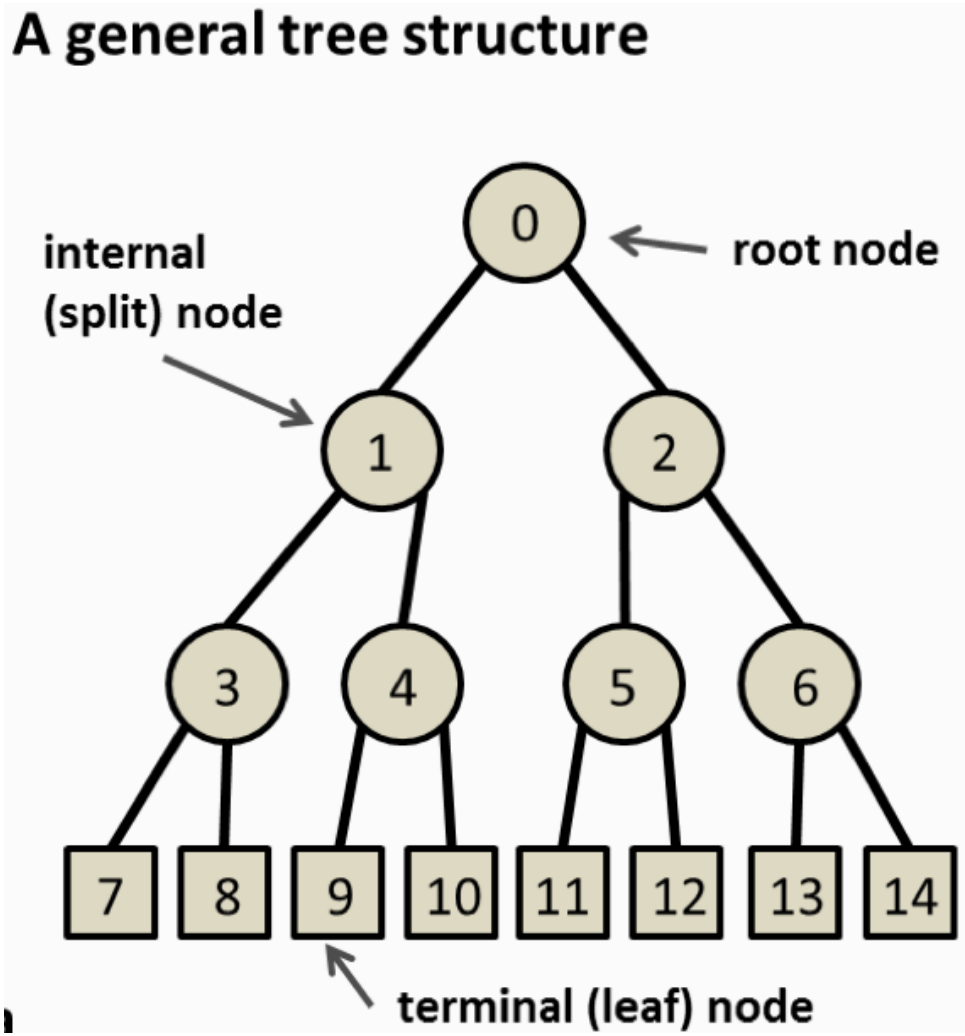
Classification and Regression Trees

David Rosenberg

New York University

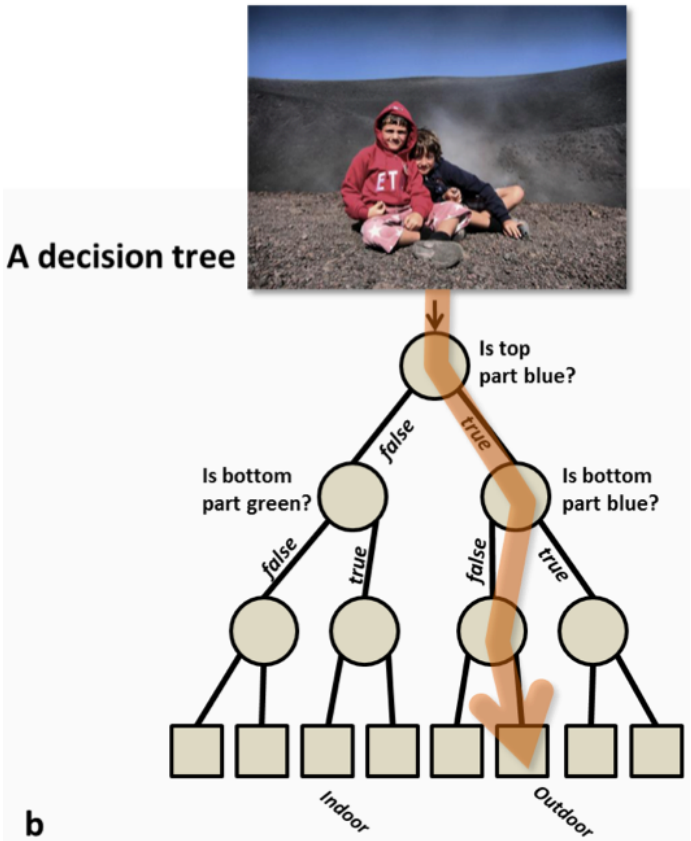
October 29, 2016

General Tree Structure



From Criminisi et al. MSR-TR-2011-114, 28 October 2011.

