Distinguish between Tree-Based Machine Learning Algorithms

BEGINNER MACHINE LEARNING SUPERVISED

This article was published as a part of the Data Science Blogathon.

Introduction

This article aims to distinguish tree-based Machine Learning algorithms (Classification and Regression Trees/CART) according to the complexity. Tree-based is a family of supervised Machine Learning which performs classification and regression tasks by building a tree-like structure for deciding the target variable class or value according to the features.

Tree-based is one of the popular Machine Learning algorithms used in predicting tabular and spatial/GIS datasets. Our discussion today will see 4 levels of tree-based Machine Learning from the simplest to the most complicated algorithms.

The tree-based model can be drawn like below. Starting from the top node, it divides into 2 branches at every depth level. The last end branches where they do not split anymore are the decisions, usually called the leaves. In every depth, there are conditions questioning the feature values. The binary answer will decide which branch we are going to next. This process continues until we reach one of the leaves, where it does not split anymore. We can get the prediction from that final leaf.

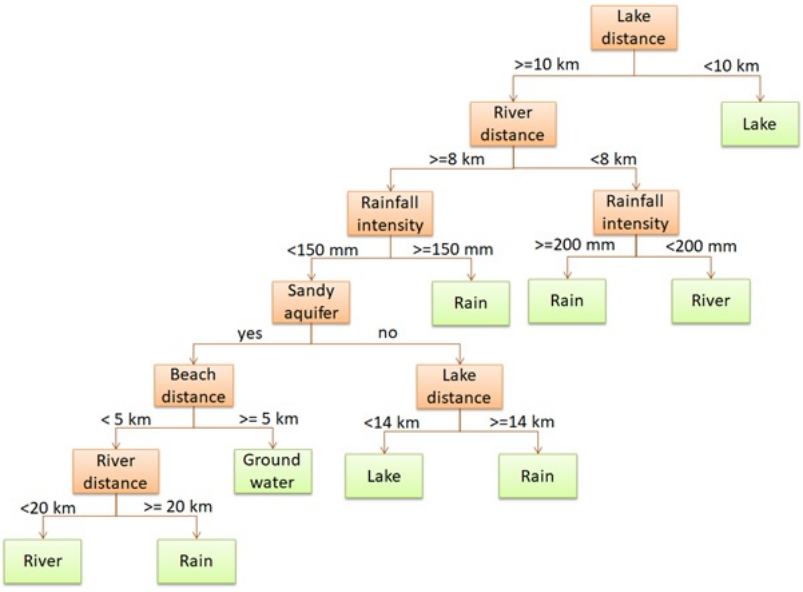

Fig. 1 Simple Decision Tree

The figure above is a simple example of a single Decision Tree created to predict which water resource a location is most suitable to utilize. For example, if there is a case where a location has features: 120 mm/month rainfall, sandy aquifer, 10 km away from the perennial river, 20 km away from the lake, and 2 km away from the beach, could you decide which water resource suitable for the community to take from, rainfall, river water, lake, or groundwater?

How Decision Tree is developed

Here is an illustration of how the Decision Tree algorithm works in segmenting a set of data points into 2 classes: “sold out” and “not sold out”. First, the algorithm will divide the data into two parts using a horizontal or vertical line. In this case, the first step is done by splitting the x-axis using a vertical line separating the price above and below $600. Next, the algorithm splits the y-axis into the left and right sides. We can see that for the price above $600, the products will be sold if the quality is above 60 and not sold if it is below 60. If the price is below $600, the algorithm needs further segmentation. Can you continue the process by observing the following figure? This illustration only accounts for 1 feature and 1 target variable. It will get more complicated with multiple features.

Fig. 2a Illustration of how Decision Tree works (source: https://medium.com/@datascience_gis/decision-tree-classification-and-regression-trees-and-random-forest-30c31988f1d5)

Fig. 2b Illustration of how Decision Tree works

Fig. 2c Illustration of how Decision Tree works

Besides tree-based, there are many other Machine Learning algorithms, such as k Nearest Neighbors (kNN), linear and logistic regression, Support Vector Machine (SVM), k-means, Principal Components Assessments, and so on. But, in my opinion, there are two things that make tree-based Machine Learning special. The first reason is that tree-based Machine Learning does not need feature scaling, like standardization or normalization in the preprocessing. The other Machine Learning algorithms, especially distance-based, usually need feature scaling to avoid features with high range dominating features with low range. The second reason is that tree-based Machine Learning has simple to complicated algorithms, involving bagging and boosting, available in packages.

1. Single estimator/model: Decision Tree

Let's start with the simplest tree-based algorithm. It is the Decision Tree Classifier and Regressor. Actually, we have discussed the Decision Tree in the second and third paragraphs. A single Decision Tree is created by fitting training data. Imagine we asked a robot to learn from 10,000 training data and later we want the robot to predict the other 20,000 test data for us. We can feel that only one piece of feedback is not enough. We need more robots to learn the training data and do more predictions for us.

Fig. 3 Single Decision Tree illustration

2. Ensemble Methods and Bagging: Random Forest

Now, we will bring it to the next level. The method of relying on more than one model is called Ensemble Method. We can apply more than one algorithms, like Decision Tree, Support Vector Machine, Neural Network, kNN, and Naive Bayes, to learn from the same training dataset and build multiple models, in this case, 5 models. The 5 models then predict the test dataset and the output is decided according to the majority voting.

