Random Forest

December 31, 2018

1 Introduction

Random Forest is a **supervised learning** algorithm which builds **multiple decision trees**. It then **merges** these decision trees and creates a more accurate and stable prediction model. Moreover, random forest also **adds additional randomness** to the model, which results in **better generalization** and hence reduces over fitting.
Each of these trees are **classification trees**.Now, each of these trees gives a classification and ***votes*** for a classification. The forest does **majority voting** on these predicted classes for all the trees and then outputs the final predicted class.

2 Growing a random forest

Each tree is grown based on the following steps:

- Sample N cases at random with replacement from the training set where N is the size of the training set.
- Feature Selection: Say there are a total of M input variables, then we pick m variables at random where $m << M$. The node is then split using the best of these m. (How do we know which is the best m?). Also note that, this value of m is held constant when the forest is growing. Each of these trees are grown to the largest possible extent with no pruning.

Size of m and its effect: Reducing m increases the error rate of the model by decreasing its strength and correlation, while the reverse happens when we increase the size of m. Thus, the optimal value of m lies somewhere in the middle. This value can be found out using OOB (Out of bag) sample.**A random forest is extremely sensitive to this OOB parameter.**

3 Features of Random Forest

- It generates an **internal unbiased estimate** of the generalization error as the forest building progresses.
  So, an obvious questions arises here: How do the random forest create an internal unbiased estimate ? Each tree is constructed using a different bootstrap sample from the original data. From this bootstrap sample, about $\frac{1}{3}^{rd}$ of the data is left out and is not used in the construction of that particular tree. Thus, an internal test set is obtained in each of these cases and at the end, using this OOB samples, we choose the class of the k^{th} using majority voting.
  Note: The proportion of times, when the j^{th} tree class did not match the truth class is defined by the OOB Error Rate Estimate.
- It gives **estimates of what variables are important** in the classification.It can handle thousands of input variables without variable deletion. Random Forest calculates feature importance based on a measure called **Gini Impurity**. *Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.* Thus, it is computed for a set of elements having J classes as follows:
  Let $i \in \{1, 2, 3, ..., J\}$ then, we can say that $I_G(p) = \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{J} p_i (1 - p_i) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} p_i^2 = 1 - \sum_{i=1}^{J} p_i^2$.
  We can also use **information gain** to calculate feature importance. Information gain is used to decide which feature to split on at each step in building the tree. Simplicity is best, so we want to keep our tree small. To do so, at each step we should choose the split that results in the purest daughter nodes. $(T) = \{I_E(p_1, p_2, ..., p_J) = -\sum_{i=1}^{J} p_i \log_2 p_i$ where $p_1, p_2, ...$ are fractions that add up to 1 and represent the percentage of each class present in the child node that results from a split in the tree.
  $\overbrace{I_G(T,a)}^{InformationGain} = \overbrace{(T)}^{Entropy(parent)} - \overbrace{(T|a)}^{WeightedSumofEntropy} = -\sum_{i=1}^{J} p_i \log_2 p_i - \sum_a p(a) \sum_{i=1}^{J} -\Pr(i|a) \log_2 \Pr(i|a)$
  Thus when training a tree, it can be computed how much each feature decreases the weighted impurity in a tree. For a forest, the impurity decrease from each feature can be averaged and the features are ranked according to this

measure.Thus with variable importance we can choose it based on it's contribution and hence create a more flexibile model.

- It has an **effective method** for estimating missing data and maintains accuracy when a large proportion of the data are missing. There are two ways with which it handles missing values:

  1. The method computes the median of all m^{th} variable that belongs to class j and replaces the remaining missing variables with this value. It is quite fast but does not yield the best results.
  2. It replaces missing values only in the training set. It begins by doing a rough and inaccurate filling in of the missing values. Then it does a forest run and computes **proximities**.This is computationally more expensive but is able to handle higher number of missing values.

     Note: *Proximities:* Since an individual tree is unpruned, the terminal nodes will contain only a small number of instances. Run all cases in the training set down the tree. If case i and case j both land in the same terminal node. increase the proximity between i and j by one. At the end of the run, the proximities are divided by twice the number of trees in the run and proximity between a case and itself set equal to one. Proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data. Thus, **Proximity is the proportion how often two data points end in the same leaf node for different trees.**

  Some other important features of random trees are as follows: It can balance error in class population unbalanced data sets. It runs efficiently on large data bases. Generated forests can be saved for future use on other data. Random Forest can also be extended to unsupervised Learning.

4 Difference between Decision Trees and Random Forest:

The fundamental difference between these two concepts is that a decision tree is built on an entire dataset, using all the features/variables of interest, whereas a random forest randomly selects observations/rows and specific features/variables to build multiple decision trees from and then averages the results. Thus a decision tree can lead to *overfitting* and thus cross-validation becomes important. But one major advantage of decision tree is easy interpret-ability, you know what variable and what value of that variable is used to split the data and predict outcome. But random forest works quite like a black-box where we have no control on the randomness which eventually turns out to be good thing.(It reduces variance!!)Accuracy increases as we increase the depth of the forest, but it eventually flattens out.

5 Random Forest: skLearn

To import: from sklearn.ensemble import RandomForestRegressor
Main parameters:

- n_estimators: this is the number of trees the algorithm building before taking maximum voting.
- max_features: The maximum number of features the random forest is allowed to try in a tree.
- min_sample_leaf: The number of leaf it can split a node in.

Note: The above parameters increases its predictive power.

- n_jobs: Number of processors it can use. Hence, we can say that it supports parallelism.
- random_state: Acts as seed, hence models results can be replicated if given same data and hyperparameters.
- oob_score: Set to True if we have OOB Error Estimation.

Note: The above parameters controls models speed.

6 Reference:

- `https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm`
- `https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd`
- `https://medium.com/machine-learning-101/chapter-5-random-forest-classifier-56dc7425c3e1`
- `https://stats.stackexchange.com/questions/285834/difference-between-random-forests-and-decision-tree`
- `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`