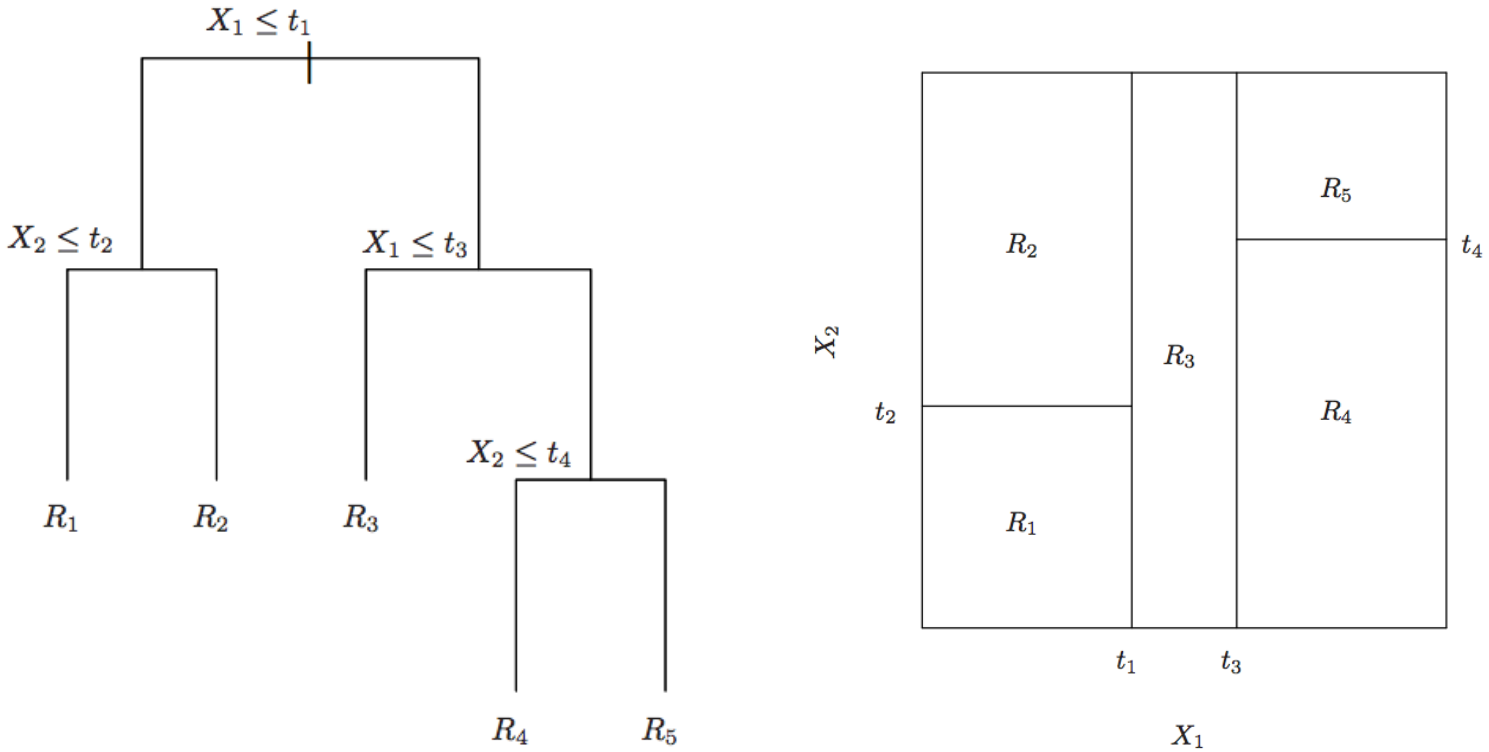
Decision Tree



From Criminisi et al. MSR-TR-2011-114, 28 October 2011.

Binary Decision Tree on $\mathbf{R}^2$

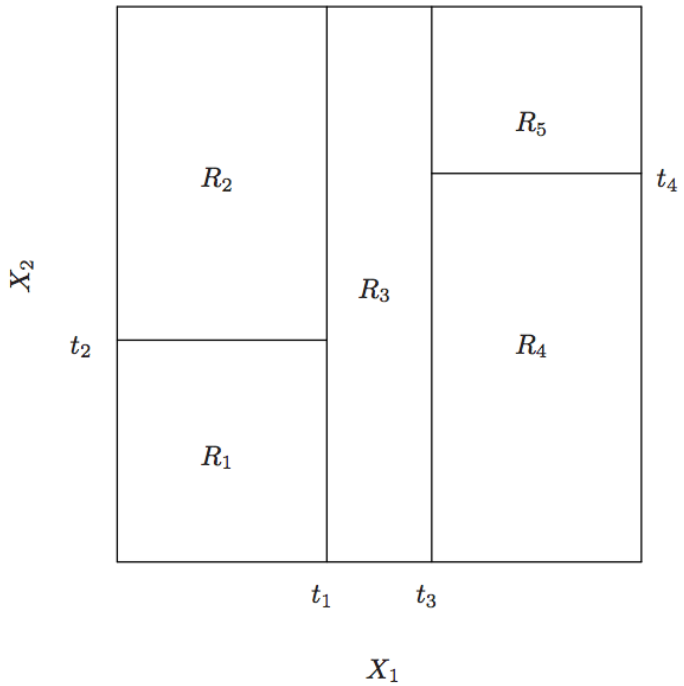
- Consider a binary tree on $\{(X_1, X_2) \mid X_1, X_2 \in \mathbf{R}\}$

