Trees, Bagging and Boosting

April 11, 2021

1 Decision Trees

- **Classification And Regression Tree (CART)** is an algorithm to train decision trees. CART is a greedy algorithm, it greedily searches for an optimum split at the top level, and repeat the process at each level. It is not guaranteed to arrive at an optimal solution.
- Finding optimal tree is NP-Complete problem, with complexity O(exp(m)), where m are the numbers of samples. (NP problem solution can be verified in polynomial time. Any NP problem can be reduce to NP-hard problem in polynomial time. NP-Complete problems are intersections of NP and NP-Hard.)
- CART has training complexity of O(n x m log(m)), where m training samples, with n features. At each node, it has to iterate through all the training samples classified in that node over all the features. And, inference complexity is log(m).
- CART can minimize (i) **Gini Impurity** or (ii) **Entropy**
  - **Gini Impurity**

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

  - Gini Impurity at i-th node, $p_{i,k}$ is ratio of training instance in class k over all the training instances in node i.
  - **Entropy**

$$H_i = 1 - \sum_{k=1}^{n} p_{i,k} log(p_{i,k})$$

  - As per Shannon's Information Theory, where it measures the average information content of a message.
  - Empirically, Gini impurity tends to isolate the most frequent class, while entropy tends to produce slightly more balanced trees.
- Decision Trees are non-parametric models, where numbers of parameters does not defined prior to training by making assumptions about data. But it varies based on the training dataset.
- Regularization Parameters, like maximum depth, minimum samples split, min samples leaf, min weight fraction at leaf, and max features evaluated at each node for decision split, etc.
- Decision Tree for Regressions use mean-squared-error to arrive at decision split.
- **Instability** Decision trees are very sensitive to training set rotation. One way to address this problem is to apply PCA for better orientation, and which could help decision trees with better decision boundaries.

2 Voting Classifiers

- Multiple weak classifiers are used for weak prediction, and mode of its prediction is taken as the final one. It is also the case of hard voting. Soft voting takes probability of each class into an account.
- 1000 weak classifiers of 51% accuracy each, could give 75% accuracy, and 10000 weak classifiers could give 97% accuracy. If and only if, they are independent.

3 Bagging and Pasting

- **Bagging** is a technique to come up with diverse set of classifiers. Each classifiers can be trained with different random subsets of training set deriving using bagging (short form for **bootstrap aggregating**)
- Each individual predictor has higher bias, but aggregations over multiple predictors would once again lower it to similar bias.
- Model has variance when it learns too much about the data, but bagging would prevent that, as different predictors has access to different data, and aggregation would lower variance.
- **Pasting** is when sampling is performed without replacement. Bagging ends up with slightly higher bias than pasting, but its variance is reduced.
- Out-of-Bag evaluation can be performed with bagging and pasting, and we need not to keep seperate validation set.

4 Random Forests

- It is a bagging technique, where additional randomness is introduced by only considering subset of features when splitting a node.
- **Extra-Trees** further increases randomness by using random split threshold instead of finding best possible thresholds.
- **Features Importance** It makes easy to compute the relative feature importance, i.e. how much the tree nodes using a specific feature reduce the impurity on average.

5 Boosting

- **Boosting** is an ensemble method which combine several weak learners into a strong learner.
- **AdaBoost** New predictor correct its predecessor by paying more attention to the training instances that predecessor under fitted.
  - At each iteration, the relative weight of mis-classified instances increased.
  - error at j-th predictor is computed as follow

$$\epsilon_j = \sum_{y_i \neq k_j(x_i)} w_i^{(j)} / \sum_{i=1}^{N} w_i^{(j)}$$

  - The weight of j-th predictor is computed as follow:

$$\alpha_j = \frac{1}{2} \ln \left(\frac{1 - \epsilon_j}{\epsilon_j} \right)$$

  - Final prediction is computed by taking weighted average over α_j over each predictor.
- **Gradient Boosting** It works by sequentially adding predictors to an ensemble, each one correcting its predecessor. Instead of tweaking instance weight, it tries to predict residual errors. When decision trees are used as weak predictors it is called **Gradient Tree Boosting**, or **Gradient Boosted Regression Trees**. Final prediction is the sum of predictions of all the predictors.
- **Boosting** reduces bias error, but it could have similar variance.

6 Stacking

- Instead simple aggregations, i.e. mode in voting classifiers or bagging models, weighted aggregation in AdaBoost, sum in gradient boosting, we can learn aggregation using stacking. Stacking learns final predictor, which is also called **blender** or **meta learner**.