Boosting Brilliance: Unveiling the Power of XGBoost, AdaBoost, LightGBM, and CatBoost

Mentorness ML Internship: Article Writing, Assignment-I, March 3, 2024

Suvrojyoti Biswas
Batch: MIP-ML-07

Concept and Overview

Suppose we have a simple dataset with one feature and the target variable is a continuous value. Our initial prediction is the mean of the target variable. In the first iteration, we can fit a decision tree to the negative gradient of the loss function, which represents the residuals from the initial prediction. We can adjust the predictions of this tree by a small fraction (learning rate) and add them to our initial prediction. This process continues for a specified number of iterations until the loss is minimized. For instance, let us assume our initial prediction for a house price is $200,000. In the first iteration, we find that the residuals (errors) are -$10,000. We will fit a decision tree to predict these residuals and adjust them by, say, 0.1, resulting in -$1,000. We then add this to our initial prediction, resulting in a new prediction of $199,000. This process continues, with subsequent weak learners focusing on the remaining errors until convergence. This iterative process effectively constructs a strong predictive model by sequentially improving upon the residuals of the previous models, leading to enhanced accuracy and generalization.

Gradient Boosting is a powerful machine learning technique used for both regression and classification tasks. Gradient Boosting builds a strong predictive model by combining multiple weak learners, typically decision trees, in a sequential manner. Gradient Boosting aims to minimize a loss function by iteratively adding weak learners. At each iteration, a weak learner is trained to predict the gradient of the loss function with respect to the current ensemble's predictions. The weak learner is then added to the ensemble in a way that reduces the loss function when combined with the existing ensemble. Gradient Boosting can be viewed as a form of gradient descent, where each weak learner is fitted to the negative gradient of the loss function. For example, let us consider a regression problem where we want to predict housing prices based on features like square footage, number of bedrooms, etc. We start with an initial prediction (often the mean of the target variable). Then, at each iteration, we fit a weak learner

(usually a decision tree) to the negative gradient of the loss function with respect to the current predictions. This weak learner captures the residuals (errors) of the previous model. The predictions of the weak learner are then scaled by a learning rate and added to the current model.

Role of Gradient Boosting

Gradient Boosting encompasses several techniques that enhance predictive performance and address various challenges encountered in machine learning tasks. There one of the methods is $\mathcal{L}_2$-regularization technique, it adds a penalty term proportional to the square of the model's coefficients to the loss function, preventing overfitting and promoting smoother solutions. The regularized objective function can be expressed as, $\sum_{i=1}^{N} L(y_i, F(x_i)) + \lambda \sum_{j=1}^{J} \theta_j^2$ where, L is the loss function, $F(x_i)$ represents the model's prediction, λ controls the regularization strength, J is the number of features and θ_j are the feature coefficients. Further this can be modified by introducing the stochastic process. In stochastic gradient boosting, a random subset of samples is used to fit each base learner. This introduces randomness and can improve generalization. Let 'S' be the randomly selected subset of samples. Then objective function becomes $\sum_{i \epsilon S} L(y_i, F(x_i))$ instead of the previous.

Another technique is Quantile regression, it is used for modeling different quantiles of the target distribution, making the model robust to outliers and providing richer insights into the data distribution. For a specific quantile 'τ' the objective function for quantile regression can be expressed as, $\Theta = \sum_{i=1}^{n} \rho_\tau(y_i - F_t(x_i)) + \lambda . \Omega(f_t)$; where '$\rho_\tau(\mathbf{z})$' is the quantile loss function, y_i is the true response for i^{th} observation, $F_t(x_i)$ is the predicted value at t^{th} iteration for i^{th} observation, '$\Omega(f_t)$' represents the regularization term on the base learner 'f_t' . Again we have another two helpful techniques which are Tree Constraints and Constraints on Tree Leaves and Handling Missing Values. Incase of tree constraints, limiting the maximum depth or minimum number of samples required to split a node can prevent overfitting and improve model interpretability. And for handling missing value techniques like surrogate splits or dedicated treatment of missing values during tree construction help utilize data effectively. There are methods such as mean imputation or probabilistic imputation that can be used to handle missing values before fitting the model. Now the objective function is given as follows, $\Phi = \sum_{i=1}^{n} L(y_i, F_k(x_i)) + \sum_{m=1}^{T} \Omega(f_{k,m})$ and $\sum_{m=1}^{T} \Omega(f_{k,m}) = \sum_{m=1}^{T} (\alpha . |f_{k,m}| + \frac{\beta}{2} . (f_{k,m})^2)$