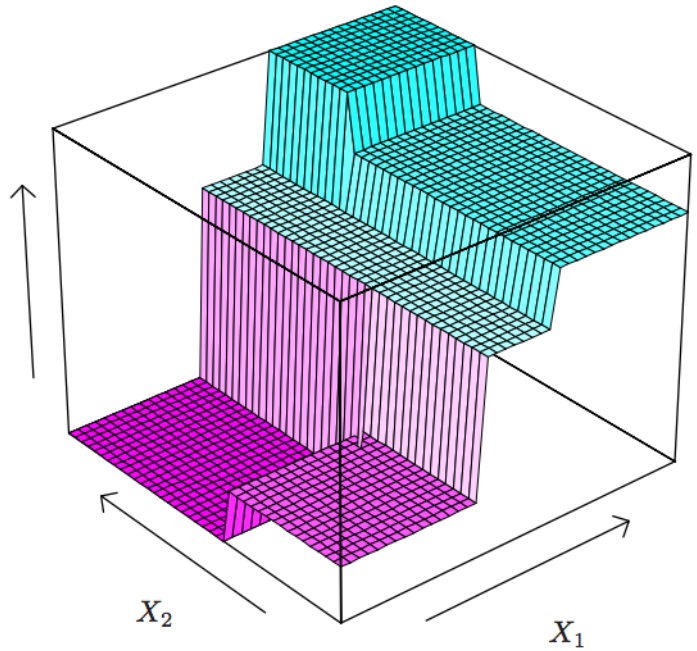


From *An Introduction to Statistical Learning, with applications in R* (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

Binary Regression Tree on $\mathbf{R}^2$

- Consider a binary tree on $\{(X_1, X_2) \mid X_1, X_2 \in \mathbf{R}\}$





From *An Introduction to Statistical Learning, with applications in R* (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

Fitting a Regression Tree

- The decision tree gives the partition of $\mathcal{X}$ into regions:

$$\{R_1, \ldots, R_M\}.$$

- Recall that a partition is a **disjoint union**, that is:

$$\mathcal{X} = R_1 \cup R_2 \cup \cdots \cup R_M$$

and

$$R_i \cap R_j = \emptyset \quad \forall i \neq j$$

Fitting a Regression Tree

- Given the partition $\{R_1, \ldots, R_M\}$, final prediction is

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

- How to choose $c_1, \ldots, c_M$?
- For loss function $\ell(\hat{y}, y) = (\hat{y} - y)^2$, best is

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

Complexity of a Tree

- Let $|T| = M$ denote the number of terminal nodes in T.
- We will use $|T|$ to measure the complexity of a tree.
- For any given complexity,
  - we want the tree minimizing square error on training set.
- Finding the optimal binary tree of a given complexity is computationally intractable.
- We proceed with a **greedy algorithm**
  - Means build the tree one node at a time, without any planning ahead.

Root Node, Continuous Variables

- Let $x = (x_1, \ldots, x_d) \in \mathbf{R}^d$.
- **Splitting variable** $j \in \{1, \ldots, d\}$.
- **Split point** $s \in \mathbf{R}$.
- Partition based on j and s:

$$R_1(j, s) = \{x \mid x_j \leqslant s\}$$
$$R_2(j, s) = \{x \mid x_j > s\}$$

Root Node, Continuous Variables

- For each splitting variable j and split point s,

$$\begin{aligned} \hat{c}_1 &= \text{ave}(y_i \mid x_i \in R_1) \\ \hat{c}_2 &= \text{ave}(y_i \mid x_i \in R_2) \end{aligned}$$

- Find j, s minimizing

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

Then Proceed Recursively

1. We have determined R_1 and R_2
2. Find best split for points in R_1
3. Find best split for points in R_2
4. Continue...

- When do we stop?

