Decision Trees and Ensemble Method

Rao Sihang 15220162202309

Decision Tree

Decision Tree is one of the most popular machine learning algorithms ,which is highly interpretable like the procedure of human thinking, and can capture the nonlinear relationship among variables. A decision tree is a tree where each node represents a feature, each link represents a decision and each external node (leaf) represents an outcome (catrgorical or continuous).
Each internal node is chosen by some specific measurement in different algorithms, for example, in CART (Classification and Regression Trees),usually, each node or features is picked based on Gini index. In ID3 (Iterative Dichotomiser 3), Entropy function and Information gain are more popular.

Entropy function and Information gain in ID3

Assume there is a binary classification problem with a vector of feartures (X), and we need to build a tree to do classification. In order to build this classification tree, we need to pick a feature that best classifies the training data as the first node. And repeat the same procedure to gain the next node. How the judge the perfomance of each feature? The answer is to use the feature with the highest information gain.

Before introduing the definition of IG, the introduction to entropy function is necessary. Entropy H(S) is a measure of the amout of uncertainty in the dataset D:

$$H(D) = \sum_{y \in C_i} -p(C_i) log_2 p(C_i)$$

D-The current dataset for which entropy is being calculated (differ in each node).

C_i-Set of classes in D

$p(C_i)$- The propotion of the number of elements in classC_i to the number of elements in whole D

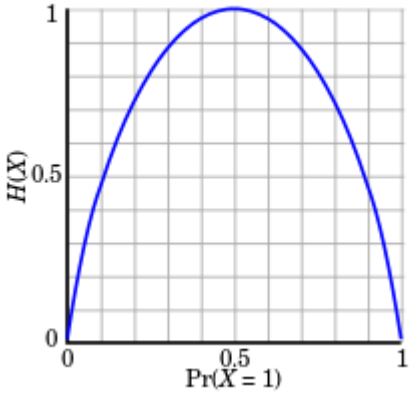
For example, in a dataset D, half elements belong to class A, and half elements belong to class B, so the entropy:

$$H(D) = -(\frac{1}{2} log_2 \frac{1}{2} + \frac{1}{2} log_2 \frac{1}{2}) = 1$$

However, if all the elements in S belong to class A, which means this classifier performs well and classifies elements purely, in this case,

$$H(D) = -(\frac{2}{2} log_2 \frac{2}{2} + \frac{0}{2} log_2 \frac{0}{2}) = 0$$

So the smaller entropy is ,the better classification is. In ID3, the feature with smallest entropy should be chosen to split the dataset D.



Information Gain (A) is the measure of the difference in entropy (uncertainty) from before to after the dataset S is split on a feature A:

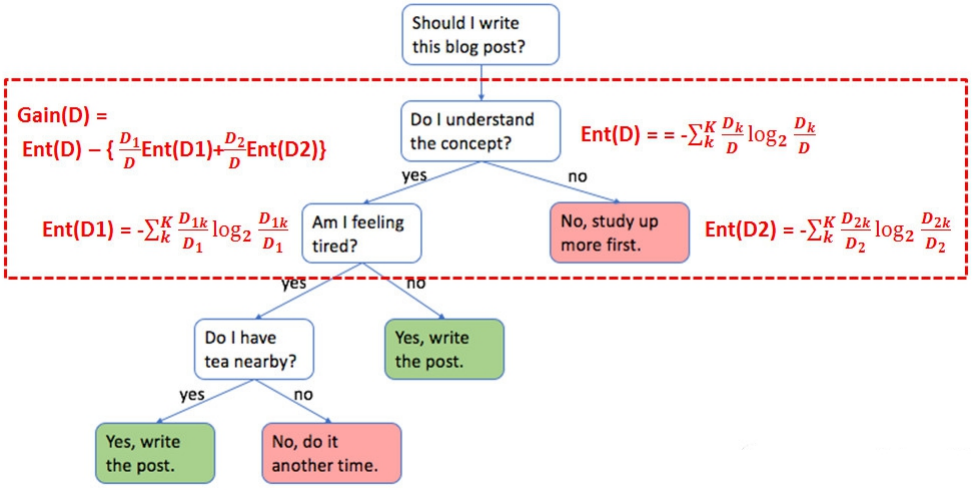
$$IG(D, A) = H(D) - H(D|A)$$

$$H(D|A) = \sum_{D_i} p(D_i)H(D_i)$$

D_i-The subsets created from spiltting D based on fearture A

$p(D_i)$-The propotion of the number of elements in subset D_i. $\sum_{D_i} p(D_i) = 1$

IG can be calculated for each feature and we pick the feature with largest one (the largest improvement from the split).



