XGBoost as Regressor and Classifier

Asha Latha
Jan 20 2021 · 5 min read

XGBoost was developed by Tianqi Chen and Carlos Guestrin and it is an ensemble machine learning technique that uses the Gradient boosting framework for machine learning prediction. 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 well known for its fast execution and Scalability.

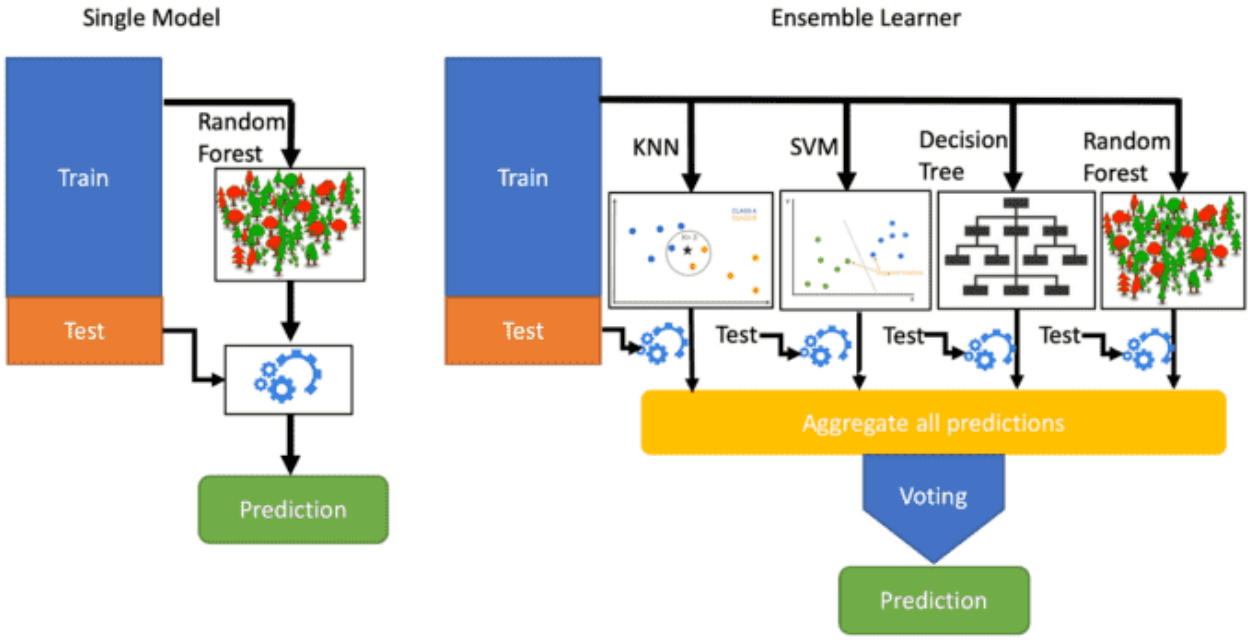
For installing XGBoost,

pip3 install xgboost

This article needs fair amount of knowledge about Decision tree,Ensemble,Boosting and Gradient Boosting.

Decision tree : Decision tree is a non-parametric machine learning algorithm that has a tree-like graph structure that is used for both classification and prediction , where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Ensemble : Ensemble methods are techniques that aim at improving the accuracy and stability of results in models by combining multiple models instead of using a single model. The combined models increase the accuracy and stability of the results significantly. This has boosted the popularity of ensemble methods in machine learning.



Single model Vs Ensemble Model

Boosting : Boosting is an ensemble modeling technique which attempts to build a strong classifier from the number of weak classifiers. It is done building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model.This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added. Generally, decision trees are default base learners for boosting.

Gradient Boosting : Gradient Boosting is a boosting technique wherein each iteration the new predictor is built to fit on the pseudo-residuals of the previous predictor.

XGBoost

XGBoost is a special implementation of the Gradient Boosting and XGBoost stands for **Extreme Gradient Boosting**, XGBoost uses more accurate approximations by employing second-order gradients and advanced regularization.

XGBoost's objective function is the sum of loss function evaluated over all the predictions and a regularization function for all predictors (j trees). In the formula fj means a prediction coming from the jth tree.

$$obj(\theta) = \sum_{i}^{n} l(y_i - \hat{y}_i) + \sum_{j=1}^{j} \Omega(f_j)$$

Loss function depends on the task being performed (classification, regression, etc.)

XGBoost uses a popular metric called '**log loss**',probability-based metric is used to measure the performance of a classification model

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

XGBoost uses popular metrics like ‘**MSE**’, or '**MAE**' used to measure the performance of a regression model .

$$MSE = \frac{\sum_{i=1}^{n} (y_i - y_i^p)^2}{n}$$

$$MAE = \frac{\sum_{i=1}^{n} |y_i - y_i^p|}{n}$$

Mathematics Involved

Unlike the other tree-building algorithms, XGBoost doesn’t use entropy or Gini indices. Instead, it utilities gradient (the error term) and hessian for creating the trees. Mathematically, Hessian is a second order derivative of the loss at the current estimate given as:

$$h_m(x) = \frac{\partial^2 L(Y, f(x))}{\partial f(x)^2}_{f(x)=f^{(m-1)}(x).}$$

Where L is the loss function . For regression problems loss function could be linear and for classification problem loss function could be log loss function

Instead of using Gini or Entropy, XGBoost uses **Similarity Score** for calculating the Gain of the root node

Formula for Similarity Score is :

For Classification Problems :

$$\text{Similarity Score} = \frac{(\sum \text{Residual}_i)^2}{\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)] + \lambda}$$

For Regression Problems :

```
Similarity Score(S.S.) =  (S.R ^ 2) / (N + λ)
Here, S.R is the sum of residuals,
N is Number of Residuals
```

Where Lambda(reg_lambda) is L2 regularization term on weights.
Increasing this value will make model more conservative.