where '$\Omega(f_{k,m})$' represents the regularization term on the m^{th}-leaf in the k^{th}-tree and α, β are controlling the strengths of $\mathcal{L}_1$ and $\mathcal{L}_2$ regularizations respectively. Lastly to deal with the gaps, for each information split in a tree at node 'm' the split information gain is calculated as, $\Im g = \frac{Ig(before\ split) - Ig(after\ split)}{Number\ of\ gaps\ in\ the\ split}$.

Overall, we can say that Gradient Boosting techniques offer flexible solutions for various machine learning tasks, each addressing specific challenges encountered in model training and optimization. By incorporating advanced mathematical and statistical principles, these techniques not only enhance predictive performance but also improve model interpretability and robustness in real-world scenarios.

Four Prime Models

Now I have delineated in this section each main boosting algorithm one by one, including XGBoost, AdaBoost, LightGBM, and CatBoost, exploring their underlying principles, computational aspects, and applications in boosting model accuracy.

Firstly, XGBoost is an optimized gradient boosting algorithm known for its speed and performance. It uses decision trees as base learners and iteratively improves them to minimize a specified loss function. One can demonstrate how XGBoost optimizes a simple regression problem using the Boston Housing dataset. It can be shown how XGBoost iteratively adds decision trees and updates their weights to minimize the mean squared error loss function. Secondly, the AdaBoost algorithm combines multiple weak learners to create a strong learner. It adjusts the weights of misclassified data points in each iteration to focus on difficult-to-classify instances. It can be illustrated how AdaBoost improves the classification of a dataset with two classes using decision stumps as weak learners hence, can be shown how it adjusts the weights of misclassified instances in each iteration to improve overall accuracy.

Next is LightGBM that uses tree-based learning algorithms. It is designed for speed and efficiency, using a novel technique called histogram-based splitting to speed up the training process. One can showcase how LightGBM efficiently handles large datasets with high dimensionality by training it on a dataset with a large number of features. Highlight its ability to handle categorical features directly without requiring preprocessing. Lastly, the CatBoost algorithm does optimization for categorical features. It automatically handles categorical variables and does not require manual preprocessing, making it convenient for real-world datasets. One can demonstrate how CatBoost improves classification accuracy on a dataset with a mix of numerical and categorical features hence, can show how it automatically handles categorical variables and produces accurate predictions without additional feature engineering.

However, the choice of gradient boosting algorithm depends on factors such as dataset size, computational resources, and the need for handling categorical features. Understanding the

theoretical foundations, advantages, and limitations of each algorithm is crucial for selecting the most appropriate algorithm for specific tasks.

Limitations over Benefits

XGBoost is renowned for its speed and performance. It supports various objective functions and provides flexibility in model tuning. Additionally, it offers built-in support for handling missing values and parallel computation. XGBoost may require careful parameter tuning, and its performance can degrade with large datasets due to memory constraints. AdaBoost is simple to implement, resistant to overfitting, and performs well with a variety of base classifiers. It effectively handles imbalanced datasets. AdaBoost is sensitive to noisy data and outliers. It can be computationally expensive, especially with large datasets. LightGBM is highly efficient and scalable, making it suitable for large datasets. It supports categorical features natively and offers superior performance in terms of training speed. LightGBM may be memory-intensive due to its leaf-wise growth strategy. It may also require careful tuning of parameters for optimal performance. CatBoost automatically handles categorical variables and reduces the need for manual preprocessing. It performs well with default parameters and is robust to overfitting. CatBoost may have longer training times compared to other algorithms, especially with large datasets.

