XGBoost Algorithm

January 27, 2025

1 Introduction

XGBoost (eXtreme Gradient Boosting) is a powerful machine learning algorithm that has gained popularity for its efficiency and performance. It is an implementation of gradient boosted decision trees designed for speed and performance.

2 Key Features

- **Speed and Performance:** XGBoost is known for its fast training speed and high performance.
- **Regularization:** It includes L1 and L2 regularization to prevent overfitting.
- **Handling Missing Values:** XGBoost can handle missing values internally.
- **Parallel Processing:** It supports parallel processing, making it suitable for large datasets.

3 How XGBoost Works?

It builds decision trees sequentially with each tree attempting to correct the mistakes made by the previous one. The process can be broken down as follows:

1. Start with a base learner: The first model decision tree is trained on the data. In regression tasks this base model simply predict the average of the target variable.

2. Calculate the errors: After training the first tree the errors between the predicted and actual values are calculated.

3. Train the next tree: The next tree is trained on the errors of the previous tree. This step attempts to correct the errors made by the first tree.

4. Repeat the process: This process continues with each new tree trying to correct the errors of the previous trees until a stopping criterion is met.

5. Combine the predictions: The final prediction is the sum of the predictions from all the trees.

4 Maths Behind XGBoost ALgorithm

It can be viewed as iterative process where we start with an initial prediction often set to zero. After which each tree is added to reduce errors. Mathematically, the model can be represented as:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)$$

Where $\hat{y}_i$ is the final predicted value for the i^{th} data point, K is the number of trees in the ensemble and $f_k(x_i)$ represents the prediction of the K^{th} tree for the i^{th} data point.

The objective function in XGBoost consists of two parts: a loss function and a regularization term. The loss function measures how well the model fits the data and the regularization term simplify complex trees. The general form of the loss function is:

$$obj(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Where:

- $l(y_i, \hat{y}_i)$ is the loss function which computes the difference between the true value y_i and the predicted value $\hat{y}_i$,
- $\Omega(f_k)$ is the regularization term which discourages overly complex trees.

Now, instead of fitting the model all at once we optimize the model iteratively. We start with an initial prediction $\hat{y}_i^{(0)} = 0$ and at each step we add a new tree to improve the model. The updated predictions after adding the t^{th} tree can be written as:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$$

Where $\hat{y}_i^{(t-1)}$ is the prediction from the previous iteration and $f_t(x_i)$ is the prediction of the t^{th} tree for the i^{th} data point.

The regularization term $\omega(f_t)$ simplify complex trees by penalizing the number of leaves in the tree and the size of the leaf. It is defined as:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Where:

- T is the number of leaves in the tree
- γ is a regularization parameter that controls the complexity of the tree
- λ is the regularization parameter that penalizes the squared weight of the leaves w_j

Finally when deciding how to split the nodes in the tree we compute the information gain for every possible split. The information gain for a split is calculated as:

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma$$

Where

- G_L, G_R are the sums of gradients in the left and right child nodes
- H_L, H_R are the sums of Hessians in the left and right child nodes

5 Conclusion

XGBoost is a versatile and powerful algorithm that has become a go-to choice for many machine learning practitioners. Its ability to handle large datasets and provide high performance makes it an essential tool in the machine learning toolkit.

References

[1] Tianqi Chen and Carlos Guestrin, *XGBoost: A Scalable Tree Boosting System*, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.