Complexity Control Strategy

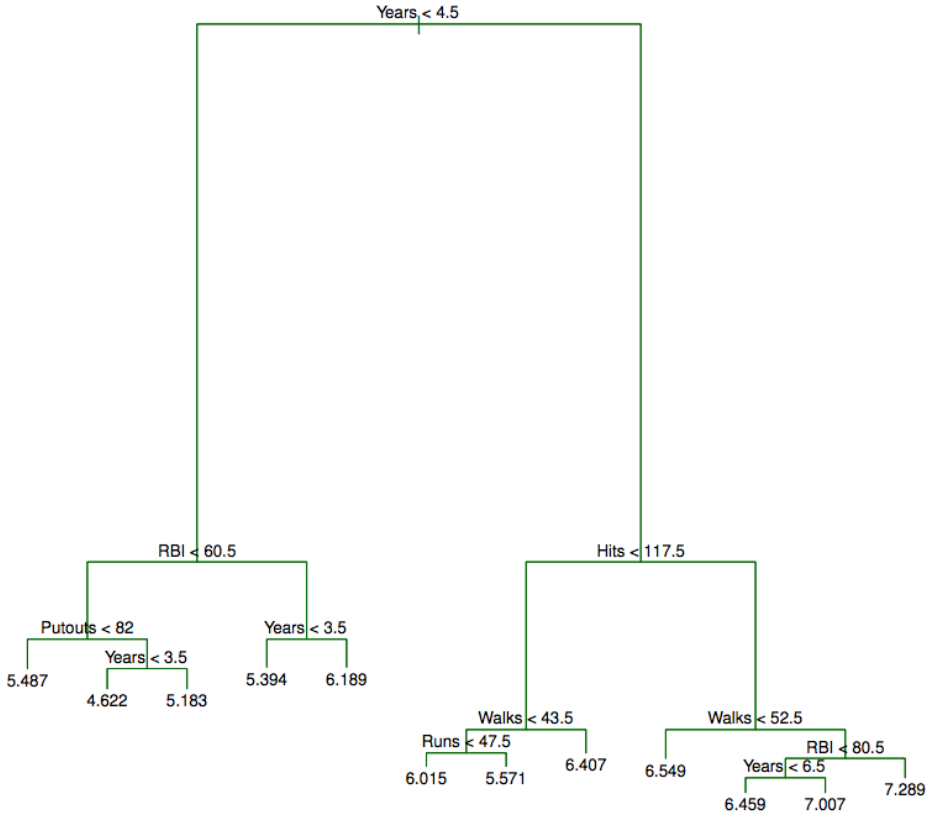
- If the tree is too big, we may overfit.
- If too small, we may miss patterns in the data (underfit).
- Typical approach:
  1. Build a really big tree (e.g. until all regions have $\leqslant 5$ points).
  2. Prune the tree.

Tree Terminology

- Each **internal node**
  - has a splitting variable and a split point
  - corresponds to binary partition of the space
- A **terminal node** or **leaf node**
  - corresponds to a region
  - corresponds to a particular prediction
- A **subtree** $T \subset T_0$ is any tree obtained by **pruning** T_0, which means collapsing any number of its internal nodes.

Tree Pruning

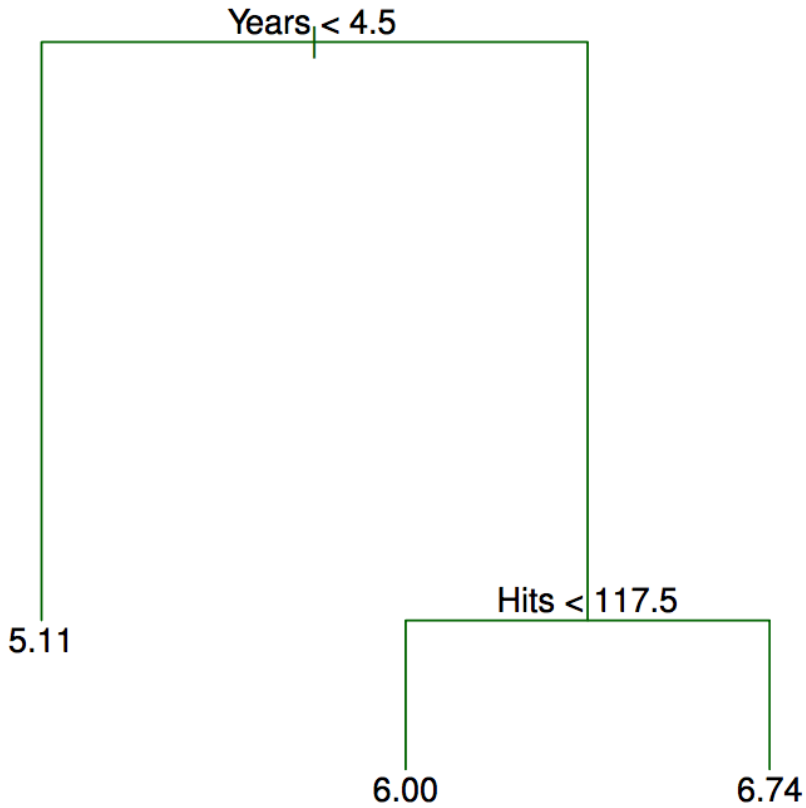
- Full Tree T_0



From *An Introduction to Statistical Learning, with applications in R* (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

Tree Pruning

- Subtree $T \subset T_0$



From *An Introduction to Statistical Learning, with applications in R* (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

Emprical Risk and Tree Complexity

- Suppose we want to prune a big tree T_0.
- Let $\hat{R}(T)$ be the empirical risk of T (i.e. square error on training)
- Clearly, for any $T \subset T_0$, $\hat{R}(T) \geqslant \hat{R}(T_0)$.
- Let $|T|$ be the number of terminal nodes in T.
- $|T|$ is our measure of complexity for a tree.

Cost Complexity (or Weakest Link) Pruning

Definitions

The **cost complexity criterion** with parameter α is

$$C_\alpha(T) = \hat{R}(T) + \alpha|T|$$

- Trades off between empirical risk and complexity of tree.
- Cost complexity pruning:
  - For each α, find the tree $T \subset T_0$ minimizing $C_\alpha(T)$.
  - Use cross validation to find the right choice of α.

