Decision Tree

Jack Yang

October 28, 2020

Outline

Decision Tree

A decision tree is a hierarchically organized structure, with each node splitting the training set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ into pieces based on value of a feature.

- This is equivalent to make a partition of $\mathbb{R}^d$ into k disjoint feature subspaces $\{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k\}$, with each $\mathcal{R}_i \in \mathbb{R}^d$.
- On each feature subspace $\mathcal{R}_i$, the same decision/prediction is made for all $x \in \mathcal{R}_i$.

Components

- **Node** is comprised of a sample of data and a decision rule.
- **Parent** of a node is the immediate predecessor node.
- **Child** of a node are the immediate successors of the node.
- **Root node** is the top node of the tree; the only node without parents.
- **Leaf nodes** are nodes which do not have children.

Entropy

Let S be a set of data in a node, $c = 1, 2, \ldots, C$ are labels, we define **entropy**

$$H(S) = -\sum_{c=1}^{C} p(c) \log_2 p(c)$$

where $p(c)$ is the proportion of the data belong to class c.

- $H(S) = 0$ if all samples are in the same class.
- $H(S)$ is large if $p(1) = p(2) = \cdots = p(C)$.

Information Gain

Consider the split $S \to S_1, S_2$.

Then, the averaged entropy of a split is

$$\frac{|S_1|}{|S|} H(S_1) + \frac{|S_2|}{|S|} H(S_2).$$

We define the **information gain** as a measurement of how good is a split. In this case,

$$H(S) - \left(\frac{|S_1|}{|S|} H(S_1) + \frac{|S_2|}{|S|} H(S_2) \right).$$

Split the Node

Given the current node, how to find a **best split**?

Goal: We want nodes as pure as possible, that is,

- We want to reduce the entropy as much as possible
- We want to maximize the difference between the entropy of the parent node and the expected entropy of the children

Result: We want to maximize the **information gain**.

For n samples and d features, we need $\mathcal{O}(nd)$ time. Training is slow, but prediction is fast.

Construction of Regression Tree

We assign a real number for each leaf node, usually the average values for each leaf.

Goal: Minimize the residual sum of squares (RSS)

$$\sum_{k=1}^{k} \sum_{i \in \mathcal{R}_k} (y_i - y_{\hat{\mathcal{R}}_k})^2$$

where $y_{\hat{\mathcal{R}}_k}$ is the mean response for the training observations within the kth subspace.
Choose a predictor X_k and a cutpoint s that minimizes the RSS for the resulting tree

$$R_1 = \{X : X_k < s\} \qquad R_2 = \{X : X_k \geq s\}$$

$$\text{RSS} = \sum_{x_i \in R_1} (y_i - y_{\hat{R}_1})^2 + \sum_{x_i \in R_2} (y_i - y_{\hat{R}_2})^2$$

Construction of Regression Tree

The construction is computationally infeasible to consider every possible partition of the feature space into k boxes.

Thus, we take a top-down, greedy approach called **recursive binary splitting**.

- It begins at the top of the tree (all observations below to a single region) and then successively splits the predictor space.
- Each split is indicated via two new branches further down on the tree.
- At each step of the tree building process, the best split is made at that particular split.

Pruning

Recursive binary splitting may be complex and overfitting, which we will get a higher testing error.

Goal: Obtain subtree that gives lowest test error rate

Tradeoff: Using a smaller tree will cause lower variance but some bias

Approach: Given a subtree, we can estimate the test error rate using cross-validation, but this will take a long time. Thus, we need a way to select a small set of subtrees to consider.

Tree vs Linear Model

- Linear Regression

$$Y = \beta_0 + \sum_{i=1}^{n} X_i \beta_i$$

- Tree

$$Y = \sum_{j=1}^{k} c_j \cdot 1_{X \in \mathcal{R}_j}$$

Regressions outperform trees if linear structure. Trees are easier to interpret and useful when complex non-linear structure.

Aggregating several trees can improve predictability: Random Forest (next time).