Classification and Regression Trees

Wei Li

Syracuse University

Spring 2024

OVERVIEW

CART (Classification and Regression Tree)

Regression trees

Classification trees

Related issues about trees

CART (Classification and Regression Tree)

Regression trees

- p inputs and a response
- training data is $(x_i, y_i)_{i=1}^n$, $x_i = (x_{i1}, \ldots, x_{ip})$.

The CART model is

$$f_{tree}(x) = \sum_{m=1}^{M} c_m 1(x \in R_m)$$

where $\mathcal{R} = \{R_1, \ldots, R_M\}$ is a partition of $\mathbb{R}^p$.

Under the sum of squares $\sum_i (y_i - f(x_i))^2$, the fitted f is given by

$$\hat{f}(x) = \sum_{m=1}^{M} \hat{c}_m 1(x \in R_m),$$

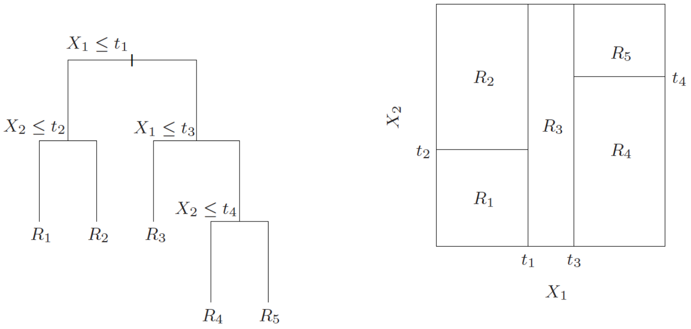
$$\hat{c}_m = \text{avg}(y_i \mid x_i \in R_m)$$

Challenge: (assuming a specific form for c_m 's) find M and the regions $R_1, \ldots, R_M$ s.t. $\hat{f} \approx f$

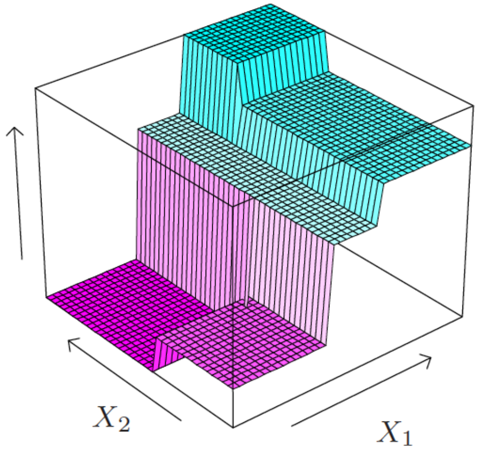
The partition that minimizes the sum-of-square training error

- not feasible to find globally
  - $\sum_{m=1}^{M} \sum_{i \in R_m} (y_i - \hat{f}(x_i))$
- locally can be found in a greedy fashion.

The partition often is achieved by recursive binary partition which corresponds to some tree structure.



ESL Fig 9.2.



ESL Fig 9.2.

Some definition of a tree and related concepts:

- T : a collection of nodes $\{t\}$ and splits $\{s\}$.
- root node
- terminal node (leaf node): each leaf node is assigned a parameter
- parent node
- child node of t
  - left node $(t_1) \rightarrow X$-region $R_{t,1}$
  - right node $(t_2) \rightarrow X$-region $R_{t,2}$
- The size of a tree: $|T|$ (the number of terminal nodes of T)

To grow a tree

Three elements:

1. Split process
   - choose splitting variables and split points
   - goodness of split criterion needed to evaluate any split s of any node t.
2. Partition process
   - partition the data into two resulting regions and repeat the splitting process on each region
   - declare a node is terminal or to continue splitting it; need a stop-splitting rule
3. Pruning Process
   - collapse some branches back.

Split process

Each split depends on the values of only one unique variable X_j:

- Select the predictor to split upon
- Select the cutoff point using the selected predictor

If X_j is continuous or ordinal, consider for some s, the pair of half-planes

$$R_1(j,s) = \{X \mid X_j \leq s\} \qquad R_2(j,s) = \{X \mid X_j > s\}$$

Determine the best pair (j,s):

- $j \in \{1, 2, \ldots, p\}$
- feasible for a given j, scan the n values for s

- If X_j is unordered categorical taking values $\{1, \dots, L\}$, the splitting regions are

$$\{X_j \in A\} \qquad \{X_j \in A^c\}$$

  A is a subset of $\{1, \dots, L\}$ (as many as $2^{L-1} - 1$ possible partitions).
- Alternative way is to expanded it into L binary indicators variables of their levels. Then the split becomes obvious by turning to 0 or 1 (as in `scikit-learn`).

Greedy algorithm

How to determine such a pair of (j, s)?

