-learning algorithms typically refer to ensemble learning algorithms like stacking that learn how to combine the predictions from ensemble members.

- ✓ **Boosting** is based on the question posed by **Kearns** and **Valiant**: "Can a set of weak learners create a single strong learner?" A weak learner is defined to be a classifier that is only slightly correlated with the true classification (it can label examples better than random guessing). In contrast, a strong learner is a classifier that is arbitrarily well-correlated with the true classification.

- ✓ **Gradient Tree Boosting** or **Gradient Boosted Decision Trees (GBDT)** is a generalization of boosting to arbitrary differentiable loss functions. **GBDT** is an accurate and effective off-the-shelf procedure that can be used for both regression and classification problems in a variety of areas including Web search ranking and



Gradient boosting

- ✓ **GBM** is a boosting algorithm used when we deal with plenty of data to make a prediction with high prediction power.
- ✓ **Boosting** is actually an ensemble of learning algorithms which combines the prediction of several base estimators in order to improve robustness over a single estimator.
- ✓ It combines multiple **weak** or average predictors to a build **strong** predictor.
- ✓ These boosting algorithms always work well in data science competitions like **Kaggle**, AV Hackathon, CrowdAnalytix.
- ✓ They are generally used as **baseline** for the Kaggle competitions, to get the lower extreme of accuracy we can achieve and apply more powerful **Deep Learning** architectures to try to increase the performance.
- ✓ Boosting might lead to over fitting if the data sample is very small

What's the difference between Random Forest and Gradient boosting?

- ✓ Like **Random Forest**, **Gradient boosting** is a set of decision trees.

- ✓ The two main differences are:

    1.How trees are built: random forests builds each tree independently while gradient boosting builds one tree at a time. This **additive model** (ensemble) works in a forward stage-wise manner, introducing a weak learner to improve the shortcomings of existing weak learners.

    2.Combining results: random forests combine results at the end of the process (by averaging or "majority rules") while gradient boosting combines results along the way.

- ✓ If you carefully tune parameters, gradient boosting can result in better performance than random forests. However, gradient boosting may not be a good choice if you have a lot of **noise**, as it can result in **overfitting**. They also tend to be harder to tune than random forests.

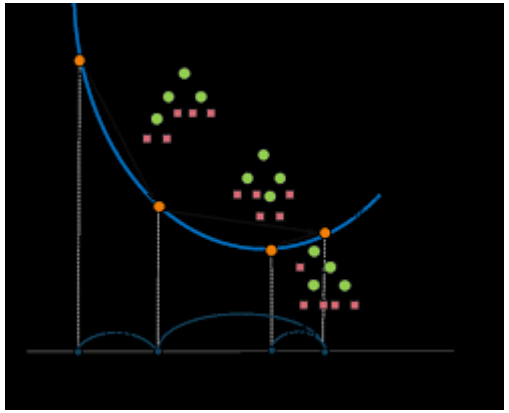
- ✓ Random forests and gradient boosting each excel in different areas. Random forests perform well for multi-class object detection and bioinformatics, which tends to have a lot of statistical noise. Gradient Boosting performs well when you have unbalanced data such as in real time risk assessment.

Gradient boosting

- ✓ In Gradient Boosting, each predictor tries to improve on its predecessor by reducing the errors. But the most interesting insight of Gradient Boosting is that instead of fitting a predictor on the data at each iteration, it actually fits a new predictor to the **residual errors** made by the previous predictor. Let's go through a step by step example of how **Gradient Boosting Classification.**

- ✓ Gradient boosting involves three elements:

1) **A loss function to be optimized**: The loss function used depends on the type of problem being solved. Regression may use a squared error and classification may use logarithmic loss such as cross-entropy. A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used.



Gradient boosting

2) A weak learner to make predictions: Decision trees are used as the weak learner in gradient boosting. Specifically, regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and "correct" the residuals in the predictions. Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss. Initially, very short decision trees were used that only had a single split, called a decision stump. Larger trees can be used generally with 4-to-8 levels.
It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes. This is to ensure that the learners remain weak, but can still be constructed in a greedy manner.

3) An additive model to add weak learners to minimize the loss function: Trees are added one at a time, and existing trees in the model are not changed. A gradient descent procedure is used to minimize the loss when adding trees. Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error. Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e, follow the gradient). We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by (reducing the residual loss. Generally, this approach is called functional gradient descent or gradient descent with functions.

Steps of Gradient boosting

- ✓ **Step 1** : Assume mean is the prediction of all variables.
- ✓ **Step 2** : Calculate errors of each observation from the mean (latest prediction).
- ✓ **Step 3** : Find the variable that can split the errors perfectly and find the value for the split. This is assumed to be the latest prediction.
- ✓ **Step 4** : Calculate errors of each observation from the mean of both the sides of split (latest prediction).
- ✓ **Step 5** : Repeat the step 3 and 4 till the objective function maximizes/minimizes.
- ✓ **Step 6** : Take a weighted mean of all the classifiers to come up with the final model.

XGBoost

- ✓ **XGBoost** is a **Gradient Boosting** algorithm that uses decision trees as its "weak" predictors. Beyond that, its implementation was specifically engineered for optimal performance and speed. Historically,

- ✓ **XGBoost** has performed quite well for structured, tabular data. If you are dealing with non-structured data such as images, deep neural networks are usually a better option.

- ✓ When using **Gradient Boosting** for regression, the weak learners are regression trees, and each regression tree maps an input data point to one of its leafs that contains a continuous score.