Greedy Pruning is Sufficient

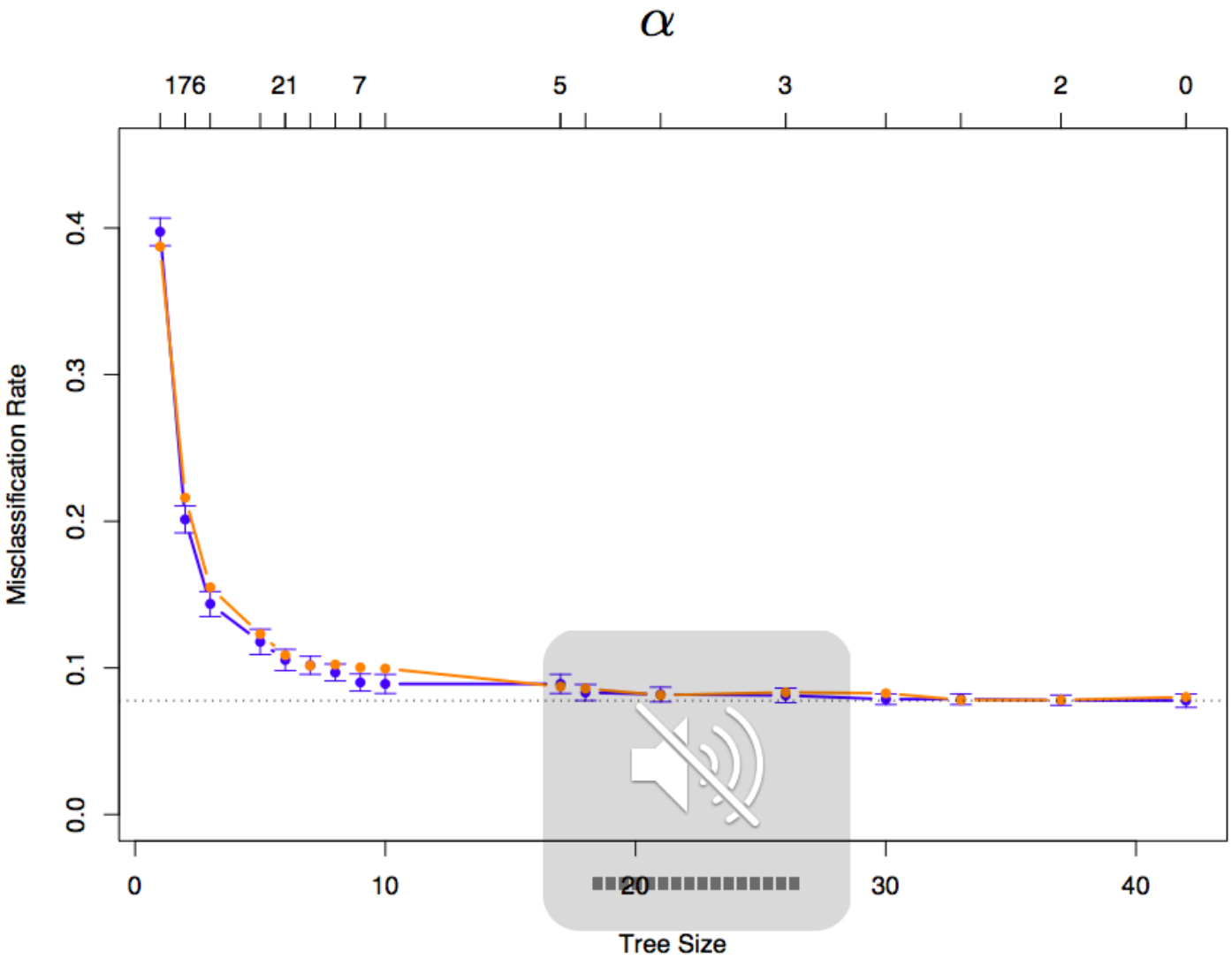
- Find subtree $T_1 \subset T_0$ that minimizes $\hat{R}(T_1) - \hat{R}(T_0)$.
- Then find $T_2 \subset T_1$.
- Repeat until we have just a single node.
- If N is the number of nodes of T_0 (terminal and internal nodes), then we end up with a set of trees:

$$\mathcal{T} = \left\{ T_0 \supset T_1 \supset T_2 \supset \cdots \supset T_{|N|} \right\}$$

- Breiman et al. (1984) proved that this is all you need. That is:

$$\left\{ \operatorname*{arg\,min}_{T \subset T_0} C_\alpha(T) \mid \alpha \geqslant 0 \right\} \subset \mathcal{T}$$

Regularization Path for Trees



Classification Trees

- Consider classification case: $\mathcal{Y} = \{1, 2, \ldots, K\}$.
- We need to modify
  - criteria for splitting nodes
  - method for pruning tree

Classification Trees

- Let node m represent region R_m, with N_m observations
- Denote proportion of observations in R_m with class k by

$$\hat{p}_{mk} = \frac{1}{m} \sum_{\{i : x_i \in R_m\}} 1(y_i = k).$$

- **Predicted classification** for node m is

$$k(m) = \arg\max_k \hat{p}_{mk}.$$

- **Predicted class probability distribution** is $(\hat{p}_{m1}, \ldots, \hat{p}_{mK})$.

