Decision Tree

date: 20261114
status: #finished
tags: #machine_learning #decision_tree #cart

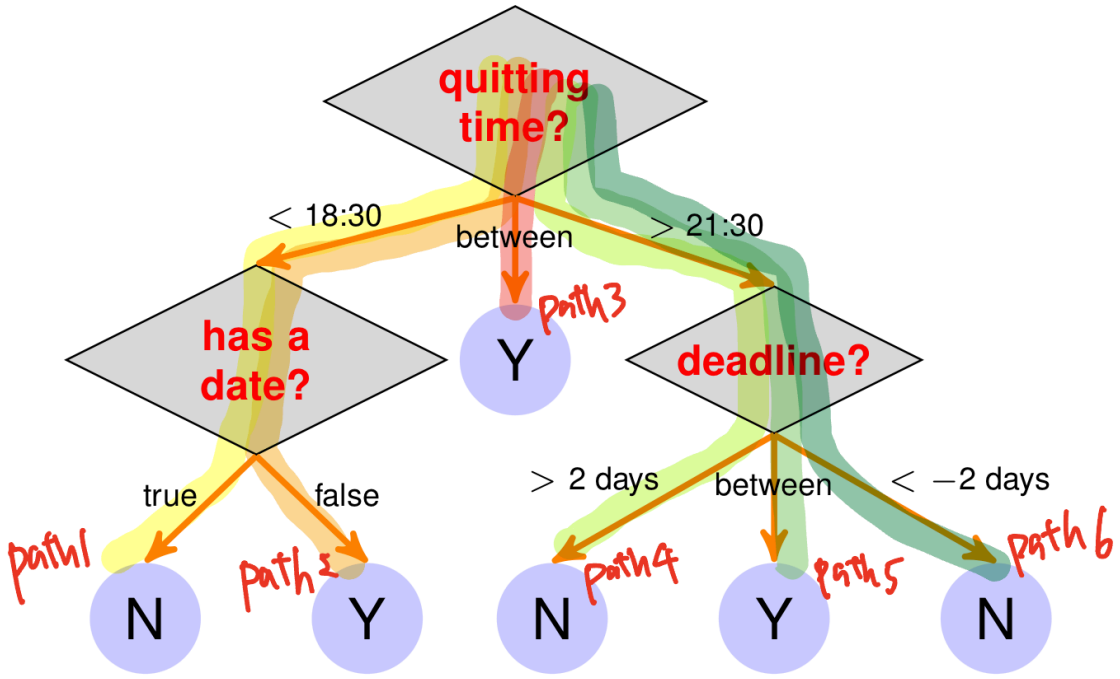
---

0. Introduction

(i) How the decision tree make the prediction?

The core idea of decision tree is splitting samples into subgroups based on X. For example, a split can be whether the age is greater than 20, which yields two subgroups: > 20 and ≤ 20. For a numeric feature, each split has two parameters: which feature to use and at what value of that feature. For a categorical feature, the second parameter is a group of categories instead of a value. For example, a split can be whether the color is red or blue, which yields two subgroups: red or blue, and all other colors.

The splits generates the tree structure and each sample will be classified to different leaves. Then, decision tree do the regression or classification based on which leaf does the underlying sample live. For regression, the predicted outcome of that sample is obtained by averaging over outcomes of samples in that leaf, and for the classification, the majority class among samples, belonging to that leaf is used. Besides, the share of each class can be viewed as the probability of being in each class in that leaf.



The above illustration comes from the course materials provided by Professor Hsuan-Tien, Lin.Each leaf of a decision tree corresponds to a path, so the number of leaves will be equal to the number of paths. Given a sample, x_i, its predicted value $\hat{y}_i$ is obtained through:

$$\hat{y}_i = h(x_i) = \sum_{t=1}^{T} \mathbf{1}\{x_i \text{ on path } t\} \cdot h_t(x_i)$$

where T is the number of leaves, and $h_t(x_i)$ is the average outcome of the corresponding leaf in regression and is the majority class in classification.

Therefore, the parameters of a decision tree, at the highest view, is $\{h_t\}_t$, which is determined by a set of features, and corresponding feature values/categories, and each (feature, value/category) pair corresponds to a node of a decision tree. For example, in the above example, $h_1(x_i)$ is determined by $(\text{quitting time}, <18{:}30)$ and $(\text{has a date}, \text{true})$.

(ii) How to split the numeric features?

Numeric features has infinite possible values, and one might be confused about how to determine the split value c? We should notice that the possible splitting at a numerical feature is finite. For example for a node decide to split at the age, and there are three samples, which have age 18, 30, and 40, respectively, and c only needs to be $\frac{18+30}{2} = 24$ or $\frac{30+40}{2} = 35$, because any other value between two neighbouring ages yields the same two subgroups.

(iii) What are the differences among variants of decision trees?

There are many variants of decision trees, e.g., ID3, C4.5, and CART, and the differences lies in how the score of the split is evaluated, how many branches are allowed at each split, and so on. CART stands for **c**lassification **a**nd **r**egression **t**ree, and it's the most important one so we will focus on it in this note. CART decides splits based on Gini impurity (classification) and variance (regression), and it create two branches at each split.

---

1. Splitting Score

(i) The Intuition for Perfect Decision Tree for Prediction

As mentioned the predicted outcome from a decision tree of a sample depends on which leaf it ends up, and a perfect decision tree for prediction should has identical outcomes among sample in the same leaf. Therefore, a good split is to lower the dissimilarity in outcome among leaf samples. For example, if we want to predict whether a student will enroll to the course, and if we find out that only students with age greater than 20 will enroll, then we can split at age of 20. Then, we have two leaves: one with all samples enrolled and the other with samples not enrolled, which will yield 0 prediction error.

