Decision Tree And Boosting: A review

by Pengyu Sun

Decision tree algorithms are widely usd in dataming especially in recommandation systems. There are many different algorithms for building an representative tree to address both classification and regression problems. In the article, I am going to give a qucik review about the decision tree algorithms from ID3, C4.5 to XGBoost. Most decision tree inductin algorithms are using greedy algorithms for induction.

Main Point

1. Tree structure with each node denotes a subset of dataset and each edge denotes a rule that split the subset into smaller but more purity subsubsets.
2. To find the optimal tree structure is NP-Hard problem. For efficiency, using greedy algorithms.
3. How to split a node: choose a subset of attributes and split according to attributes' values.
4. Measurement: whether a split maximum the purity gain or any other objective functions. Search by sorted examples' attributes' values. Mostly divide the subset into to parts for simplicity.
5. Information Gain, Gain ratio, Gini index, miss classification rate. The intuition behind ‘greedy’ is that we can get better reuslt after any splitting step. The problem is we might be trapped by local optimal. Solved by pruning, adding regularization, resampling and resembling multiple models(bagging, rf, boosting).

Purity Method

Only for classification

ID3

ID3 is the most simple but effective algorithm for decision tree induction. It use information gain as a measurement to split a tree. ID3 do not use pruning to deal with overfitting. It use so-called pre-pruning to stop tree growing early. A threshold σ is used to decide whether to split a node.

> 1. overfitting
> 2. cannot deal with continuous attributes
> 3. information agin tend to split a node into more subsets, which makes the tree more complex and big
>   : also overfitting.

4. can not deal with correlations between attributes and can not deal with irrelavent attributes(reduncy).

C4.5

C4.5 use Gain ratio as a measurement. Besides, it can deal with continuous attributes. It sorts the samples first by the candidate attribute and split it by the already showing values from low to high and find the splitting maximum the gain ratio. For pruning, C4.5 use a objective function:

$$J = C(T) + \alpha|T|$$

where

$$C(T) = \sum_{t=1}^{|T|} N_t \sum_{i=1}^{k} (-\frac{N_i}{N_t} log \frac{N_i}{N_t})$$

, ITI is the number of leafs in a tree and k is the number of classes of the dataset. Start from a leaf, if $J' < J$, (J' is the value of objective function after deliminate all the leaves of a specific node), we delete the leaves. We do this recursively until we get to the root node or until no delimination can improve the objective function.

For both classification and Regression

CART (Classification and Regression Tree)

Cart can be used for both classification and regression problems. For classification problems, it use Gini index as a measurement of purity. Besides, it only split each tree node into to part to expand the decision tree, and each attribute used as a rule can be used again in a subspace. This difference makes it suitable for regression problems. It is just like using piecewise constant function to approximate some continuous functions. For CART, such approximation is really sketchy. We can see it as suing pixels to appriximate really world images, too. The other difference in which CART is different from ID3 and C4.5 is that it use a more complicated pruning algorithms which is efficient and generalizing well.

Objective function for regression tree.

Since one leave can only output one specific value, we hope to minimize the square error of each leave, namely:

$$J = \sum_{t=1}^{T} N_t \sum_{i=1}^{K_t} (y_i - C_t)^2$$

It is complex to get a regression tree that optimize the objective function. We still use greedy algorithms here.

1. We induct from the root node which contains all the training examples and for each node we are currently

deal with, we choose an attribute, find a split that maximize:

$$J_{node_t} = \min_{c_{tl}, c_{tr}} \sum_{i=1}^{K_l} (y_i - C_{tl})^2 + \sum_{i=1}^{k_r} (y_i - C_{tr})^2$$

We find the optimal attribute with least J_{node_t}, split the node into two.

2. Pruning the tree.

The pruning algorithm of CART is different from normal ones. It will generate a sequence of subtrees by using different αs until we only have a root node for the tree. We use validation set (or cross-validation) to choose the optimal subtree from the sequence.

