Chapter 4 Decision Tree

Jinghao.Zhao

Abstract

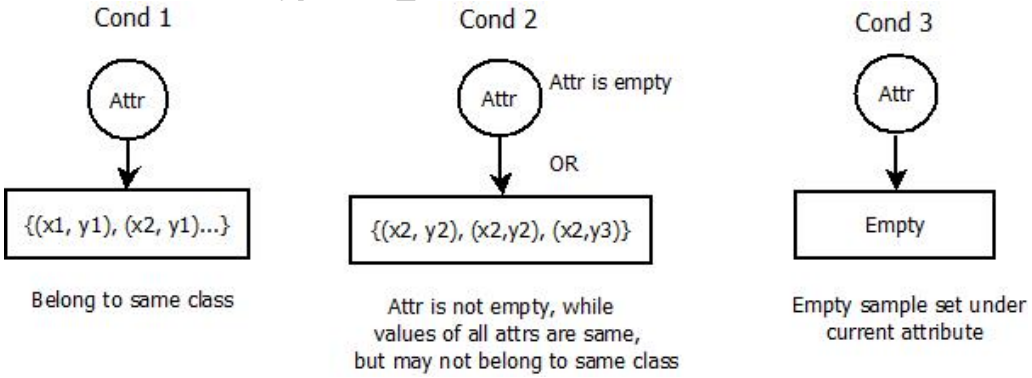
This is the notes for decision tree, section 4.1 to 4.4, including formula and key points.

1 Decision Tree

Decision tree is one of the most simple classifier. There are one root node, several inner node of attribute test which includes samples divided by attributes and several leaf nodes represents the classification result. In decision tree algorithm, there are three conditions to stop the recursion:

1. Samples of current node are belongs to same class;
2. Current attribute set is empty(all attributes have been used.) or all samples of current node have same value of all attributes(although they may not belong to same class).
3. There are no samples under current node.

Those three conditions are represented as figure shows



In condition 2, current node will be treated as a leaf node and its class will be the largest proportion of the samples. In condition 3, it will choose the largest proportion of class in the samples of its father node.

2 Devision Principle

Selecting the most optimal attribute means we want to make sample of node as same as possible belong to same class(the highest purity). There are three index to represent the purity.

2.1 Information gain

We define information entropy

$$Ent(D) = -\sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k, \tag{1}$$

where D is dataset and p_k is the proportion of class k. Then $\boldsymbol{purity} \propto \frac{1}{Ent(D)}$. If attribute a has V different values $\{a^1, a^2, ..., a^v\}$, then node a will have v branches. Support D^v is the sample set that the value of a is a^v, the information gain is

$$Gain(D, a) = Ent(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} Ent(D^v), \tag{2}$$

and we can choose the attribute which has the largest information gain(ID3 algorithm). For discrete attributes, they will not be reused in the division.

2.2 Gain ratio

Information gain will prefer the class with more value. For eg, if we have n samples and one of the attributes have n different values, this attribute will be the best classification basis because it will every sample a leaf node, but such a tree have no ability of generalization. We improve it by using gain ratio(C4.5 algorithm).

$$Gain\_ratio(D, a) = \frac{Gain(D, a)}{IV(a)}, \tag{3}$$

where

$$IV(a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}, \tag{4}$$

is the intrinsic value(固有值) of D. But gain ratio has preference on attribute which has less values, C4.5 will select attributes whose information gain are larger than the mean level, then choose the largest gain ratio one.

2.3 Gini index

CART(Classification and Regression Tree) used gini index to represent purity of dataset D.

$$Gini(D) = \sum_{k=1}^{\mathcal{Y}} \sum_{k' \neq k} p_k p_{k'} = 1 - (\sum_{k=1}^{\mathcal{Y}} p_k^2). \tag{5}$$

Gini index means the probability when randomly select two **different** samples from D. Then the gini index of attribute a can be represented as

$$Gini\_index(D,a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} Gini(D^v) \tag{6}$$

The CART will choose the smallest gini index attribute.