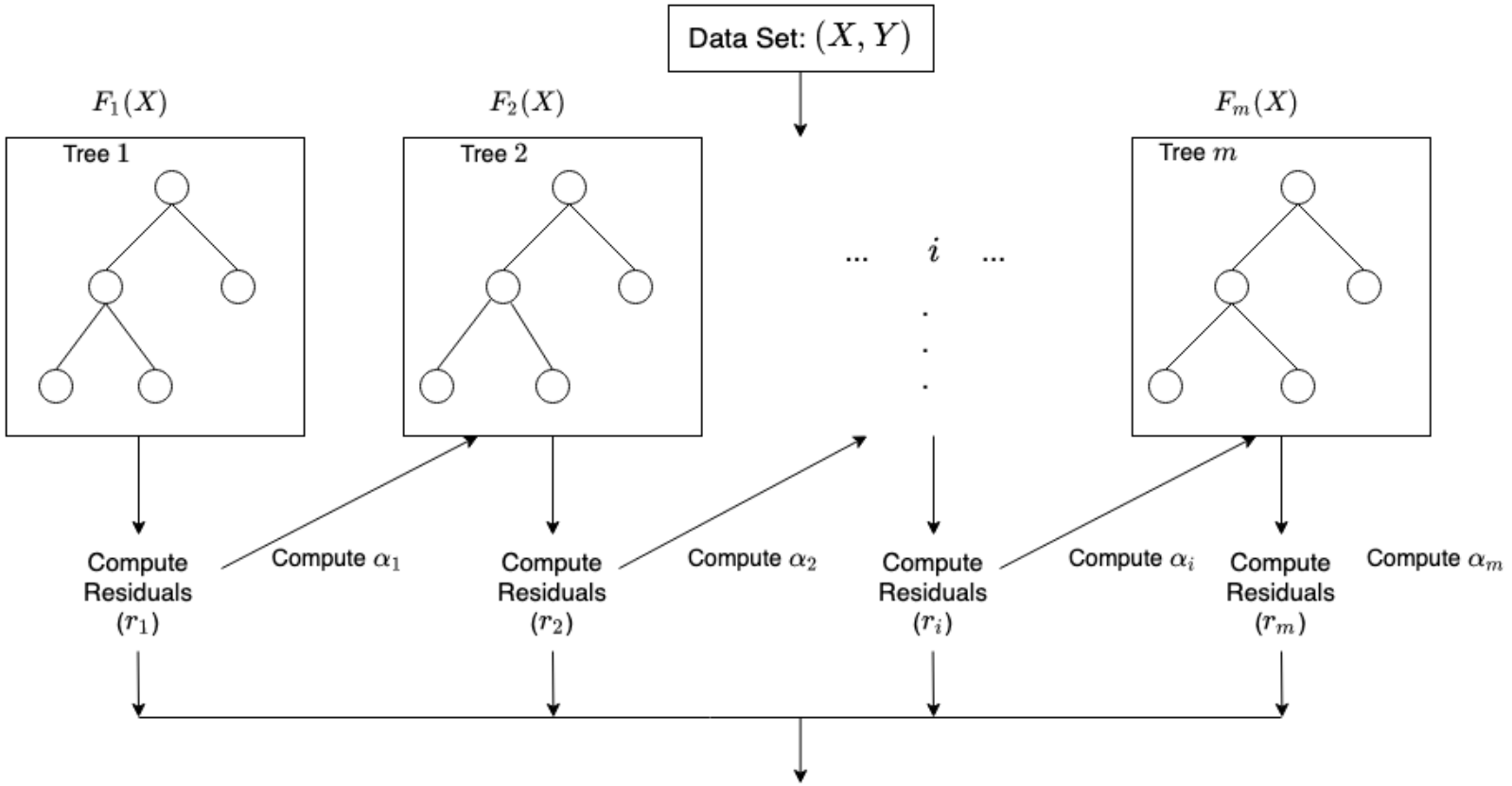
- ✓ **XGBoost** minimizes a regularized (L1 and L2) objective function that combines a convex loss function (based on the difference between the predicted and target outputs) and a penalty term for model complexity (in other words, the regression tree functions).

- ✓ The **training** proceeds iteratively, adding new trees that predict the residuals or errors of prior trees that are then combined with previous trees to make the final prediction. It's called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

XGBoost

- ✓ The XGBoost has an immensely high predictive power which makes it the best choice for accuracy.
- ✓ It possesses both linear model and the tree learning algorithm, making the algorithm almost **10x faster** than existing gradient booster techniques.
- ✓ The support includes various objective functions, including regression, classification and ranking.
- ✓ One of the most interesting things about the **XGBoost** is that it is also called a regularized boosting technique. This helps to reduce overfit modelling.
- ✓ It has a massive support for a range of languages such as Scala, Java, R, Python, Julia and C++.
- ✓ Supports distributed and widespread training on many machines that encompass GCE, AWS, Azure and Yarn clusters.
- ✓ **XGBoost** can also be integrated with Spark, Flink and other cloud dataflow systems with a built in cross validation at each iteration of the boosting process.

XGBoost



$$F_m(X) = F_{m-1}(X) + \alpha_m h_m(X, r_{m-1}),$$

where α_i, and r_i are the regularization parameters and residuals computed with the i^{th} tree respectfully, and h_i is a function that is trained to predict residuals, r_i using X for the i^{th} tree. To compute α_i we use the residuals computed, r_i and compute the following: $arg \min_{\alpha} = \sum_{i=1}^{m} L(Y_i, F_{i-1}(X_i) + \alpha h_i(X_i, r_{i-1}))$ where $L(Y, F(X))$ is a differentiable loss function.

Applications of XGBoost and Gradient Boosting

- ✓ Regression Task.
- ✓ Classification.
- ✓ Ranking.
- ✓ User-defined prediction problems.

References

- ✓ https://towardsdatascience.com/top-10-algorithms-for-machine-learning-beginners-149374935f3c

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., & Chen, K. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, *1*(4), 1-4.

Francesco **Pugliese**