# Gradient Boosting Decision Trees

“In many GBDTs (e.g., XGBoost, LightGBM) building next tree comprises two steps: choosing

the tree structure and setting values in leafs after the tree structure is fixed. To choose the best

tree structure, the algorithm enumerates through different splits, builds trees with these splits, sets

values in the obtained leafs, scores the trees and selects the best split. Leaf values in both phases are

calculated as approximations for gradients or for Newton steps. In CatBoost the second phase is

performed using traditional GBDT scheme and for the first phase we use the modified version.”

Started as a research project in 2014, XGBoost [1] became famous in 2016 when Chen et. al. successfully used it in Kaggle competitions and formally published it. The main contribution of this implementation of GBDT algorithm is an improved algorithm for finding split value for a feature and distributed computing provision. XGBoost offers two methods for finding the best split, one is exact and the other is approximate.

The exact greedy algorithm is the slower one of the two, as it iterates over all the features and sorts the data according to the feature value before the gradient statistics are calculated for an instance. A split is then chosen so that it maximises the information gain. Analysing every possible split makes this algorithm highly precise; however, this also leads to computational overload and slower processing.

On the other hand, approximate algorithm uses the bins of histograms of the feature values instead of sorting to propose possible splits. The best split is then decided based on aggregate statistics of the proposed splits. If the proposal of splits is given only in the beginning during tree construction, the algorithm is faster and the authors call it global variant, wherein same proposals are used for splitting at all levels. The local variant re-proposes the splits after each split. This could potentially lead to improved performance when building deeper trees.

Another performance improvement in XGBoost comes by means of handling missing values in sparse data. The ‘sparsity-aware algorithm’, which gives missing value instances a default direction, has been reported to perform better than other algorithms which are generally designed for dense data as opposed to sparse data.

Finally, an effective cache-aware block structure helps to scale the algorithm for out-of-core tree learning. This facilitates distributed computing and therefore lets the user build trees for data with billions of instances.

LightGBM [2] is another implementation of GBDT which builds on the innovations proposed in XGBoost and other algorithms, while addressing the limitations of these. The authors argue that the existing algorithms were inefficient and slow in big data applications as their computation time was proportional to the number of instances and number of features because they scan all data instances in order to find the best split. The authors introduce two novel ideas for optimisation, namely, Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB).

GOSS handles the size of data vertically; it is based on the premise that the instances with small gradient can be removed from the data as they already have small training error. In order to compensate for the change in the distribution of data, such low gradient instances are randomly sampled and then a constant multiplier is used when computing information gain with them. This helps in giving higher importance to instances with large gradient.

In order to tackle the problem of large number of features, EFB reduces feature count by bundling them together. The authors claim that many features are mutually exclusive in that they generally do not take non-zero values simultaneously; as such, these features can be bundled together. The features to be bundled together are selected such that the histogram of bundled feature is same as that of the individual features.

CatBoost [3] is the most recent in popular implementations of the GBDT algorithm, proposed by Yandex group. The authors contend that the limitations of all existing GBDT implementations is due to a special form of target leakage called prediction shift- a phenomenon caused by the models getting biased towards the training examples because it is based on the targets of all the training examples. This problem affects the overall model as well as categorical features encoded using target statistics methods and is tackled in both cases by ordering of training samples.

Ordering principle is inspired from online learning algorithms, wherein the training samples are obtained sequentially in time. Similarly, CatBoost creates artificial time by creating random permutations of the training samples. Different permutations are required for different steps of gradient boosting to reduce variance. This prevents target leakage during encoding of categorial features because the values of target statistic only rely on the observed history.

The proposed ordered boosting works as follows. The algorithm takes one random permutation of the samples and builds trees such that each tree is built only on as many number of samples as the index number of the that tree. The residuals for next observation is calculated using the previous tree. This makes sure that the residuals are calculated using a model which was not trained on the current observation.

In addition, CatBoost uses oblivious trees as base predictors. This is done in order to simplify the complex models because oblivious trees are simpler, balanced and less prone to overfitting.

# Handling Categorical Features

While CatBoost and LightGBM support categorical features, XGBoost requires pre-processing of the data as it provides support for only numerical features. Therefore, the techniques such as one-hot encoding, target encoding and others are necessary.

LightGBM [4] requires the categorical features to be encoded as integers, and such features need to be specified explicitly using parameter *categorical\_feature*. The categories are split into two subsets. First, the histogram of categorical features are sorted according to its accumulated values (sum\_gradient/sum\_hessian) and then the best split on the sorted histogram is obtained.

In order to handle categorical features, CatBoost [5] provides the option of both one-hot encoding as well as target statistics methods. One-hot encoding is generally used when number of categories in a feature is relatively small. The hyperparameter *one\_hot\_max\_size* can be used to set a value threshold for the number of categories in a feature, above which target statistic method is used. For encoding using target statistics, CatBoost first generates random permutations of the training data in order to facilitate ‘online learning’. For each permutation , a subset of data ={} is taken in the training phase, which ensures that the target statistic is calculated only with the observed history of data. In the testing phase all data is taken. The category value is then calculated as

where p is a prior value and a is the weight of the prior. This is required to minimize the effect of noise from categories which have lower frequency. For regression tasks, the standard technique for calculating prior is to take the average label value in the dataset. For classification task a prior is usually an a priori probability of encountering a positive class.

After first split in a tree, CatBoost combines all categorical features of dataset with the existing features in the tree. This ensures that any strong combination of features is acknowledged. Therefore, a general solution of ordered boosting with ordered Target Statistic is reached, which solves the problem of prediction shift in other gradient boosting tree methods.

# References

[1] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’16). Association for Computing Machinery, New York, NY, USA, 785–794. DOI:https://doi.org/10.1145/2939672.2939785

[2] LightGBM: Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: a highly efficient gradient boosting decision tree. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS’17). Curran Associates Inc., Red Hook, NY, USA, 3149–3157.

[3] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS’18). Curran Associates Inc., Red Hook, NY, USA, 6639–6649.

[4] Optimal Split for Categorical Features, <https://lightgbm.readthedocs.io/en/latest/Features.html#optimal-split-for-categorical-features>

[5] Transforming categorical features to numerical features, <https://catboost.ai/docs/concepts/algorithm-main-stages_cat-to-numberic.html> , Last accessed: 20-6-2020