Gradient Boosting

Jan-Philipp Kolb

21 Mai, 2019

Gradient Boosting

- Extremely popular, successful across many domains and one of the leading methods for winning Kaggle competitions.
- GBMs build an ensemble of shallow and weak successive trees with each tree learning and improving on the previous.
- When combined, these many weak successive trees produce a powerful "committee" that are often hard to beat with other algorithms.

Preparation

- The following slides are based on UC Business Analytics R Programming Guide on **gbm regression**

```
library(rsample)      # data splitting
library(gbm)          # basic implementation
library(xgboost)      # a faster implementation of gbm
library(caret)        # an aggregator package for machine learni
library(h2o)          # a java-based platform
library(pdp)          # model visualization
library(ggplot2)      # model visualization
library(lime)         # model visualization
```

The dataset

```r
set.seed(123)
ames_split <- initial_split(AmesHousing::make_ames(), prop = .7)
ames_train <- training(ames_split)
ames_test  <- testing(ames_split)
```

Advantages

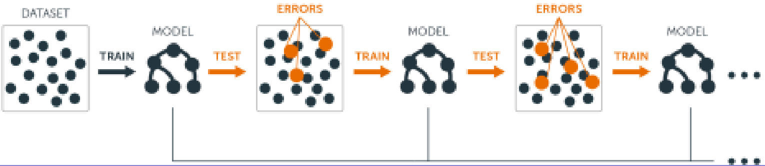
- Often provides predictive accuracy that cannot be beat.
- Lots of flexibility - can optimize on different loss functions and provides several hyperparameter tuning options that make the function fit very flexible.
- No data pre-processing required - often works great with categorical and numerical values as is.
- Handles missing data - imputation not required.

Disdvantages

- GBMs will continue improving to minimize all errors. This can overemphasize outliers and cause overfitting. Must use cross-validation to neutralize.
- Computationally expensive - GBMs often require many trees ($>$1000) which can be time and memory exhaustive.
- The high flexibility results in many parameters that interact and influence heavily the behavior of the approach (number of iterations, tree depth, regularization parameters, etc.). This requires a large grid search during tuning.
- Less interpretable although this is easily addressed with various tools (variable importance, partial dependence plots, LIME, etc.).

The idea

- Several supervised machine learning models are founded on a single predictive model (i.e. linear regression, penalized models, naive Bayes, support vector machines).
- Other approaches such as bagging and random forests are built on the idea of building an ensemble of models where each individual model predicts the outcome and then the ensemble simply averages the predicted values.
- The main idea of boosting is to add new models to the ensemble sequentially.
- At each particular iteration, a new weak, base-learner model is trained with respect to the error of the whole ensemble learnt so far.



Important concepts

Base-learning models

- Boosting is a framework that iteratively improves any weak learning model.
- Many gradient boosting applications allow you to “plug in” various classes of weak learners at your disposal.
- In practice, boosted algorithms often use decision trees as the base-learner.

Training weak models

- A weak model is one whose error rate is only slightly better than random guessing.
- The idea behind boosting is that each sequential model builds a simple weak model to slightly improve the remaining errors.
- Shallow trees represent a weak learner.
- Trees with only 1-6 splits are used.
- Combining many weak models (versus strong ones) has a few benefits:
- Speed: Constructing weak models is computationally cheap.
- Accuracy improvement: Weak models allow the algorithm to learn slowly; making minor adjustments in new areas where it does not perform well. In general, statistical approaches that learn slowly tend to perform well.
- Avoids overfitting: Due to making only small incremental improvements with each model in the ensemble, this allows us to stop

Sequential training with respect to errors

- Boosted trees are grown sequentially;
- each tree is grown using information from previously grown trees.
- The basic algorithm for boosted regression trees can be generalized to the following where x represents our features and y represents our response:

1.) Fit a decission tree: $F_1(x) = y$

2.) the next decission tree is fixed to the residuals of the previous: $h_1(x) = y - F_1(x)$

3.) Add this new tree to our algorithm: $F_2(x) = F_1(x) + h_1(x)$

4.) The next decission tree is fixed to the residuals of $F_2 : h_2(x) = y - F_2(x)$

5.) Add the new tree to the algorithm: $F_3(x) = F_2(x) + h_1(x)$

Continue this process until some mechanism (i.e. cross validation) tells us to stop.

Basic algorithm for boosted regression trees

The basic algorithm for boosted regression trees can be generalized to the following where the final model is simply a stagewise additive model of b individual regression trees:

$$f(x) = B\sum_{b=1}^{B} f^b(x)$$

Gradient descent

- Many algorithms, including decision trees, focus on minimizing the residuals and, therefore, emphasize the MSE loss function.
- This algorithm outlines the approach of sequentially fitting regression trees to minimize the errors.
- This specific approach is how gradient boosting minimizes the mean squared error (MSE) loss function.
- Often we wish to focus on other loss functions such as mean absolute error (MAE) or to be able to apply the method to a classification problem with a loss function such as deviance.
- The name gradient boosting machines come from the fact that this procedure can be generalized to loss functions other than MSE.

A gradient descent algorithm

- Gradient boosting is considered a gradient descent algorithm.
- Gradient descent is a very generic optimization algorithm capable of finding optimal solutions to a wide range of problems.
- The general idea of gradient descent is to tweak parameters iteratively in order to minimize a cost function.
- Suppose you are a downhill skier racing your friend.
- A good strategy to beat your friend to the bottom is to take the path with the steepest slope.
- This is exactly what gradient descent does - it measures the local gradient of the loss (cost) function for a given set of parameters (Φ) and takes steps in the direction of the descending gradient.
- Once the gradient is zero, we have reached the minimum.