Steps (According to the graph above):

1. Calculate entropy Ent(D)
2. For each feature:

   - calculate entropy for all subset $\{Ent(D_1), Ent(D_2)\}$
   - calculate weighted-entropy after split $\{\frac{D_1}{D}Ent(D_1) + \frac{D_2}{D}Ent(D_2)\}$
   - calculate information gain Gain(D).

3. Pick the feature with largest IG
4. Repeat the procedures to build the whole tree.

Gain Ratio in C4.5

However, there is a shortcoming in IG, a preference on those features with more classes, which leads to a larger reduction on uncertainty. In order to avoid this disadvantage, Gain Ratio is created by Ross Quinlan in 1993. Gain ratio uses split information to punish those features with relatively many classes.

$$GainRatio(D, A) = \frac{IG(D, A)}{SplitInf(D, A)}$$

$$SplitInf(D, A) = -\sum_{D_i} \frac{D_i}{D} log_2 \frac{D_i}{D}$$

Gini index in CART

CART algorithm was proposed in 1984, which is applied more than ID3 and C4.5. For classification tree, Gini index is a measure of node impurity:

For a dataset D:

$$Gini(D) = 1 - \sum_{C_i} (\frac{C_i}{D})^2$$

When dataset D is split into two subset based on a specific value of a specific feature (A) (in ID3, each feature can only be used once, because we split the data based on feature, but in CART, each feature can be used for many times and we use the value of features as the node.):

$$Gini(D, A) = \frac{D_1}{D} Gini(D_1) + \frac{D_2}{D} Gini(D_2)$$

GBDT

Boosting Tree is an ensemble method combining multiple learners to improve the quality of fit of each base learner. In boosting tree, for a dataset$\{(x_1,y_1), (x_2,y_2)...\}$, you can use a function $F(x)$ to fit this set, $F(x) = argminL(y, F(x)) = argmin\frac{1}{n}\sum\limits_{i}(y_i - F(x_i))$. How to further improve this fitting model without changing the initial one? A feasible method is to establish a new model that add an estimator $h(x)$.$F_2(x) = F(x) + h(x)$,when loss function (L2) is

$$L(y, F_2(x)) = L(y, F(x) + h(x)) = (y - F(x) - h(x))^2 = (r - h(x))^2$$

Therefore, gradient boosting will fit $h(x)$ to the residual $r = y - F(x)$.
For general loss function, it is not easy to optimize, and Friedman proposed **Gradient Boosting**, which uses the negative gradient of loss function as the approximation of residual:

$$r = -\frac{\partial(L(y, F(x))}{\partial F(x)}$$

GBDT(Gradient Boosting Decision Tree) is the combination of Gradient boosting and Decision tree.

1. Initialize model$f_0(x) = argminL(x, f_0(x))$
2. For m=1,2,...,M,$r_m = -\frac{\partial(L(y,F(x))}{\partial F(x)}$

- Use(x, r_m)as the new data for next tree $f_m(x)$ and the external node(leaf)
- For each leaf i, calculate the best prediction value,$\hat{R}_{im} = argminL(y, f_{m-1}(x) + \hat{R}$
- Update the model:$f_m(x) = f_{m-1}(x) + \sum \hat{R}_{im} I(x \in R_{im})$

3. The final model is $F(x) = f_M(x)$

XGBoost

Compared to general GBDT, XGBoost is more efficient with several advantages:

1. Adds a regularization term into the loss function to avoid overfitting.

$$L = \sum (y - \hat{y}) + \sum_k \Omega(f_k)$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2$$

k-Number of regression trees
T-Number of leaf nodes
w-Value of each node

2. Can handle with sparse data
3. Use the second order deviation of loss function
4. Parallel and distributed computing makes learning faster which enables quicker model exploration.