May 8, 2023

Machine Learning Methods and Applications

Week 9. Ensemble learning | Boosting models and GBMs

Remember

- Ensembling is a technique of combining two or more models of similar or dissimilar types.
- Ensembling methods: bagging, boosting and stacking.
- The primary goal of ensemble models is to improve the prediction performance of the model, and also to overcome high variance and overfitting problems in tree-based models.
- Bootstrap aggregating trees is a general method for fitting multiple versions of a decision tree and then combining them into an aggregated prediction.
- The difference is random forests use the subset of the features to train each tree while bagging trees use the all features.

Boosting

Boosting

- Several supervised ML models are based on a single predictive model such as linear regression, decision tree, or etc.
- Bagging and random forests, work by combining multiple models together into an overall ensemble. New predictions are made by combining the predictions from the individual base models that make up the ensemble.
- Since averaging reduces variance, bagging are most effective applied to models with low bias and high variance.
- While boosting is a general algorithm for building an ensemble out of simpler models. It is more effectively applied to models with high bias and low variance.

Boosting

- The main idea of boosting is to add new models to ensemble sequentially.
- Boosting attacks the bias-variance-tradeoff by starting with a weak model (e.g. a decision tree with only a few splits) and sequentially boosts its performance by continuing to build new trees.

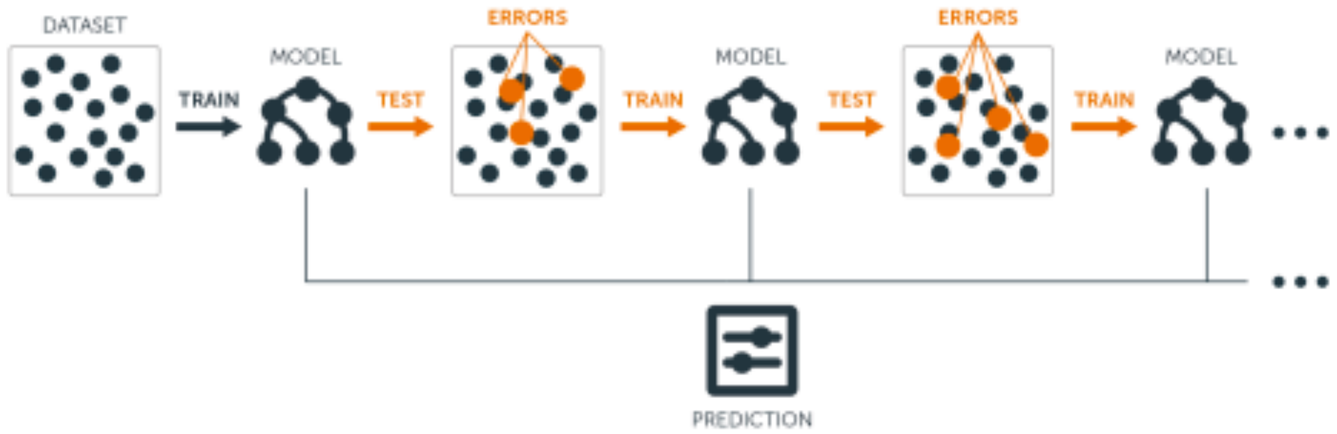


Image credit: Boehmke & Greenwell - Hands-On Machine Learning with R

Boosting

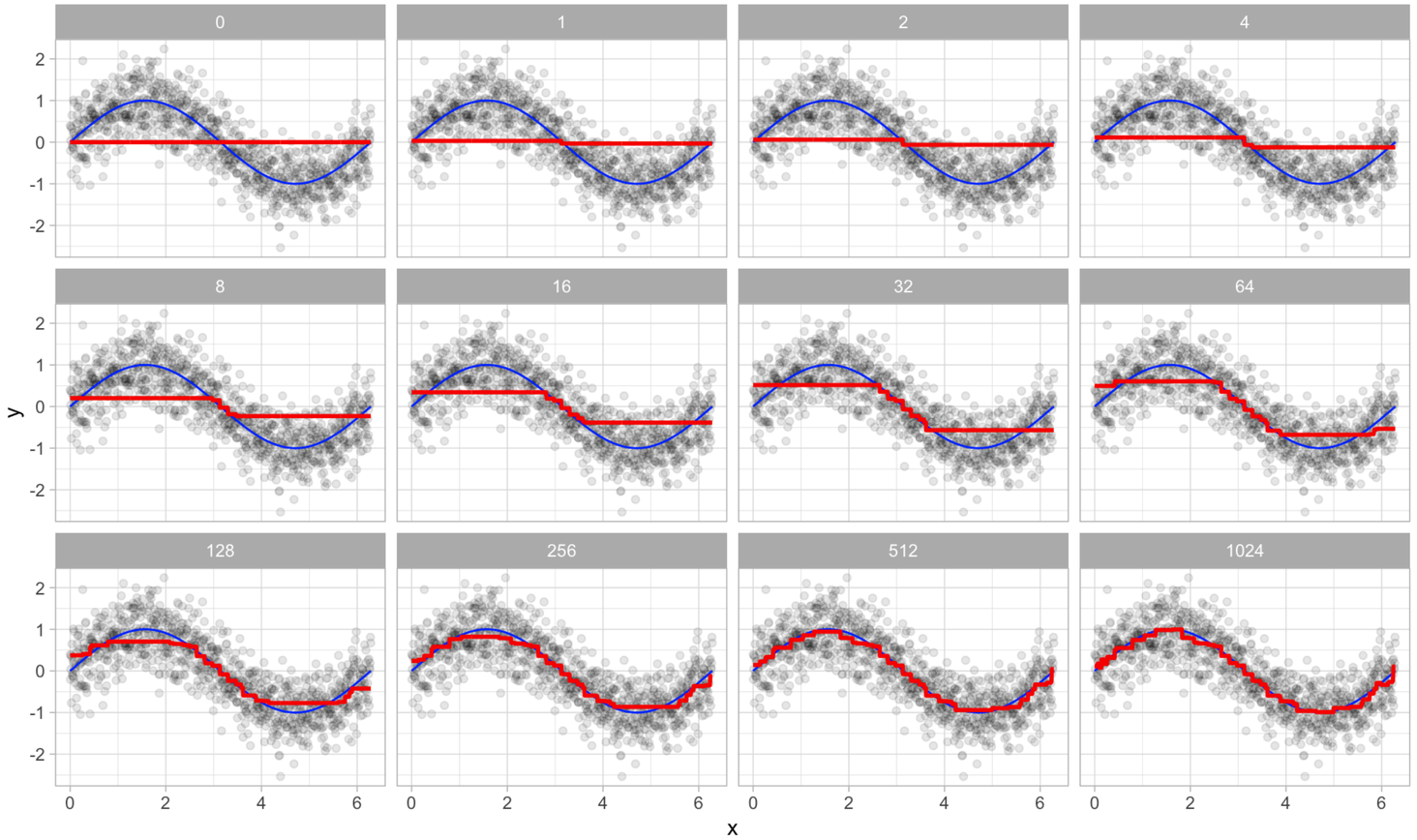


Image credit: Boehmke & Greenwell - Hands-On Machine Learning with R

1. Fit a decision tree to the data: $F_1(x) = y$,
2. We then fit the next decision tree 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. Fit the next decision tree to the residuals of F_2: $h_2(x) = y - F_2(x)$,
5. Add this new tree to our algorithm: $F_3(x) = F_2(x) + h_2(x)$,
6. Continue this process until some mechanism (i.e. cross validation) tells us to stop.

The final model here is a stagewise additive model of *b* individual trees:

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

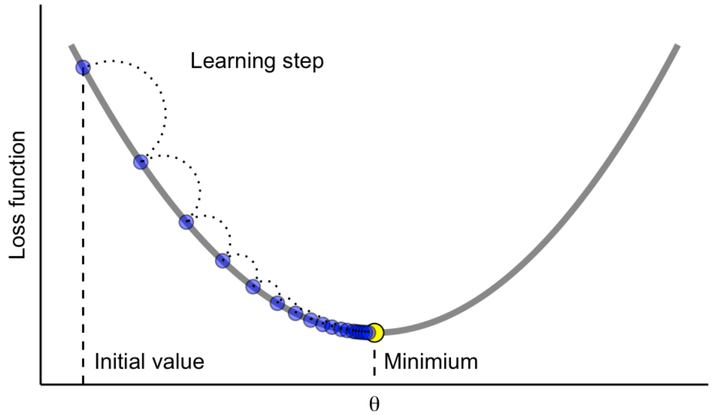
Boosting methods

1. **Gradient boosting machines**
2. **Extreme gradient boosting machines**
3. AdaBoost
4. Stochastic gradient boosting machines
5. …

Gradient Boosting Machines

Gradient descent

- **G**radient **B**oosting **M**achines is considered a Gradient descent algorithm.
- Gradient descent is a generic optimization algorithm to find an optimal solutions to a wide range of problems.

