Mondragon Unibertsitatea

Extreme Gradient Boosting

XGBoost

Xabier Etxezarreta Argarate
7-11-2019

Table of contents

Introduction 2

Internals 3

  Gradient Boosting 3

  Extreme Gradient Boosting 4

How does XGBoost perform so well? 5

  Algorithm optimizations 5

    Parallelization 5

    Hardware Optimization 5

  Algorithm improvements 5

    Regularization 5

    Sparsity Awareness 5

    Weighted Quantile Sketch 5

    Cross-Validation 5

XGBoost framework on python 6

  XGBoost algorithm parameters 6

Bibliography 8

Introduction

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework (github.com, 2019). This algorithm was developed as a research project at the University of Washington. Tianqi Chen and Carlos Guestrin presented their paper (arxiv.org, 2016) at SIGKDD Conference in 2016. Since its introduction, this algorithm has won numerous competitions in Kaggle.

The algorithm differentiates itself in the following ways:

- A wide range of applications:
    - Can be used to solve regression, classification, ranking, and user-defined prediction problems.

- Portability / Multiplatform:
    - Runs smoothly on Windows, Linux, and OS X.

- Languages:
    - Supports all major programming languages including C++, Python, R, Java, Scala, and Julia.

- Cloud Integration:
    - Supports AWS, Azure, and Yarn clusters and works well with Flink, Spark, and other ecosystems.

Internals

The best way to understand the internals of XGBoost is understanding and knowing the differences between Gradient Boosting, Gradient Boosted Trees and Extreme Gradient Boosting (medium.com, 2018).

Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which generates a prediction model in the form of an ensemble of weak prediction models, typically decision trees. Like 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.

Gradient boosting is typically used with decision trees of a fixed size as base learners, also called Gradient Boosted Trees.

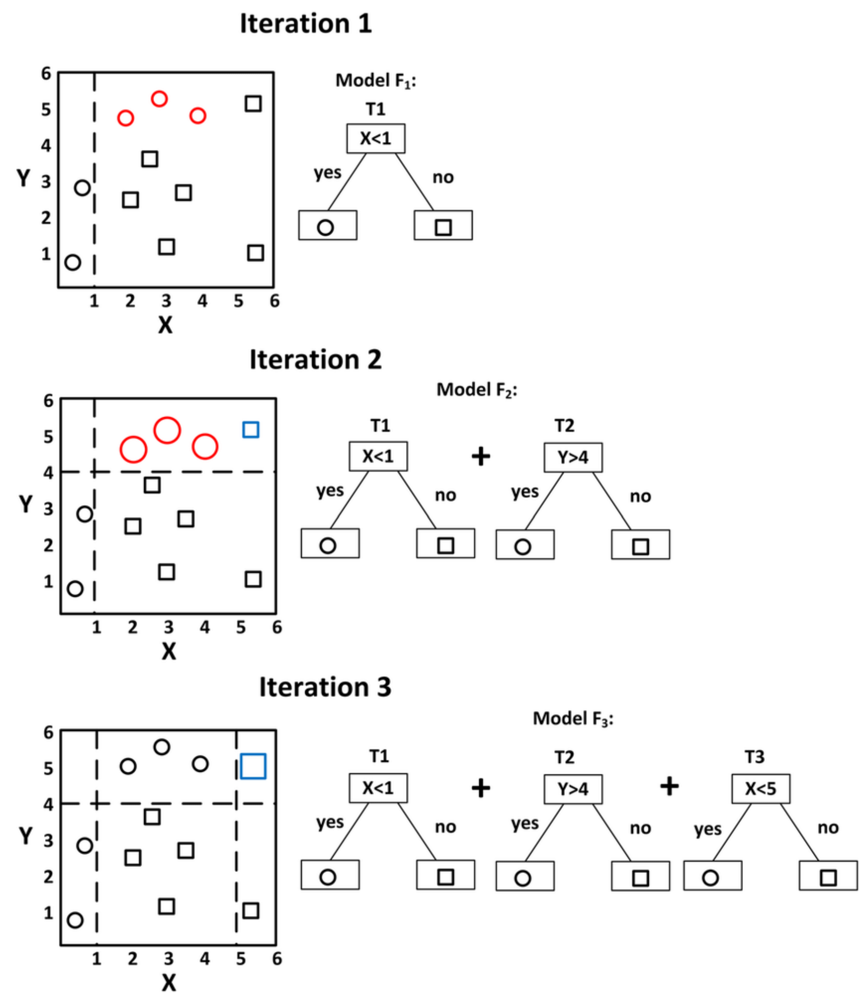


Figure 1: Gradient Boosting visual explanation.

