Classification and Regression Trees

Matthew S. Shotwell, Ph.D.

Department of Biostatistics
Vanderbilt University School of Medicine
Nashville, TN, USA

March 20, 2020

Introduction

- "Classification And Regression Trees"
- trees partition feature space into a set of rectangles
- fit simple model in each partition (e.g., constant)
- can handle quantitative/categorical inputs/outputs

- quantitative response Y and inputs X_1 and X_2, all with support in $[0, 1]$
- top-left partition is complex
- top-right partitions are recursive, can be described by tree at bottom-left
- trees have (root, internal, and terminal) nodes and branches
- trees with small number of terminal nodes (e.g., 2) called "stump"
- terminal nodes also called leaf node

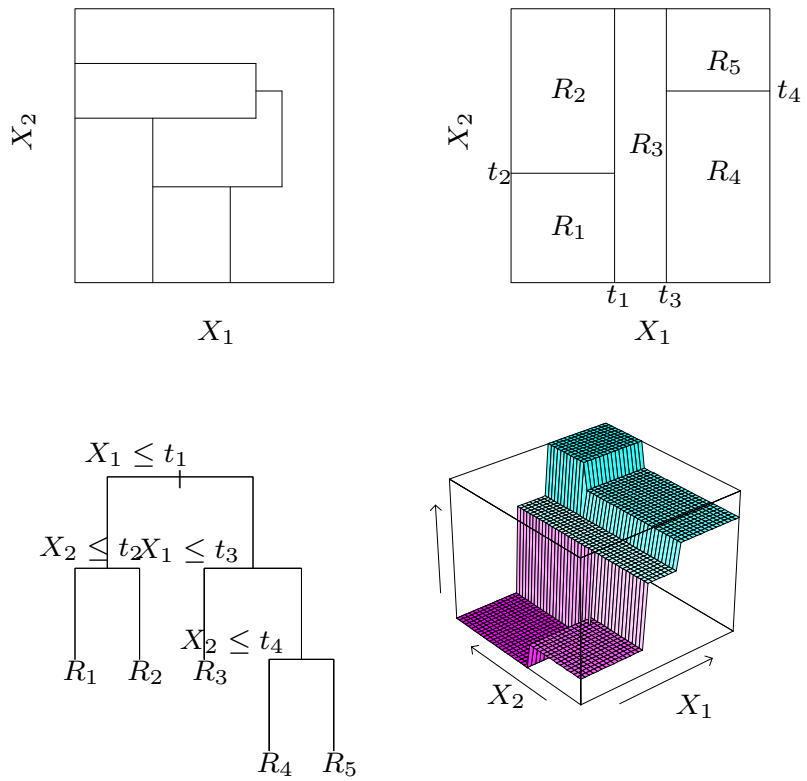
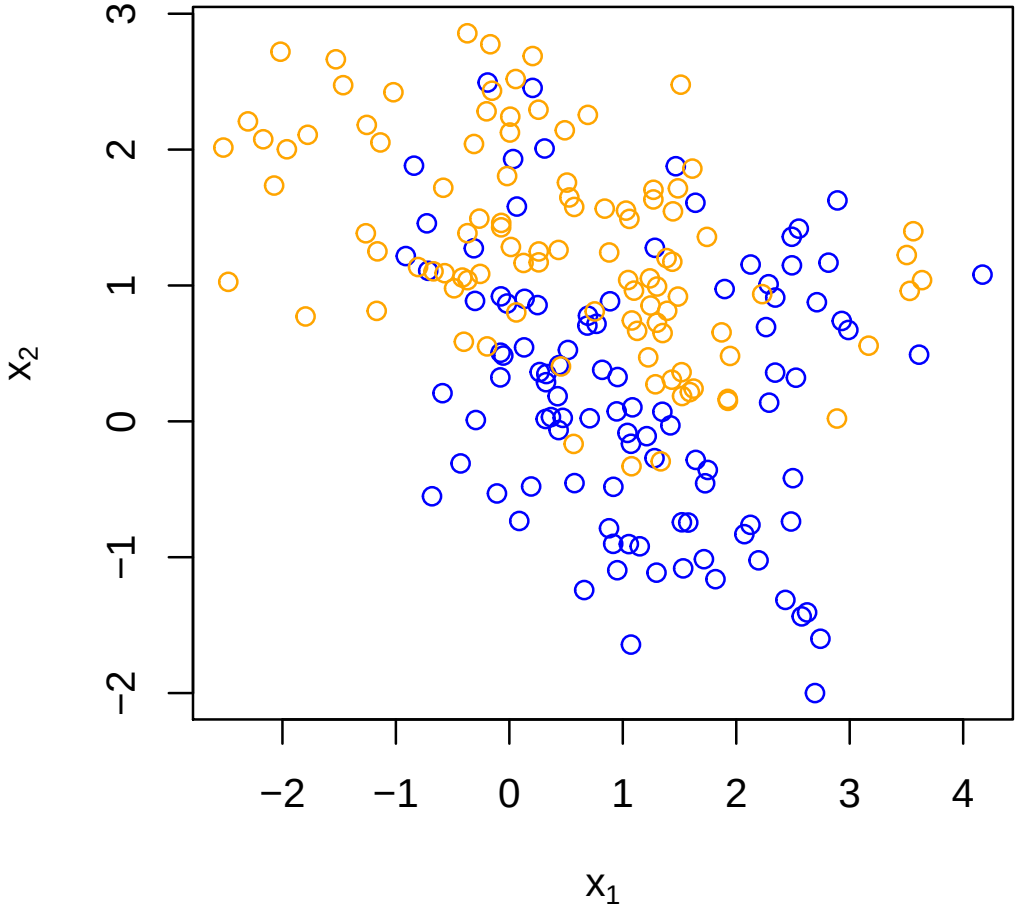