- $R_1(j,s) = \{X \mid X_j \leq s\}$, $R_2(j,s) = \{X \mid X_j > s\}$
- Choose (j, s) to minimize for observations $\{(x_i, y_i)\}_{i=1}^n$

$$\min_{c_1} \sum_i (y_i - c_1)^2 \, 1(x_i \in R_1(j,s)) + \min_{c_2} \sum_i (y_i - c_2)^2 \, 1(x_i \in R_2(j,s))$$

  - For a fixed (j, s) the (inner) minimum occurs when

$$\hat{c}_1 = \frac{1}{|R_1(j,s)|} \sum_i y_i 1(x_i \in R_1(j,s))$$
$$\hat{c}_2 = \frac{1}{|R_2(j,s)|} \sum_i y_i 1(x_i \in R_2(j,s))$$

- Once the best split is determined, the data split into two regions.
- Continue with next node...

Loss: impurity measure

More generally, at the m-th node, let R_m denote the X-region in the node, the loss (aka **impurity measure**) is given by

$$Q(R_m) = |R_m|^{-1} \sum_i (y_i - \hat{c}(R_m))^2 \, 1(x_i \in R_m)$$
$$\hat{c}(R_m) = |R_m|^{-1} \sum_i y_i 1(x_i \in R_m)$$

Considering a split for the m-node, let $R_{m,1}(j,s)$ and $R_{m,2}(j,s)$ denote the left and right child partition of R_m, the resulting loss is

$$Q(R_{m,1}(j,s)) + Q(R_{m,2}(j,s))$$

The general principle is choose (j,s) that yields the least loss, i.e.,

$$\min_{j,s} \{|R_{m,1}(j,s)| \times Q(R_{m,1}(j,s)) + |R_{m,2}(j,s)| \times Q(R_{m,2}(j,s))\}$$

Stopping

Stopping rule?

We repeat the binary partition recursively until a tree is large enough.

- A very large tree might overfit the data
- A small tree might not capture the important structure

The optimal tree size should be adaptively chosen from the data.

When should we stop the partition? One possible approach is to split tree nodes ONLY IF the decrease in loss (impurity) due to splits exceeds some threshold.

Drawback: A seemingly worthless split might lead to a very good split below it (short-sighted).

A preferred approach is to grow a large tree stopping the split process when some minimum node size is reached. Then apply the so-called pruning to prune back the tree.

Pruning process

Suppose we have grown a large tree T_0. A subtree of T is any tree that can be obtained by collapsing any number of the internal (non-terminating) nodes of T.

weakest link pruning procedure:

- successively collapse the internal node that produces the smallest per-node increase in $\sum_{m=1}^{|T|} n_m Q_m(T)$
    - $Q_m(T)$ is the loss measure of node m.
- continue until we produce the single-node (root) tree.

This gives a finite sequence of subtrees.

Cost-complexity of a tree

But how to characterize the pruning?

For each subtree T of T_0, a measure of its **cost complexity** is given by

$$C_\alpha(T) = \sum_{m=1}^{|T|} n_m Q_m(T) + \alpha|T|$$

where m 's run over all the terminal nodes in T, and α governs a tradeoff between tree size $|T|$ and its goodness of data.

For any fixed $\alpha > 0$, we can find a unique subtree T_α that minimizes $C_\alpha(T)$.

- Large α results in smaller trees;
- Small α results in large trees ($T_\alpha \to T_0$ as $\alpha \to 0$).

Cost-complexity pruning

The weakest link pruning procedure effectively traces the sequence of subtrees that minimize the so-called complexity as α increases.

This “optimal” subtree T_α can be found through the weakest linked prune procedure, as it can be shown that the sequence produced by the procedure contains the T_α as α varies.

This procedure is also called **Cost-complexity pruning**.

Selection of α

The α can be selected by CV.

Algorithm:

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:
   - Repeat Steps 1 and 2 on all but the k th fold of the training data.
   - Evaluate the mean squared prediction error on the data in the left-out k th fold, as a function of α.
4. Average the results from Step 3 for each value of α, and pick α to minimize the average error.
5. Return the subtree from Step 2 that corresponds to the chosen value of α.

Classification trees

Suppose outcome has $1, 2, \ldots, K$ values. In the node m, represented by a region R_m

$$\hat{p}_{mk} = \Pr(y = k \mid R_m) = \frac{1}{|R_m|} \sum_{x_i \in R_m} 1\left(y_i = k\right)$$

the proportion of class k observations in node m. We classify the observations in node m to class

$$\hat{k}_m := \arg\max_{k=1,\cdots,K} \hat{p}_{mk}$$

by the majority class in node m.