Another Ensemble Method is to build multiple models using the same Machine Learning algorithm with a different subset of the training dataset. This is called Bagging Classifier or Regressor. For instance, we want to train 5 Decision Tree models. Each Decision Tree learns from subsamples of the training dataset. Observations in the training dataset can be learned by some models and not learned by other models. This will result in 5 similar Decision Tree models, not the same.

Next, the 5 models will predict the test data and the final prediction is decided by majority voting for classification or average for regression. For example, if the 4 models vote for “River” and the other one votes for “Rain”, the output is “River”. In the regression task, the final output is the average of each model output.

Bagging Classifier and Regressor can be applied to Decision Tree and other algorithms, such as KNN or SVM. Decision Tree is so popular for Bagging Machine Learning that it has its own package named Random Forest. Random Forest sounds like a large group of trees. It is indeed built from a number of Decision Tree models, 100 models by default, from sub-samples of the training dataset. Random Forest is at a higher level above the Decision Tree. Predicting using ensemble methods, like Random Forest, usually has higher accuracy than that using only a single model, such as a Decision Tree.

Fig. 4a Random Forest Classifier illustration

Fig. 4b Random Forest Regressor illustration

3. Boosting: Adaptive and Gradient Boosting Machine

Bagging or Random Forest Machine Learning creates a number of models at the same time separately. Each of the models is independent of the other. We can still improve our model accuracy using Boosting. Boosting, unlike Bagging, creates models one by one. Firstly, the first model learns from the training data. The second model then also learns from the same training dataset and the mistakes done by the first model. The third model, again, learns from the same training dataset and the mistakes of the previous model.

This continues until a number of models are built. Boosting can have higher accuracy because each base model/estimator gets to learn from the previous model mistakes. It is commonly described as turning individual weak learners into strong learners together. It is like having a number of cooperating robots, instead of learning individually.

Fig. 5 Boosting illustration

There are two kinds of Boosting algorithms. Adaptive Boosting works as the below pictures describe. The picture below tries to segment the training data to later create a Decision Tree. After the first model is created, the second model learns from the mistaken classification of the first model. The data points which are mistakenly predicted by the first model are given higher weights. The correctly predicted data points can also have their weights decrease. The second model is built by paying more attention to the data points with higher weights. The third model will again learn from the mistaken prediction of the previous model.

Fig. 6 Adaptive Boosting illustration

Another Boosting algorithm is Gradient Boosting. Just like Random Forest in Bagging method, tree-based Machine Learning has Gradient Boosting Machine (GBM) package ready in the Gradient Boosting method. Gradient Boosting, unlike Adaptive Boosting, learns from the residual errors (of the true values and predicted values) of the previous model to minimize it. This algorithm creates a series of models to lower the residual errors gradually.

The following figure explains how Gradient Boosting is developed. Determining the loss function is the first step to compute the residual errors. In this example, we will examine how RMSE is minimized gradually in the regression task. The model below fits the feature (x) and target variable (y) to build a model for predicting predicted variables (y1 until y5 and y50). This illustration only has 1 feature as a predictor for simplicity. The model also only has a maximum depth of 2 for simplicity.

Before the first model is created, the average of the target variable is computed as y0: 8.033. y0 is applied the same to all rows. Residual errors r1 is y – y0. The RMSE is still 1.96. Then, the first model will predict the r1 as r1_pred and the predicted target variable as y1 = y0 + r1_pred. The residual errors r2 are calculated from y – y1. The RMSE now is 1.84.

Now, let's observe the second model. The second model tries to fit X again to predict r2 as r2_pred. The second model also predicts the target variable y2 = y1 + r2_pred. The RMSE this time lowers again to be 1.74. The third model will continue with the same procedure. The RMSE of the 3rd, 4th, 5th, and 50th models keep minimizing to be 1.66, 1.58, 1.52, and finally 0.75 respectively.

In Figure 7, see how the prediction results gradually getting near to the true value y. Notice that every r1_pred to r5_pred consists of only 4 groups of numbers. This happens because, in the beginning, we set the maximum depth of GBM to be 2, which makes the maximum number of "leaves" to be $2^2=4$.

Fig. 7a How GBM works

Fig. 7a How GBM works

4. Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost), as its name suggests, is the next level of ordinary Gradient Boosting. It is expected to achieve “extremely” high accuracy. XGBoost, like Random Forest and Gradient Boosting, is built by multiple Decision Tree models as its base estimator/model. If you are using Python Programming Language. XGBoost is not in “scikit-learn” (sklearn) but in “XGBoost” package.

Gradient Boosting has the weakness of overfitting and XGBoost is designed as a regularized Gradient Boosting to reduce the overfitting problem. Overfitting is a problem because the model can predict well for the training dataset, but bad for the test dataset.

Summary

In summary, this article distinguishes tree-based Machine Learning into 4 complexity levels. The simplest model is the Decision Tree. A combination of Decision Trees builds a Random Forest. Random Forest usually has higher accuracy than Decision Tree does. A group of Decision Trees built one after another by learning their predecessor is Adaptive Boosting and Gradient Boosting Machine.

Adaptive and Gradient Boosting Machine can perform with better accuracy than Random Forest can. Extreme Gradient Boosting is created to compensate for the overfitting problem of Gradient Boosting. Thus, we can say that in general Extreme Gradient Boosting has the best accuracy amongst tree-based algorithms. Many say that Extreme Gradient Boosting wins many Machine Learning competitions. If you find this article useful, please feel free to share.

Fig. 8 Summary

About author

Connect with me here https://www.linkedin.com/in/rendy-kurnia/

Article Url - https://www.analyticsvidhya.com/blog/2021/04/distinguish-between-tree-based-machine-learning-algorithms/

rendyk