FIGURE 9.2. *Partitions and CART. Top right panel shows a partition of a two-dimensional feature space by recursive binary splitting, as used in CART, applied to some fake data. Top left panel shows a general partition that cannot be obtained from recursive binary splitting. Bottom left panel shows the tree corresponding to the partition in the top right panel, and a perspective plot of the prediction surface appears in the bottom right panel.*

Fitting recursive binary trees

- consider all ways to make a single split of a feature into two regions (must consider each feature separately)
- simple prediction (e.g., mean of Y) in each region
- choose feature and split-point to achieve the best fit
- one or both resulting regions are split again
- repeat until some stopping rule is applied


3
2
1
0
−1
−2
x_2
−2
−1
0
1
2
3
4
x_1

Fitting (growing) regression trees

Regression trees have this form:

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

Under squared-error loss, conditional on R_m, $\hat{c}_m$ is the sample mean of Y in region R_m. Finding partitions R_m is more difficult. Recursive partitioning is "greedy algorithm" to find R_m: consider a splitting variable j and split point s and define resulting regions as follows:

$$R_1(j,s) = \{X : X_j \leq s\} \quad \text{and} \quad R_2(j,s) = \{X : X_j > s\}$$

At each iteration, task is to find split; find j and s that minimize:

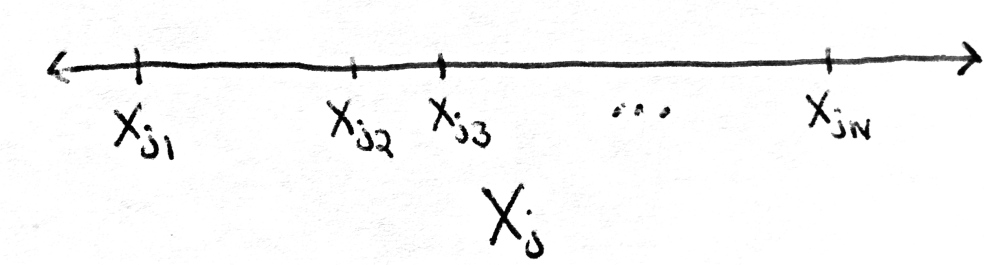
$$RSS(j,s) = \sum_{x_i \in R_1(j,s)} (y_i - \hat{c}_1)^2 + \sum_{x_i \in R_2(j,s)} (y_i - \hat{c}_2)^2$$

Fitting (growing) regression trees

The splitting task is then to find j and s that minimize:

$$RSS(j,s) = \sum_{x_i \in R_1(j,s)} (y_i - \hat{c}_1)^2 + \sum_{x_i \in R_2(j,s)} (y_i - \hat{c}_2)^2$$

- this may seem difficult at first, but
- for variabe j, only $N-1$ distinct splits s in the training data



- for given j, $RSS(j,s)$ is constant between X_{j1} and X_{j2}
- for tractable $N \times p$, can simply enumerate all $RSS(j,s)$
- split at optimal j and s, then repeat within each split
- proceed this way until stopping rule triggered

Stopping rules

- measure of tree complexity are tuning parameters
  - maximum tree depth (number of splits)
  - minimum number of training obs per region (i.e., "node size")
- another approach: grow a large tree, stopping only when minimum node size is reached, then "prune" tree back using a "cost-complexity" criterion