Misclassification Error

- Consider node m representing region R_m, with N_m observations
- Suppose we predict

$$k(m) = \arg\max_k \hat{p}_{mk}$$

  as the class for all inputs in region R_m.
- What is the misclassification rate on the training data?
- It's just

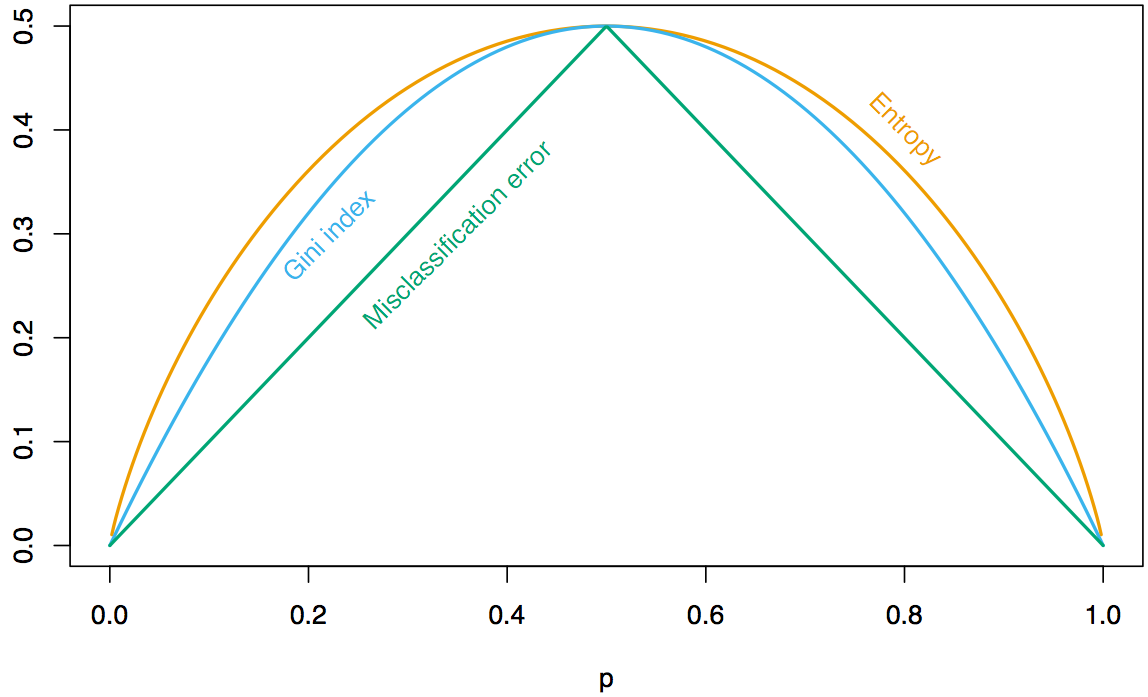
$$1 - \hat{p}_{mk(m)}.$$

Classification Trees: Node Impurity Measures

- Consider node m representing region R_m, with N_m observations
- How can we generalize from squared error to classification?
- We will introduce some different measures of **node impurity**.
  - We want **pure** leaf nodes (i.e. as close to a single class as possible)
- We'll find splitting variables and split point **minimizing node impurity**.

Two-Class Node Impurity Measures

- Consider binary classification
- Let p be the relative frequency of class 1.
- Here are three node impurity measures as a function of p