Gain of the root node is :

$$\text{Gain} = \text{Left}_{\text{Similarity}} + \text{Right}_{\text{Similarity}} - \text{Root}_{\text{Similarity}}$$

$$\text{Output Value} = \frac{\left(\sum \text{Residual}_i\right)}{\sum \left[\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)\right] + \lambda}$$

Output values are calculated only for leaves of a tree.

XgBoost Working

1. Initial prediction

Initialize the base predictor with mean of the the dependent variable for all data points in the dataset that is Fo(x) = ho(x).

2. Residuals calculation

Calculate residuals Ŷ for all datapoints from previous predictions from the last step where Ŷ = Y — Fo(x)

3. Train the model

Train the First model(Training model here refer to building trees) that is M1 using [X, Ŷ] as data and the model is a special Xgboost tree that is constructed differently when compared to a normal decision tree. (ie, using gradients and hessians)

$$Similarity = \frac{Gradient^2}{hessian + \lambda}$$

The similarity Score formula can be generalized in the above way.

(a) The trees are constructed by splitting data into two partitions of various possible splits

(b)The Threshold for root is calculated by taking an average of two close points among the split and the residual go to the respected leaf.

(c) If the dataset contain n number of points then n-1 number of trees can be constructed

(d) Calculate Similarities and Gain for all the constructed trees to find the tree with the optimal split.

(e) Choose the tree with maximum gain

If the gain from a node is found to be minimal then it just stops constructing the tree to a greater depth which can overcome the challenge of overfitting to a great extend.

Once the tree is built, it's time for Tree Pruning which is done to reduces the size of decision trees by removing sections of the tree that has little power to classify instances. The parameter Gamma is used for tree pruning.

If $Gain - \gamma < 0$, remove that branch. Else, keep the branch

(f) Split the selected tree further till a value of max_depth (default value is 6)for complete tree construction.

4. Obtain the the predictions , ie, the residuals from the model M1.

Now pass all data points through the Final Tree(Model-1) to get h1(x) and calculate the prediction F1(x) and residuals

New Residuals for the second model M2 is **y^ = y-F1(x)**

F1(x) Calculation

let Learning rate or eta(η) = 1.0

F1(x)=ho(x)+η*h1(x) --> For Regression

F1(x)=σ([ho(x)/(1-ho(x))]+[η*h1(x)]) --> For Classificatic

solve for F1(x) on classification to get

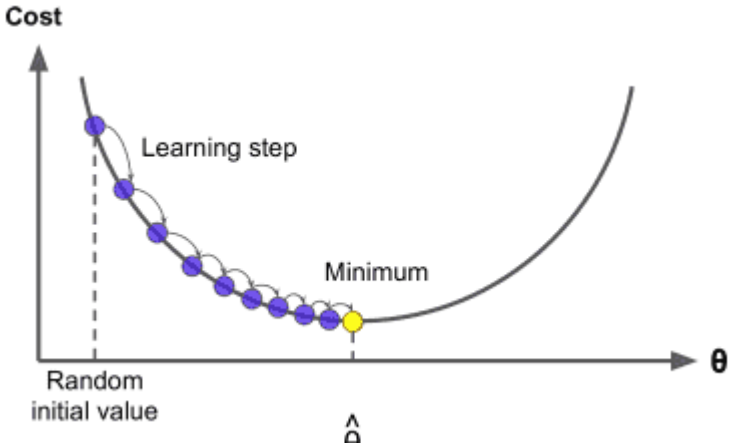
F1(x) = σ (0+1*(h1(x)))

Sigmoid Function:

σ (x)= 1/(1 + exp(-x))

5. Obtain Predictions from Model M2 , Now pass all data points through the model M2 to get h2(x) and calculate the predictions F2(x) and residuals ie, **y^ = y-F1(x)**

Continue adding Xg trees sequentially with input as all independent variables and the residuals . The trees getting added are trying to find the global minimum for the loss function using Gradient descent . The calculated contribution of each tree is based on minimizing the overall error of the strong learner.



Prediction from XGBoost

After training,

For Classification use cases:

if we want to predict for a new data point then we will use constructed trees or models to get all values to solve the equation **F2(x) = σ (0+1*h1(x)+1*h2(x))**) where the resulting value of F2(x) is considered as the prediction from XgBoost model.

For Regression use cases :

After training, if we want to predict for a new data point then we will use constructed trees or models to get all values to solve the equation **F2(x) =H0(x)+eta(H1(x)) +eta(H2(x))** where the resulting value of F2(x) is considered as the prediction from XgBoost model.

Comments

- Newest
- 3/17/2021
  VENKATESWARA REDDY

  Nice blog ,Thanks for explanation
- 1/20/2021
  Haneesha H

  Good explanation👍