Impurity function (loss)

To determine the (j, s) when splitting, choose (j, s) which reduces the node impurity the most. A node is more "pure" if one class dominates the node than if multiple classes equally present in the node.

An impurity function is a function Q defined on the set

$$\left\{(p_1, \ldots, p_K) : p_k \geq 0, k = 1, \ldots, K, \sum_{k=1}^{K} p_k = 1\right\}$$

satisfying

- Q is a maximum only at the points $(1/K, \ldots, 1/K)$ (most impure case).
- Q gets minimum only at $(1, 0, \cdots, 0), (0, 1, \cdots, 0), \cdots, (0, 0, \cdots, 1)$ (most pure case).
- Q is a symmetric function of $(p_1, \ldots, p_K)$.

Examples, suppose on the m-th node R_m, choice of $Q_m(\hat{p}_1, \ldots, \hat{p}_K)$:

1. Misclassification error:

$$\frac{1}{|R_m|} \sum_{i \in R_m} 1\left(y_i \neq \hat{k}_m\right) = 1 - \hat{p}_{m\hat{k}_m}$$

2. Gini index

$$\sum_{k \neq k'} \hat{p}_{mk}\hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}\left(1 - \hat{p}_{mk}\right)$$

3. Cross-entropy or deviance

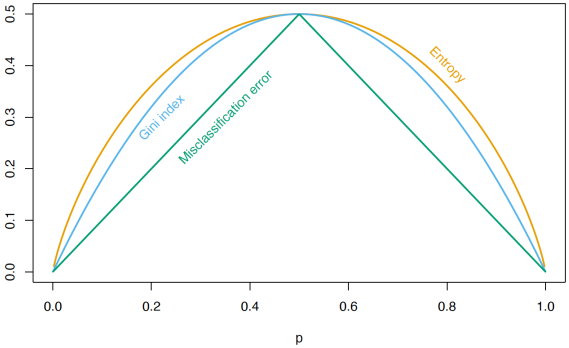
$$-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

1. Gini: a measure of diversity–if a species is present with p, weight it by $1-p$; Entropy: weighted by $\log(1/p)$, in thermodynamics, a measure of disorder; information theory, a measure of uncertainty.
2. Compare the cross-entropy with the K-class multinomial deviance ($-2\times$log-likelihood): $-\sum_{k=1}^{K} 1(y=k)\log p_k$

When there are only two classes,

- Misclassification error $= 1 - \max(p, 1-p)$
- Gini index $= 2p(1-p)$
- Cross-entropy $= -p\log(p) - (1-p)\log(1-p)$

$0\log(0) = 0$



ESL Fig 9.3.

Remark:

Gini index or the cross-entropy are both more sensitive to node purity. It is possible some splits yield two terminal nodes that have the same predicted value (predicted label).This is possible because one node may be purer than the other, but still have the same majority class.

which measure to use?

- To grow a tree: use Gini index or cross-entropy
- To prune
    - if the goal is to estimate the probability of falling into each class, then use Gini or cross-entropy to prune.
    - if the goal is to predict (high prediction accuracy), use the misclassification error rate.

Related issues about trees

Related issues about trees

Advantages of trees

- Handles both categorical and ordered variables in a simple and natural way.
- Automatic stepwise variable selection and complexity reduction
- It provides an estimate of the misclassification rate for a query sample.
- It is invariant under all monotone transformations of individual ordered variables.
- Robust to outliers and misclassified points in the training set.
- Easy to interpret.

Limitation of trees

- Instability: high variance. Errors at top nodes propagate to lower ones, therefore small change in data may result in a very different series of splits, making interpretations somewhat precautious.
- Lack of smoothness
- difficulty in capturing additive structures like $Y = c_1 1(X_1 < t_1) + c_2 1(X_2 < t_2) + \epsilon$

More on splits

Why binary splits?

- Multiway splits fragment the data too quickly, leaving insufficient data at the next level down
- Multiway splits can be achieved by a series of binary splits

Linear Combination Splits

- Use the splits of the form $\sum a_j X_j \leq s$. The weights a_j and split point s are optimized to minimize the relevant criterion.
- It can improve the predictive power of the tree, but hurts interpretability.
- The amount of computation increases significantly. Model becomes more complex.

Handling of missing values

Missing values for some variables are often encountered in high dimensional data.

Traditional approaches:

- Discard observations with missing values or example, depletion of training set
- Impute the missing values, for example, via the mean over complete observations

Two better approaches:

- For categorical variables, create the category "missing": helps to discover that observations with missing values behave differently than those complete observations.
- *Surrogate split*: Construct surrogate variables besides the best splitting variable (CART, MARS and PRIM). If the primary splitting variable is missing, use surrogate splits in order. (Section 9.2.4)