Comparative Study

Comparative analysis of XGBoost, AdaBoost, LightGBM, and CatBoost, focusing on performance metrics, training speed, quantifying impact factors, measuring capacity, and ease of use, all of these were discussed one by one as follows:

1. *Performance Metrics*
    - XGBoost- Known for its robust performance across various metrics such as accuracy, precision, recall, and F1 score. It handles imbalanced datasets well and can optimize different loss functions for regression and classification tasks.
    - AdaBoost- Achieves good performance in terms of accuracy and generalization. It is effective in boosting weak learners and improving overall model performance.
    - LightGBM- Offers competitive performance metrics with a focus on accuracy, precision, and recall. It is optimized for speed and efficiency, making it suitable for large-scale datasets and real-time applications.
    - CatBoost- Excels in handling categorical features and achieves high accuracy and generalization performance. It is robust to overfitting and performs well with default parameters.
2. *Training Speed*
    - XGBoost- Offers efficient training speed, especially with parallel computation support. It may slow down with large datasets due to memory constraints.

- AdaBoost- Generally slower compared to other algorithms due to sequential training of weak learners. It may be less suitable for large datasets and real-time applications.
- LightGBM- Known for its fast training speed, thanks to its histogram-based approach and leaf-wise growth strategy. It efficiently utilizes resources and scales well with large datasets.
- CatBoost- Offers competitive training speed, although it may be slower compared to LightGBM. It optimizes training time through effective handling of categorical features and parallel computation.

3. *Quantifying Impact Factors*
    - XGBoost- Provides flexibility in optimizing various hyperparameters, including tree depth, learning rate, and regularization. It allows fine-tuning to optimize model performance and generalization.
    - AdaBoost- Relatively simpler in terms of parameter tuning compared to other algorithms. It focuses on adjusting the weights of misclassified instances to improve overall model performance.
    - LightGBM- Offers a wide range of parameters for optimization, including feature fraction, bagging fraction, and maximum depth of trees. It allows for fine-grained control over model complexity and training speed.
    - CatBoost- Requires minimal parameter tuning and performs well with default settings. It automatically handles categorical features and focuses on minimizing overfitting while maximizing predictive performance.
4. *Measuring Capacity:*
    - XGBoost- Known for its high capacity to capture complex patterns and relationships in the data. It is capable of building deep trees and handling nonlinearities effectively.
    - AdaBoost- Offers moderate capacity, focusing on iteratively improving weak learners to create a strong ensemble model. It may struggle with highly complex datasets and nonlinear relationships.
    - LightGBM- Provides high capacity with its ability to build deep trees and capture intricate patterns in the data. It is well-suited for complex datasets with high-dimensional features.
    - CatBoost- Offers moderate to high capacity, balancing model complexity with robustness against overfitting. It effectively handles categorical features and captures nonlinear relationships in the data.
5. *Ease of Use*
    - XGBoost- Requires moderate expertise for parameter tuning and optimization. It offers extensive documentation and community support for troubleshooting.
    - AdaBoost- Simple to implement and requires minimal parameter tuning. It is suitable for beginners and practitioners looking for a straightforward boosting algorithm.
    - LightGBM- Offers ease of use with intuitive parameter settings and efficient training process. It provides clear documentation and examples for users to get started quickly.

- CatBoost- Requires minimal parameter tuning and offers simplicity in handling categorical features. It is suitable for users looking for a hassle-free boosting algorithm with competitive performance.

Practical Employment and Application Approach

Before I have described XGBoost can be used for predicting housing prices based on features such as location, size, and amenities. Its ability to handle missing values and optimize different loss functions makes it suitable for regression tasks. Now, the rest of the three are also described in this section as AdaBoost can be used for face detection in images. By iteratively adjusting the importance of features based on misclassified instances, AdaBoost improves the classification accuracy of face detection algorithms. LightGBM can be used for click-through rate prediction in online advertising. Its speed and efficiency allow advertisers to quickly process large volumes of data and make real-time bidding decisions. CatBoost can be used for customer churn prediction in telecommunications. Its ability to handle categorical features allows telecom companies to analyze customer behavior and proactively retain high-value customers.

Here, I mainly want to emphasize how these four ensemble techniques can be utilized to observe a machine learning problem based on current-world scenarios. I considered a critical machine learning problem of credit risk assessment. In this situation, my aim is to predict whether a borrower is likely to default on a loan based on various features such as credit history, income, debt-to-income ratio, and employment status.