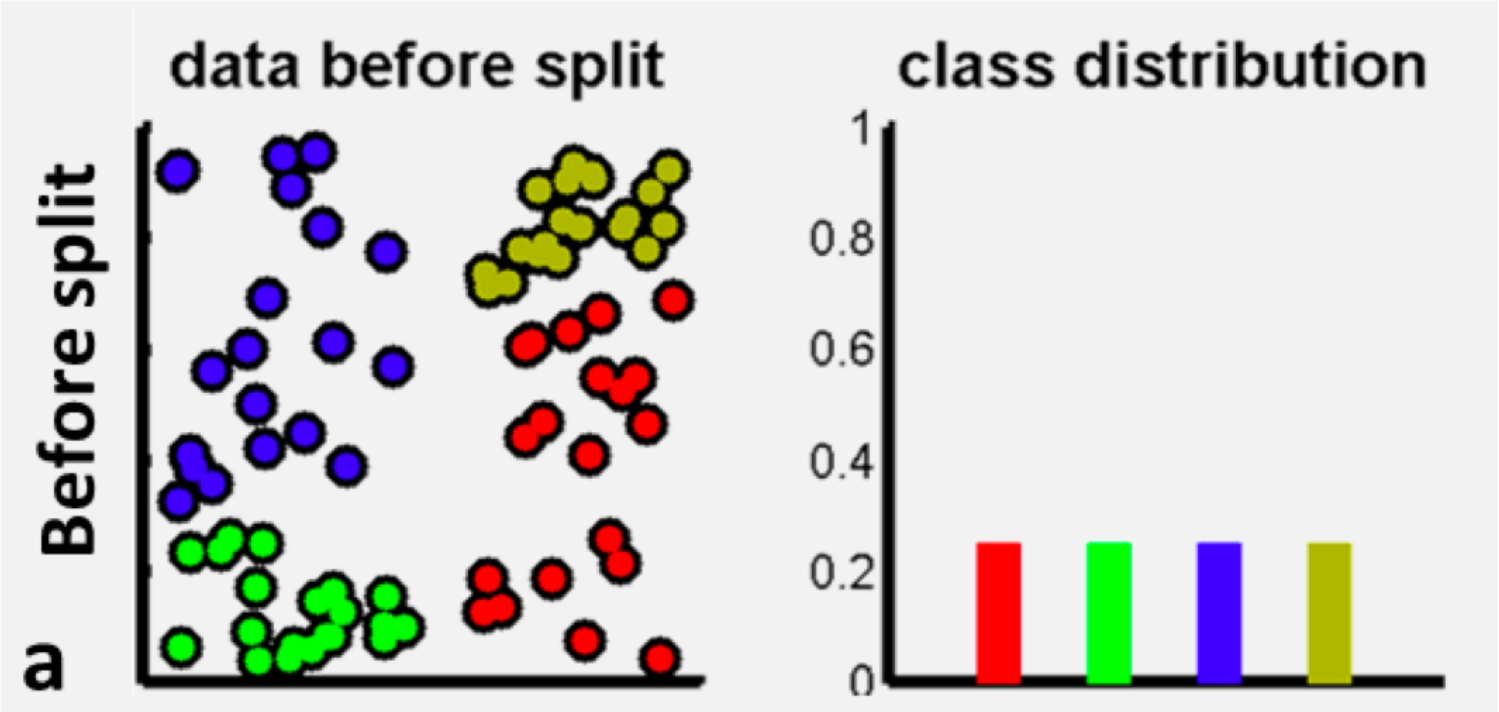


HTF Figure 9.3

Classification Trees: Node Impurity Measures

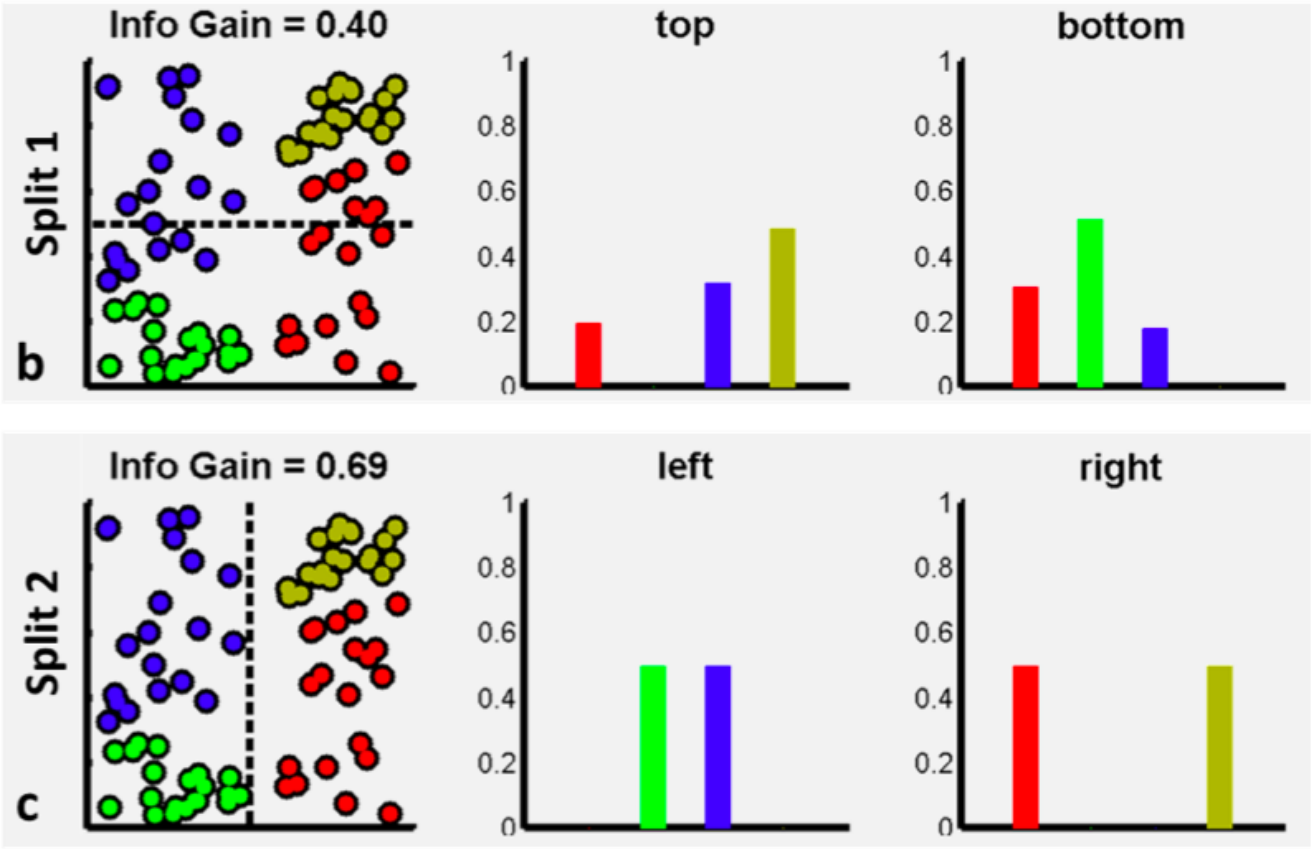
- Consider leaf node m representing region R_m, with N_m observations
- Three measures $Q_m(T)$ of **node impurity** for leaf node m:
  - Misclassification error:
$$1 - \hat{p}_{mk(m)}.$$
  - Gini index:
$$\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$
  - Entropy or deviance:
$$-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

Class Distributions: Pre-split



From Criminisi et al. MSR-TR-2011-114, 28 October 2011.