Extreme Gradient Boosting

It is a specific implementation of the Gradient Boosting method which uses more accurate approximations to find the best tree model. It employs several nifty tricks that make it exceptionally successful, particularly with structured data.

Considering the potential loss for all possible splits to create a new branch, especially if you consider the case where there are thousands of features, and therefore thousands of possible splits. XGBoost treats this inefficiency by looking at the distribution of features across all data points and using this information to reduce the search space of possible feature splits.

Also, XGBoost implements a few regularization tricks, this speed up is by far the most useful feature of the library, allowing many hyperparameter settings to be investigated quickly. This is helpful because there are many hyperparameters to tune. Nearly all of them are designed to limit overfitting.

How does XGBoost perform so well?

XGBoost and Gradient Boosting are both ensemble tree methods that apply the principle of boosting weak learners using the gradient descent architecture. However, XGBoost improves upon the base Gradient Boosting framework through systems optimization and algorithmic enhancements (towardsdatascience.com, 2019).

Algorithm optimizations

Parallelization

XGBoost approaches the process of sequential tree building using parallelized implementation. XGBoost doesn't run multiple trees in parallel. Rather it does the parallelization within a single tree my using to create branches independently. XGBoost uses three parallelization methods: **Node Building at Each Level, Split Finding on Each Node** and **Split Finding at Each Level by Features.** More detailed information about these three parallelization methods can be found on the following reference (zhanpengfang.github.io, s.f.).

Hardware Optimization

This algorithm has been designed to make efficient use of hardware resources. This is accomplished using cache, allocating internal buffers in each thread to store gradient statistics.

Algorithm improvements

Regularization

Advanced regularization (L1 and L2), which improves model generalization. It penalizes more complex models to prevent overfitting. More information about L1 and L2 regularization can be found on (towardsdatascience.com, 2019).

Sparsity Awareness

XGBoost naturally admits sparse features for inputs by automatically 'learning' best missing value depending on training loss and handles different types of sparsity patterns in the data more efficiently.

Weighted Quantile Sketch

XGBoost employs the distributed weighted Quantile Sketch algorithm to effectively find the optimal split points among weighted datasets.

Cross-Validation

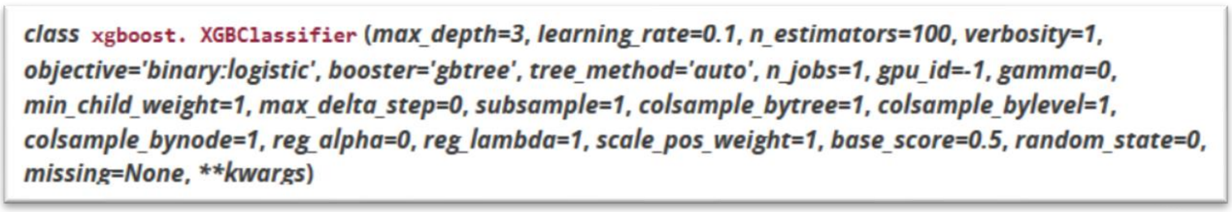
The algorithm comes with built-in cross-validation method at each iteration.

XGBoost framework on python

The XGBoost library on python can be divided in 2 parts. On the one hand we find the implementations compatible with Sklearn (XGBClassifier and XGBRegressor) and on the other hand we find 'xgboost.train' that is the low-level API to train the model via gradient boosting method.