A. Problem Statement

Given a dataset containing information about past loan applicants, including both defaulters and non-defaulters, our goal is to build a robust predictive model that accurately identifies potential defaulters to mitigate financial risk for lending institutions.

B. Drawn on for Implementation

(i) XGBoost:

- Objective Function- XGBoost optimizes a regularized objective function, combining the loss function and regularization terms. In our credit risk assessment problem, we can use a binary logistic loss function to model the probability of default.
- Tree Construction- XGBoost constructs decision trees in a gradient-boosting framework, iteratively minimizing the logistic loss function and updating model parameters.
- Regularization- XGBoost applies ridge and lasso regularization to prevent overfitting and improve model generalization, essential for handling complex datasets with potential noise and outliers.

- Handling Categorical Features- Categorical features, such as employment status and loan purpose, require preprocessing before fitting into the model to ensure compatibility with XGBoost's framework.

(ii) AdaBoost:

- Objective Function- AdaBoost optimizes the exponential loss function by sequentially adding weak learners. In our problem, each weak learner could be a decision stump, learning to classify instances based on specific features.
- Tree Construction- AdaBoost constructs an ensemble of weak learners, emphasizing misclassified instances in subsequent iterations. The algorithm assigns higher weights to misclassified instances, effectively focusing on difficult-to-classify samples.
- Regularization- AdaBoost does not directly incorporate regularization. However, early stopping and constraints on weak learners' complexity help prevent overfitting.
- Handling Categorical Features- Similar to XGBoost, AdaBoost requires preprocessing of categorical features to transform them into numerical representations.

(iii) LightGBM:

- Objective Function- LightGBM optimizes the objective function using gradient-based techniques, with built-in support for various loss functions. For our problem, we can use the binary cross-entropy loss to model default probabilities.
- Tree Construction- LightGBM employs an efficient histogram-based approach for tree construction, reducing computation and memory usage. This approach is particularly advantageous for handling large datasets with numerous features.
- Regularization- LightGBM offers lasso and ridge regularization to control model complexity, enhancing robustness and preventing overfitting.
- Handling Categorical Features- LightGBM directly handles categorical features during tree construction, eliminating the need for preprocessing and improving training efficiency.

(iv) CatBoost:

- Objective Function- CatBoost optimizes the objective function using gradient-based methods, incorporating categorical feature support directly. The objective function may include various loss functions, such as binary classification loss for our problem.
- Tree Construction- CatBoost constructs trees with built-in support for categorical features, leveraging ordered boosting to optimize the objective function efficiently.

- Regularization- CatBoost applies lasso and ridge regularization to control model complexity and prevent overfitting, similar to other boosting algorithms.
- Handling Categorical Features- CatBoost provides native support for categorical features, eliminating the need for preprocessing and reducing the risk of information loss during feature transformation.

So, from the problem of credit risk assessment one can get that each boosting algorithm offers unique advantages and approaches to model construction and optimization. While XGBoost and LightGBM excel in handling large datasets with extensive feature engineering, AdaBoost's sequential learning approach may provide interpretability and robustness in certain scenarios. CatBoost's native support for categorical features makes it particularly suitable for datasets with heterogeneous feature types. Understanding the mathematical formulations and application nuances of each algorithm is crucial for selecting the most appropriate approach based on the problem requirements and dataset characteristics.

Conclusion

After careful analysis, it is evident that each of the four gradient boosting models - XGBoost, AdaBoost, LightGBM, and CatBoost - offers unique strengths and characteristics. XGBoost stands out for its speed and performance, making it suitable for various applications. AdaBoost, known for its simplicity and resistance to overfitting, remains a popular choice for beginners. LightGBM's efficiency in training and prediction, especially with large datasets, sets it apart. CatBoost excels in handling categorical features and offers robust performance with minimal parameter tuning. Understanding the specific requirements of the task, including dataset size, computational resources, and the need for handling categorical features, is crucial in selecting the most appropriate model. Each model contributes distinct advantages to the gradient boosting landscape, catering to diverse machine learning challenges.

Github Implementation

Programming URL: https://github.com/suvro5495/Gradient-Boosting-Ensemble-Learning-