3 Pruning

Pruning is used to reduce the risk of overfitting. There are two methods, prepruning and postpruning.

3.1 Prepruning

Prepruning is pruning the tree during the process of generating a tree. Before the division based on an attribute, it first calculate the accuracy to decide if division or not in following two direction.

1. Not division, use condition 2 to make current node be a leaf.
2. Do division, use information gain or ratio to divide current node.

Then calculate the accuracy by **test set** and choose the better decision.

3.2 Postpruning

Postpruning is pruning after built the tree. For each layer of leaf nodes, calculate the accuracy of current tree and the tree after pruning all brunches of its father node. Suppose node A has three child nodes B, C and D. Postpruning will calculate the accuracy on the whole tree also on the node A without divided into B, C and D by using **test set**.

4 Concrete Attributes

If an attribute is concrete value, in decision tree, we try to find a middle value that can divide this attribute into two parts, larger or smaller than the middle value. For attribute a who has values of $\{a_1, a_2, ..., a_n\}$, we build candidate division set

$$T_a = \{\frac{a^i + a^{i+1}}{2} | 1 \le i \le n-1\} \tag{7}$$

Then for each element in T_a, we calculate the information gain

$$Gain(D,a) = \max_{t \in T_a} G(D,a,t) = \max_{t \in T_a} (Ent(D) - \sum_{\lambda \in \{-,+\}} \frac{|D_t^\lambda|}{|D|} Ent(D_t^\lambda)) \tag{8}$$

$\sum \frac{|D_t^\lambda|}{|D|} Ent(D_t^\lambda)$表示基于t二分后，正负类(二分类问题)的权重与信息熵乘积之和.

5 Missing value

Missing value will cause two problems in both

1. Choosing attribute to divided samples;
2. Arranging samples with missing value on a certain attribute.

For the first problem, we only concerned samples **without missing value on current attribute** when calculate gain or ratio. For the second problem, we divided those sample into each branch, then give them a weight to represent the probability for each sample $\mathbf{x}$ to be divided into every child nodes(赋予权重后，下次选择结点时会考虑权重).

When concerned an attribute $\mathbf{a}$, let $\tilde{D}$ represent samples whose value of $\mathbf{a}$ is not none, $\tilde{D}^v$ represent samples whose value of $\mathbf{a}$ is a^v and $\tilde{D}_k$ means sample belongs to class k in $\tilde{D}$. Then we assign weights ω_x(initial value is 1) for each sample on $\mathbf{a}$ and get 3 parameters.

$$\begin{aligned} \rho &= \frac{\sum_{x\in\tilde{D}}\omega_x}{\sum_{x\in D}\omega_x} \\ \tilde{p}_k &= \frac{\sum_{x\in\tilde{D}_k}\omega_x}{\sum_{x\in\tilde{D}}\omega_x} \\ \tilde{r}_v &= \frac{\sum_{x\in\tilde{D}^v}\omega_x}{\sum_{x\in\tilde{D}}\omega_x} \end{aligned} \tag{9}$$

where ρ represent the proportion for samples whose $\mathbf{a}$ is not none, $\tilde{p}_k$ means proportion for class k in $\tilde{D}$, $\tilde{r}_v$ means proportion of samples whose $\mathbf{a}$ is a^v in $\tilde{D}$. Then the information gain with missing value is

$$\begin{aligned} Gain(D,a) = \rho \times Gain(\tilde{D},a) = \rho \times (Ent(\tilde{D}) - \sum_{v=1}^{V} Ent(\tilde{D}^v)), \\ where\ Ent(\tilde{D}) = -\sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k \log_2 \tilde{p}_k \end{aligned} \tag{10}$$

(10) will solve problem 1 by calculate new information gain, and for problem 2, the process is

1. If a sample is not none on $\mathbf{a}$, it can be correctly divided into a node, and the weight of this sample will not be changed.
2. If a sample is none on $\mathbf{a}$, first it will be divided into all nodes, then update the weight ω_x by $\tilde{r}_v \cdot \omega_x$.

The weight will be update in each recursion until the whole three is built.