Pruning

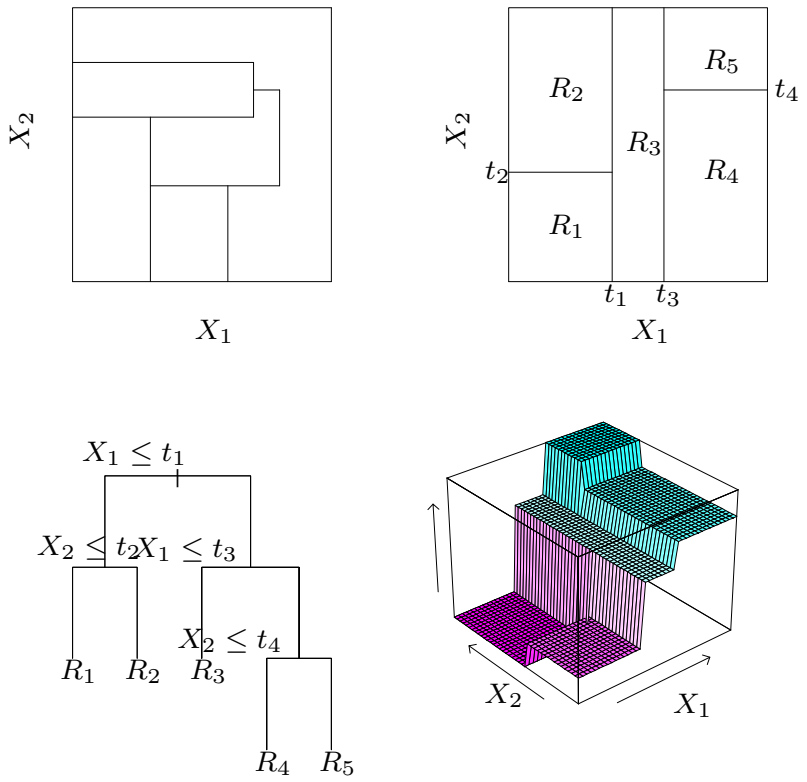


FIGURE 9.2. *Partitions and CART. Top right panel shows a partition of a two-dimensional feature space by recursive binary splitting, as used in CART, applied to some fake data. Top left panel shows a general partition that cannot be obtained from recursive binary splitting. Bottom left panel shows the tree corresponding to the partition in the top right panel, and a perspective plot of the prediction surface appears in the bottom right panel.*

Cost-complexity pruning

- let $T \subset T_0$ be a sub-tree obtained by pruning T_0
- let $|T|$ be the number of terminal nodes
- let N_m be the node size $\sum_{i=1}^{N} I(x_i \in R_m)$
- let $\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$
- let $Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$ be lack of fit (LOF)
- the "cost-complexity" criterion is

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

Cost-complexity pruning

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

- pruning increases cost, lowers complexity
- for given α, find T_α that minimizes $C_\alpha(T)$
- greedy algorithm (weakest-link pruning):
  1. collapse internal node that produces smallest increase in $C_\alpha(T)$
  2. continue until just one node
  3. select among sequence of trees; T_α must be part of this sequence
- α is a tuning parameter; selected using, e.g., cross validation

Classification trees

- for $k = 1, \ldots, K$ classes
- let $\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$
- $\hat{p}_{mk}$ is fraction of class k in node m
- apply loss function-specific classification rule to $\hat{p}_{mk}$
- to fit and prune tree, also need metric of node impurity and LOF (for cost-complexity pruning):
  - misclassification rate (depends on classification loss/rule)
  - Gini index - $\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
  - cross-entropy deviance - $\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

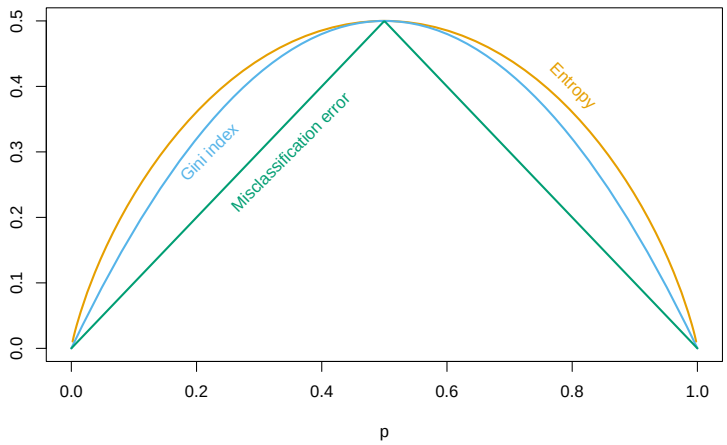


FIGURE 9.3. *Node impurity measures for two-class classification, as a function of the proportion p in class 2. Cross-entropy has been scaled to pass through* $(0.5, 0.5)$.

Why prefer misclassification vs. Gini vs. cross-entropy?

- say we have a two-class problem with 400 in each class, denoted (400,400)
- consider two candidate splits, where the classes are distributed among the two splits as follows:
  - s_1: (300,100) and (100,300)
  - s_2: (200,400) and (200,0)
- both splits have misclassification rate 0.25 (assuming zero-one classificationloss), but second split produces a "pure" node, which is preferable in many cases
- both Gini and cross-entropy give preference to split with pure node; thus often used in growing trees
- often misclassification is used for pruning

Problems with trees

- instability; sample variability in tree structure
- lack of smoothness of prediction surface in feature space
- categorical features; $2^{q-1} - 1$ partitions of categorical predictor with q values into 2 parts; can be simplified for binary outcomes using Gini or entropy loss, and quantitative outcomes using squared-error loss
- many tuning parameters (max depth, min node size, cost-complexity penalty)

R package `rpart`

- R package `rpart` implements recursive partitioning for classification and regressoin trees
- great vignette written my Terry Therneau and others

Code example

`mixture-data-rpart.R`