1. $k = 0, T = T_0$
2. $\alpha = \infty$
3. Bottom up, calculate the loss function for each inner node n with or without leaves, namely:

   $$C_\alpha(n) = C(n) + \alpha$$

   (only one leave)

   $$C_\alpha(T_n) = C(T_n) + \alpha |T_n|$$

   calcualte $\alpha = g(n)$ when $C_\alpha(n) = C_\alpha(T_n)$:

   $$g(n) = \frac{C(n) - C(T_n)}{|T_n| - 1}$$

   $\alpha = \min(\alpha, g(n))$
4. Deliminate node n, with $g(n) = \alpha$ to a single leave. Enqueue the new subtree to the sequence.
5. $k = k + 1, \alpha_k = \alpha, T_k = T$
6. Repeat 3,,4,5 until T_k is a tree with only root and two nodes.
7. Using cross-validation to find the optimal from the sequence.

My understanding of cart pruning.

Use cross-validation to find the tree generalizes best. Add regularization to generate a tree that generalizes better. Use different α to counteract the effect of greedy search. It is a liitle similar to dynamic programming by using $g(n)$.

Ensemble Method

Weak learnable equals to strong learnable problems. So we may could combine many weak predictors to a

strong one.

$$\lim_{n->\infty} \sum_{k=(n+1)/2}^{n} C_n^k p^k (1-p)^{n-k} = 0$$

Voting Model

Bagging

Bagging is an ensemble method which will generate a set of trees and get the result of an input but voting for classification problems and by averaging for regression problems. It use bootstrap method to resample a subset of dataset and train a decision tree based on the sampled subset. It will repeat this for many times and get a group of decision trees.

Random forset

Different from bagging method, random forest not only resample the records, it also randomly choose a subset of attributes when constructing a new tree. Such processing reduces the correlation bewteen different trees and make the construction of each tree more fast.More robust to overfitting.

Additive Model

Boosting

Basic idea: after get a new weak classifier, increase the weights of samples that wasn't classified correctly ans decreas the weights of samples that was correctly classified.

Adaboost.

Adaptive weight for both samples and the classifiers. For a dataset $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ and $y_i \in \{+1, -1\}$:

1. $W_0 = \{\frac{1}{n}, \ldots, \frac{1}{n}\}$
2. train a model G_0
3. For W_m, G_m, calcualte the error e_m:

$$e_m = \sum_{i=0}^{n} w_{m,i} I(G_m(x_i), y_i)$$

4. Calculate the weight for classifier by their accuracy:

$$g_m = \frac{1}{2} log \frac{1 - e_m}{e_m}$$

5. Calculate the weight for each samples:

$$w_{m+1,i} = \frac{w_{m,i} \cdot \exp(-g_m \cdot y_i \cdot G_m(x_i))}{Z_m}$$

   Where Z_m is a normalization factor:

$$Z_m = \sum_{i=1}^{n} w_{m,i} \cdot \exp(-g_m \cdot y_i \cdot G_m(x_i))$$

6. We come back to step 3 until we get to the limitation of number of weak classifiers we set. Namely, M classifiers.
7. $f(x) = \sum_{m=1}^{M} g_m G_m(x)$
8. $C(x) = sign(f(x))$

> 1. Adaboost is an additive model and can be resolved by forward stagewise algorithm. The loss function is
>
> $$\exp(-yf(x))$$

GBDT (Gradient Boosting Decision Tree)

GBDT is also an additive model with loss function defined as mean square error. Each iteration, we train a tree that fitting the residual from the target to the previous classifier. We shrinkage the effect of the new tree we get by multiply a $\eta << 1$ to the result. This make a smooth way to th optimal result and thus prevent divergence and overfitting.

XGboost

Developed by Tianqi Chen, XGboost is a dramatically boosting tree algorithm that is scalable and outperform many other algorithms in data mining. It use Newton Method to optimize the objective function with regularization.

$$J = L(y_i, f_{m-1}(x_i) + a_m G_m(x_i)) + \alpha |T| + \sum_{t=1}^{|T|} |W_t^2|$$

The XGboost is a little difficult for me to totally understand and I have not yet finished reading the paper. I will update this part for more details generally.