Class Distributions: Split Search



- (Maximizing information gain is equivalent to minimizing entropy)

From Criminisi et al. MSR-TR-2011-114, 28 October 2011.

Classification Trees: How exactly do we do this?

- Let R_L and R_R be regions corresponding to a potential node split.
- Suppose we have N_L points in R_L and N_R points in R_R.
- Let $Q(R_L)$ and $Q(R_R)$ be the node impurity measures.
- The we search for a split that minimizes

$$N_L Q(R_L) + N_R Q(R_R)$$

Classification Trees: Node Impurity Measures

- For building the tree, Gini and Entropy are more effective.
  - They push for more pure nodes, not just misclassification rate
- For pruning the tree, use misclassification error – closer to risk estimate.

Missing Features (or "Predictors")

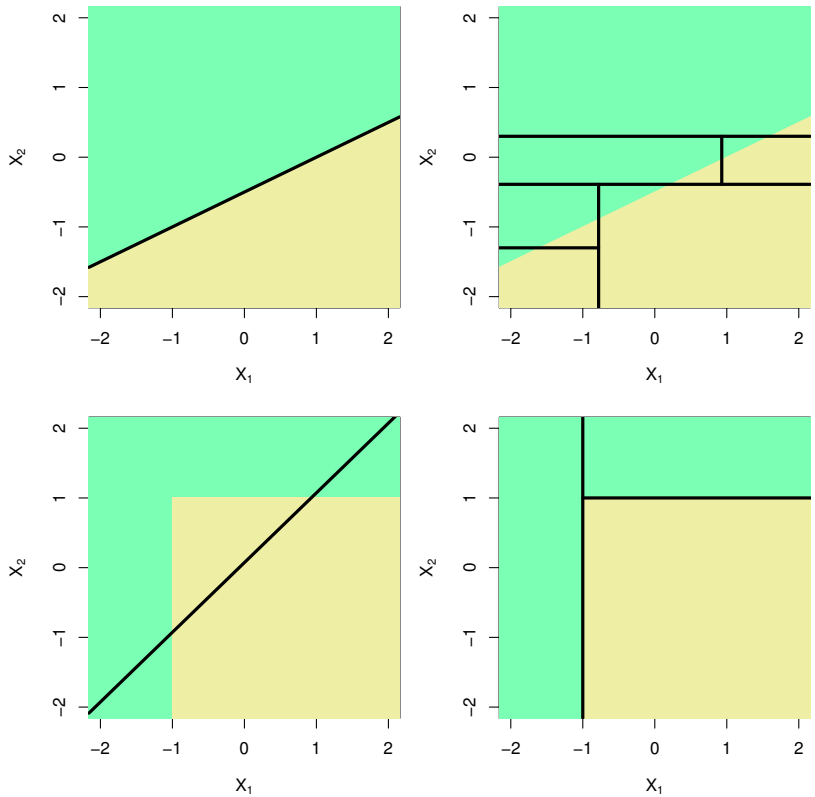
- Features are also called **covariates** or **predictors**.
- What to do about missing features?
  - Throw out inputs with missing features
  - Impute missing values with feature means
  - If a categorical feature, let "missing" be a new category.
- For trees, can use **surrogate splits**
  - For every internal node, form a list of surrogate features and split points
  - Goal is to approximate the original split as well as possible
  - Surrogates ordered by how well they approximate the original split.

Categorical Features

- Suppose we have feature with q possible values (unordered).
- We want to find the best split into 2 groups
- There are $2^{q-1} - 1$ possible partitions.
- Search time?
  - For binary classification ($K = 2$), there is an efficient algorithm. (Breiman 1984)
  - Otherwise, can use approximations.
- Statistical issue?
  - If a category has a very large number of categories, we can overfit.
  - Extreme example: Row Number could lead to perfect classification with a single split.

Trees vs Linear Models

- Trees have to work much harder to capture linear relations.



From *An Introduction to Statistical Learning, with applications in R* (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

Interpretability

- Trees are certainly easy to explain.
- You can show a tree on a slide.
- Small trees seem interpretable.
- For large trees, maybe not so easy.

Trees for Nonlinear Feature Discovery

- Suppose tree T gives partition $R_1, \ldots, R_m$.
- Predictions are

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

- If we make a feature for every region R:

$$1(x \in R),$$

  we can view this as a **linear model**.
- Trees can be used to discover nonlinear features.

Instability / High Variance of Trees

- Trees are **high variance**:
  - If we randomly split the data, we may get quite different trees from each part
- By contrast, linear models have low variance (at least when well-regularized)
- Later we investigate several ways to reduce this variance

Comments about Trees

- Trees make no use of **geometry**
  - No inner products or distances
  - called a "nonmetric" method
  - **Feature scale irrelevant**
- Predictions are not continuous
  - not so bad for classification
  - may not be desirable for regression