# XGBoost

## What is XGBoost (Extreme Gradient Boosting)?

XGBoost, or Extreme Gradient Boosting, is a state-of-the-art machine learning algorithm renowned for its exceptional predictive performance. It is the gold standard in ensemble learning, especially when it comes to gradient-boosting algorithms. It develops a series of weak learners one after the other to produce a reliable and accurate predictive model. Fundamentally, XGBoost builds a strong predictive model by aggregating the predictions of several weak learners, usually decision trees. It uses a boosting technique to create an extremely accurate ensemble model by having each weak learner after it correct the mistakes of its predecessors.

The optimization method (gradient) minimizes a cost function by repeatedly changing the model's parameters in response to the gradients of the errors. The algorithm also presents the idea of "gradient boosting with decision trees," in which the objective function is reduced by calculating the importance of each decision tree that is added to the ensemble in turn. By adding a regularization term and utilizing a more advanced optimization algorithm, XGBoost goes one step further and improves accuracy and efficiency.

## What Makes XGBoost "eXtreme"?

XGBoost extends traditional gradient boosting by including regularization elements in the objective function, XGBoost improves generalization and prevents overfitting.

## Preventing Overfitting

The learning rate, also known as shrinkage, is a new parameter introduced by XGBoost. It is represented by the symbol "eta." It quantifies each tree's contribution to the total prediction. Because each tree has less of an influence, an optimization process with a lower learning rate is more resilient. By making the model more conservative, regularization terms combined with a low learning rate assist avoid overfitting.

XGBoost constructs trees level by level, assessing whether adding a new node (split) enhances the objective function as a whole at each level. The split is trimmed if not. This level growth along with trimming makes the trees easier to understand and easier to create.

## The algorithm

XGBoost works as Newton-Raphson in function space unlike gradient boosting that works as gradient descent in function space, a second order Taylor approximation is used in the loss function to make the connection to Newton Raphson method.

A generic unregularized XGBoost algorithm is:

Input: training set $\{(x_i, y_i)\}_{i=1}^{N}$, a differentiable loss function $L(y, F(x))$, a number of weak learners $M$ and a learning rate $\alpha$.

Algorithm:

1. Initialize model with a constant value:

   $$\hat{f}_{(0)}(x) = \arg\min_{\theta} \sum_{i=1}^{N} L(y_i, \theta).$$

   [*further explanation needed*]

2. For $m$ = 1 to $M$:
   1. Compute the 'gradients' and 'hessians':[*clarification needed*]

      $$\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}_{(m-1)}(x)}.$$

      $$\hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\right]_{f(x)=\hat{f}_{(m-1)}(x)}.$$

   2. Fit a base learner (or weak learner, e.g. tree) using the training set $\left\{x_i, \frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\right\}_{i=1}^{N}$ [*clarification needed*] by solving the optimization problem below:

      $$\hat{\phi}_m = \arg\min_{\phi \in \mathbf{\Phi}} \sum_{i=1}^{N} \frac{1}{2}\hat{h}_m(x_i)\left[\phi(x_i) - \frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\right]^2.$$

      [*clarification needed*]

      $$\hat{f}_m(x) = \alpha\hat{\phi}_m(x).$$

   3. Update the model:

      $$\hat{f}_{(m)}(x) = \hat{f}_{(m-1)}(x) - \hat{f}_m(x).$$

3. Output $\hat{f}(x) = \hat{f}_{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x).$