Gradient Boosting Machines

Gradient Boosting is a powerful technique to perform supervised machine learning classification. It is an ensemble learner that creates the final model based on a collection of weak predictive models, decision trees in most instances, which are prone to overfitting but their combination in an ensemble and iteratively learning from each of the weak models results in better overall prediction. The key concept of the algorithm is to create new base-learners having a maximum correlation with the negative gradient of the loss function of the entire ensemble. Weaker predictive models in the ensemble are trained in a gradually, additively, and sequentially, and their shortcomings are identified by the use of gradients in the loss function which indicates the acceptability of the model’s coefficients at fitting the underlying data.

XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

XGBoost is an algorithm based on gradient boosting machines, that is optimized for parallel tree boosting with a better performance and the faster execution speed on tabular data.

---------------------

The principle idea behind this algorithm is to construct the new base-learners to be maximally correlated with the negative gradient of the loss function, associated with the whole ensemble. The loss functions applied can be arbitrary, but to give a better intuition, if the error function is the classic squared-error loss, the learning procedure would result in consecutive error-fitting.

---------------------

Gradient Boosting trains many models in a gradual, additive and sequential manner. The major difference between AdaBoost and Gradient Boosting Algorithm is how the two algorithms identify the shortcomings of weak learners (eg. decision trees). While the AdaBoost model identifies the shortcomings by using high weight data points, gradient boosting performs the same by using gradients in the loss function (y=ax+b+e , e needs a special mention as it is the error term). The loss function is a measure indicating how good are model’s coefficients are at fitting the underlying data. A logical understanding of loss function would depend on what we are trying to optimise. For example, if we are trying to predict the sales prices by using a regression, then the loss function would be based off the error between true and predicted house prices. Similarly, if our goal is to classify credit defaults, then the loss function would be a measure of how good our predictive model is at classifying bad loans. One of the biggest motivations of using gradient boosting is that it allows one to optimise a user specified cost function, instead of a loss function that usually offers less control and does not essentially correspond with real world applications.

------------------

Like Random Forest, Gradient Boosting is another technique for performing supervised machine learning tasks, like classification and regression. The implementations of this technique can have different names, most commonly you encounter Gradient Boosting machines (abbreviated GBM) and XGBoost. XGBoost is particularly popular because it has been the winning algorithm in a number of recent Kaggle competitions. Similar to Random Forests, Gradient Boosting is an ensemble learner. This means it will create a final model based on a collection of individual models. The predictive power of these individual models is weak and prone to overfitting but combining many such weak models in an ensemble will lead to an overall much improved result. In Gradient Boosting machines, the most common type of weak model used is decision trees - another parallel to Random Forests.

Boosting builds models from individual so called “weak learners” in an iterative way. In the Random Forests part, I had already discussed the differences between Bagging and Boosting as tree ensemble methods. In boosting, the individual models are not built on completely random subsets of data and features but sequentially by putting more weight on instances with wrong predictions and high errors. The general idea behind this is that instances, which are hard to predict correctly (“difficult” cases) will be focused on during learning, so that the model learns from past mistakes. When we train each ensemble on a subset of the training set, we also call this Stochastic Gradient Boosting, which can help improve generalizability of our model. The gradient is used to minimize a loss function, similar to how Neural Nets utilize gradient descent to optimize (“learn”) weights. In each round of training, the weak learner is built and its predictions are compared to the correct outcome that we expect. The distance between prediction and truth represents the error rate of our model. These errors can now be used to calculate the gradient. The gradient is nothing fancy, it is basically the partial derivative of our loss function - so it describes the steepness of our error function. The gradient can be used to find the direction in which to change the model parameters in order to (maximally) reduce the error in the next round of training by “descending the gradient”. In Neural nets, gradient descent is used to look for the minimum of the loss function, i.e. learning the model parameters (e.g. weights) for which the prediction error is lowest in a single model. In Gradient Boosting we are combining the predictions of multiple models, so we are not optimizing the model parameters directly but the boosted model predictions. Therefore, the gradients will be added to the running training process by fitting the next tree also to these values.