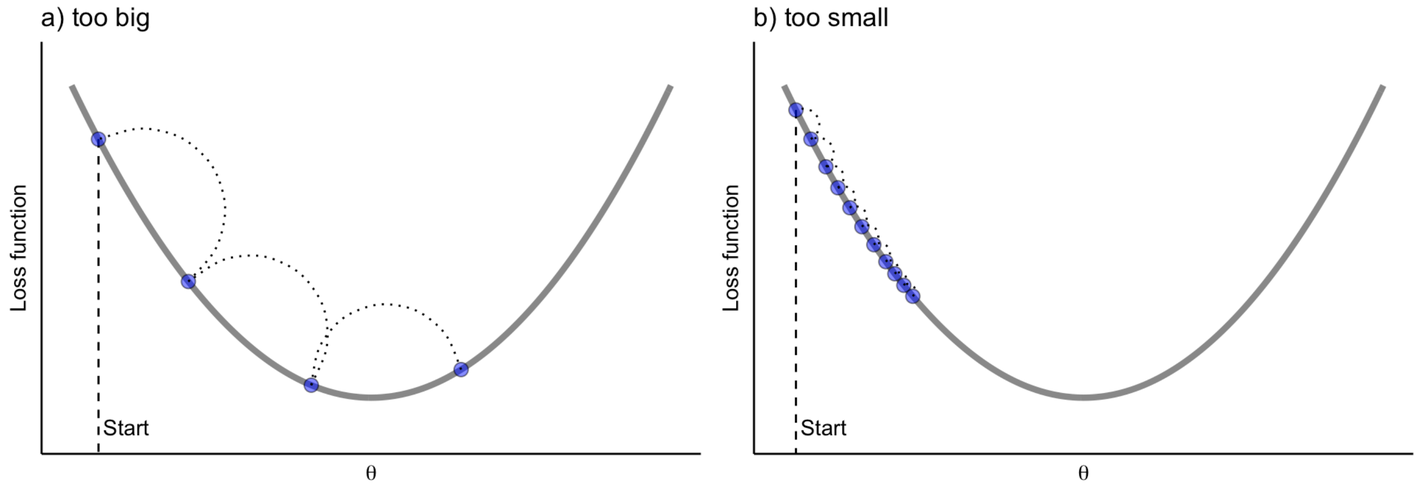


Gradient descent is the process of gradually decreasing the cost function (i.e. MSE) by tweaking parameter(s) iteratively until you have reached a minimum.
Image credit: Boehmke & Greenwell - Hands-On Machine Learning with R

The general idea of gradient descent is to tweak parameter(s) iteratively in order to minimize a cost function.

Gradient descent

- An important parameter in gradient descent is the size of the steps which is controlled by the learning rate.
- If the learning rate is too small, then the algorithm will take many iterations (steps) to find the minimum. On the other hand, if the learning rate is too high, you might jump across the minimum and end up further away.



A learning rate that is too small will require many iterations to find the minimum. A learning rate too big may jump over the minimum.
Image credit: Boehmke & Greenwell - Hands-On Machine Learning with R

Gradient boosting hyperparameters

- Number of trees
- Learning rate
- Tree depth
- Minimum number of observations in terminal nodes

Boosting hyperparameters
Tree hyperparameters

Gradient boosting hyperparameters

1. Number of trees

- The total number of trees in the sequence.
- The averaging/voting of independently grown trees make it very difficult to overfit with too many trees.

Gradient boosting hyperparameters

2. Learning rate

- Determines the contribution of each tree on the final outcome and controls how quickly the algorithm proceeds down the gradient descent (learns).
- Values range from 0–1.
- Smaller values make the model robust to the specific characteristics of each individual tree, thus allowing it to generalize well.
- Smaller values also make it easier to stop prior to overfitting; however, they increase the risk of not reaching the optimum with a fixed number of trees and are more computationally demanding.

Gradient boosting hyperparameters

3. Tree depth

- Controls the depth of the individual trees.
- Typical values range from a depth of 3–8.
- Smaller depth trees such as decision stumps are computationally efficient.
- However, higher depth trees allow the algorithm to capture unique interactions but also increase the risk of over-fitting.

Gradient boosting hyperparameters

4. Minimum number of observations in terminal nodes

- Controls the complexity of each tree.
- Typical values range from 5–15.
- Higher values help prevent a model from learning relationships which might be highly specific to the particular sample selected for a tree (overfitting).
- Smaller values can help with imbalanced target classes in classification problems.

Extreme Gradient Boosting

Extreme gradient boosting

- Extreme gradient boosting (XGBoost) is an optimized distributed gradient boosting library that is designed to be efficient, flexible, and portable across multiple languages.
- Although XGBoost provides the same boosting and tree-based hyperparameter options illustrated in the previous sections, it also provides a few advantages over traditional boosting.
- **xgboost** provides multiple regularization parameters to help reduce model complexity and guard against overfitting.

Interpretation

What about the interpretation of the bagging and boosting trees?

Application

See the R codes on the course GitHub repository!

The materials of today's lecture will be available on **GitHub**.
Feel free to contact me via e-mail: mustafacavus@eskisehir.edu.tr