To measure the dissimilarity in outcome of samples, CART adopt Gini impurity for classification task, and sample variance for regression task. It's intuitive that the higher the sample variance is, the higher the dissimilarity of the samples are, and we will introduce the Gini Impurity in the next section.

Moreover, as mentioned, a sample's predicted value depends on which leaf the sample sit it, and for the regression task, we utilize the sample mean within a node as the predicted values for all samples in that leaf. If we define the prediction error as the mean square error, then the mean square error in a node is equal to the sample variance of that node, which is the dissimilarity measure. Therefore, as the tree grows, the prediction error on the training samples never increases.

(ii) Gini Impurity

For classification task, each split at a tree node, which is a leaf during splitting, needs to decide at what feature and value/category of that feature should the

two branches be created. The score of the split is defined as the change in Gini impurity.

Definition (Gini Impurity)

The Gini impurity of a node, G, is defined as

$$G = 1 - \sum_{k=1}^{K} p_k^2$$

where K is the number of classes, and p_k is the share of samples with class k in the node.

The intuition of Gini Impurity is that if we draw two samples with replacement from a node, the probability of these two samples with different classes is $1 - \sum_k p_k^2$. It can be used as a measure of dissimilarity of classes among samples in that node because the higher the probability for these two draws with distinct classes is, the higher the dissimilarity is.

(iii) The splitting score is based on the decrease in dissimilarity

So far we know the goal of splitting is to decrease the dissimilarity in outcomes among leaf samples. How the score of splitting is computed and can be used to determine optimal split will be stated here.

Let say we split at feature j and at value/category c, and the node at which we split contains N_{parent} samples. The node is splitted into two branches, resulting in two children with N_L and N_R samples, respectively. The change in Gini impurity of this split is defined as

$$\Delta G(j, c) = G_{\text{parent}} - \left(\frac{N_L}{N_{\text{parent}}} G_L + \frac{N_R}{N_{\text{parent}}} G_R \right) \geq 0$$

where $G_{\text{parent}}, G_L, \text{and}, G_R$ are the Gini impurity of the node, left child, and right child, respectively. The change measures how much the impurity has dropped from the parent to its two children, and the optimal j and c at the node is determined by maximizing the change

$$j^*, c^* = \arg\max_{j,c} \Delta G(j, c).$$

For regression task, we just need to replace the Gini impurity by the sample variance.

(iv) Class Weights

Class imbalance is a known problem in classification task, and one way to overcome it is to increase the weight for minority class. Originally, the (i) Gini Impurity, (ii) change in dissimilarity at a split, (iii) majority class evaluation for final prediction are all computed with the assumption that each class has equal weight, 1 . Suppose we now want to set the weight vector $w \in \mathbb{R}^K$ with positive entries, where K is the total number of classes, the Gini Impurity in a node l will be changed to

$$G_l = 1 - \sum_k \tilde{p}_{l,k}^2$$

where $\tilde{p}_{l,k}$ is the share of class k's total weights over all weights in the node:

$$\tilde{p}_{l,k} = \frac{\sum_{i \in S_l} \mathbf{1}\{y_i = k\} \cdot w_k}{\sum_{i \in S_l} w_{y_i}} \quad \text{where} \quad S_l = \{i : \text{sample } i \text{ is in node } l\}.$$

Without setting class weights, $w_k = 1$ for all k so

$$p_{l,k} = \frac{\sum_{i \in S_l} \mathbf{1}\{y_i = k\} \cdot 1}{\sum_{i \in S_l} 1}.$$

Besides, the sample counts $N_{\text{parent}}, N_L, \text{and}, N_R$ will be the sums of weights in each node, and the majority class at node l will be $\arg\max_k \tilde{p}_{l,k}$ instead of $\arg\max_k p_k$.

(v) Prediction Error and the Dissimilarity Measure

In regression task, the prediction error, the mean square error, is equal to the dissimilarity measure, the sample variance. However, in the original CART design, the prediction error in the classification task is defined as the 0/1 error, and with the majority class prediction rule, it will gives the share of samples that is not in the majority class, which doesn't equal to the Gini impurity. Nevertheless, in the scikit-learn implementation, the prediction error is defined as the Gini impurity by default.

---

2. Early Stopping and Pruning

(i) When to stop splitting?

A node in CART stops splitting if one the following three cases happens.

1. The node has only one sample
2. All samples in the node have identical outcomes
3. All samples in the node have identical features but outcomes are not the same

If CART stop splitting at these conditions, we say that CART is fully grown, and its leaves are pure except in the third case.

(ii) Early Stopping

We can set thresholds based on properties of a decision tree to let it early stop, and not be fully grown. Typically thresholds include

1. Maximum depth of the tree
2. Minimum samples for a node to split
3. Minimum samples for a node to be a leaf
4. Minimum change in variance/Gini Impurity required for a split

However, for the original CART design, it first lets the tree to fully grow and prune it, picking the tree that has the lowest penalized error:

$$\mathrm{Err}(T) + \alpha|T|$$

where $\mathrm{Err}(T)$ is the prediction error of the decision tree and $|T|$ is the number of leaves of T, and $\alpha \geq 0$ is the penalty for each leaf.

In practice, CART is rarely used alone and the tree ensemble method is adopted. The 4 early stop parameters and α can be used as hyperparameters in tree ensemble, but they are rarely used and if we want an early stop condition, the maximum depth and the minimum samples for a leaf are two common choices.