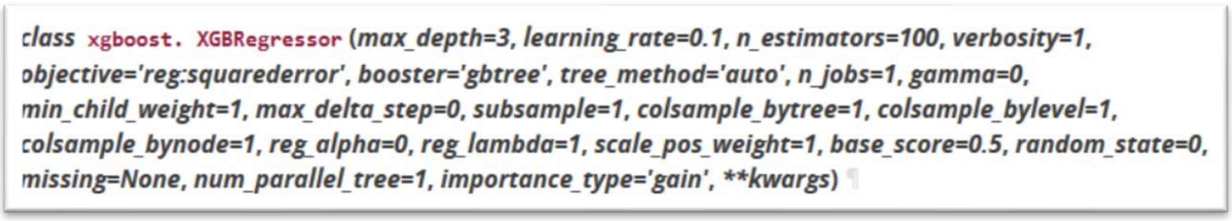
I have created a notebook using these two methods, implementing them in a classification problem.
https://github.com/xetxezarreta/Aprendizaje_Automatico/blob/master/XGBoost/xgboost.ipynb

XGBoost algorithm parameters



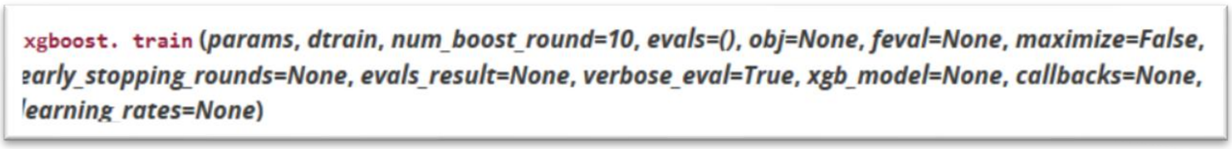
```
class xgboost. XGBClassifier (max_depth=3, learning_rate=0.1, n_estimators=100, verbosity=1,
objective='binary:logistic', booster='gbtree', tree_method='auto', n_jobs=1, gpu_id=-1, gamma=0,
min_child_weight=1, max_delta_step=0, subsample=1, colsample_bytree=1, colsample_bylevel=1,
colsample_bynode=1, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, base_score=0.5, random_state=0,
missing=None, **kwargs)
```

Figure 2: XGBClassifier parameters.



```
class xgboost. XGBRegressor (max_depth=3, learning_rate=0.1, n_estimators=100, verbosity=1,
objective='reg:squarederror', booster='gbtree', tree_method='auto', n_jobs=1, gamma=0,
min_child_weight=1, max_delta_step=0, subsample=1, colsample_bytree=1, colsample_bylevel=1,
colsample_bynode=1, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, base_score=0.5, random_state=0,
missing=None, num_parallel_tree=1, importance_type='gain', **kwargs)
```

Figure 3: XGBRegressor parameters



```
xgboost. train (params, dtrain, num_boost_round=10, evals=(), obj=None, feval=None, maximize=False,
early_stopping_rounds=None, evals_result=None, verbose_eval=True, xgb_model=None, callbacks=None,
learning_rates=None)
```

Figure 4: xgboost.train parameters

As we see, there are lot of parameters to tune. In 'xgboost.train' you define the booster parameters in 'params', as it appears in the example notebook. We can find a detailed information about the parameters on the official XGBoost wiki (xgboost.readthedocs.io, 2019).

We will now describe some relevant booster parameters:

- **max_depth:** maximum depth of each tree
- **eta:** training step for each iteration

- **silent:** logging mode
- **objective:** error evaluation for multiclass training
- **num_class:** the number of classes that exists in the dataset
- **num_round:** the number of training iterations

Bibliography

arxiv.org. (2016). Retrieved from https://arxiv.org/pdf/1603.02754.pdf

github.com. (2019). Retrieved from https://github.com/dmlc/xgboost

medium.com. (2018). Retrieved from https://medium.com/@gabrieltseng/gradient-boosting-and-xgboost-c306c1bcfaf5

towardsdatascience.com. (2019). Retrieved from https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

towardsdatascience.com. (2019). Retrieved from https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c

xgboost.readthedocs.io. (2019). Retrieved from https://xgboost.readthedocs.io/en/latest/python/python_api.html

zhanpengfang.github.io. (n.d.). Retrieved from https://zhanpengfang.github.io/418home.html