Lecture 20: Classification and Regression Trees

Hao Helen Zhang

Outline

- Basic Ideas
- Tree Construction Algorithm
- Specific Issues
  - Parameter Tuning
  - Choice of Impurity Measure
  - Missing Values
- Properties

Characteristics of Classification Trees

- Main Characteristics:
  - very flexible, very intuitive
  - non-model based
  - hierarchical nature
  - natural graphical display, easy to interpret
- Classification trees are widely used in applied fields including
  - medicine (diagnosis), computer science (data structures), botany (classification), psychology (decision theory).
- Popular tree method:
  - CART (Classification and Regression Trees) by Breiman, Friedman, Olshen, Stone (1984)

Classification Tree Decision Example

When heart attack patients are admitted to a hospital, dozens of tests are often performed to obtain various measures such as heart rate, blood pressure, age, medical history, and so on.

- Short-term goal: to predict whether they can survive the heart attack, say, at least 30 days.
- Long-term goals: to develop treatments for patients, identify high-risk patients, advance medical theory on heart failure.

Three-Question Decision Tree

Breiman et al. (1984) addressed this problem using a simple, three-question decision tree.

- "If the patient's minimum systolic blood pressure over the initial 24 hour period is greater than 91, then if the patient's age is over 62.5 years, then if the patient displays sinus tachycardia, then and only then the patient is predicted not to survive for at least 30 days."

Hierarchical Nature of Classification Trees

The hierarchical nature of classification trees is one of their most basic feature:

- A hierarchy of questions are asked and the final decision depends on the answers to all the previous questions.
- Similarly, the relationship of a leaf to the tree on which it grows can be described by the hierarchy of splits of branches (starting from the trunk) leading to the last branch from which the leaf hangs
- most decision trees are drawn downward on paper (a upside-down tree)

Basic Ideas of Classification Trees

Assume the outcome Y takes values $1, 2, \cdots, K$.

- A classification tree repeatedly partitions the feature space into a set of rectangles
  - first split the space into two regions, and model the response by the majority vote of Y in each region.
  - One or both of the regions are split into two more regions.
  - This process is continued, until some stopping rule is applied.
- At each step, *we choose the variable and split-point to achieve the best fit*. In the following example:
  - four splits: $X_1 = t_1, X_2 = t_2, X_1 = t_3, X_2 = t_4$
  - five regions: $R_1, \cdots, R_5$

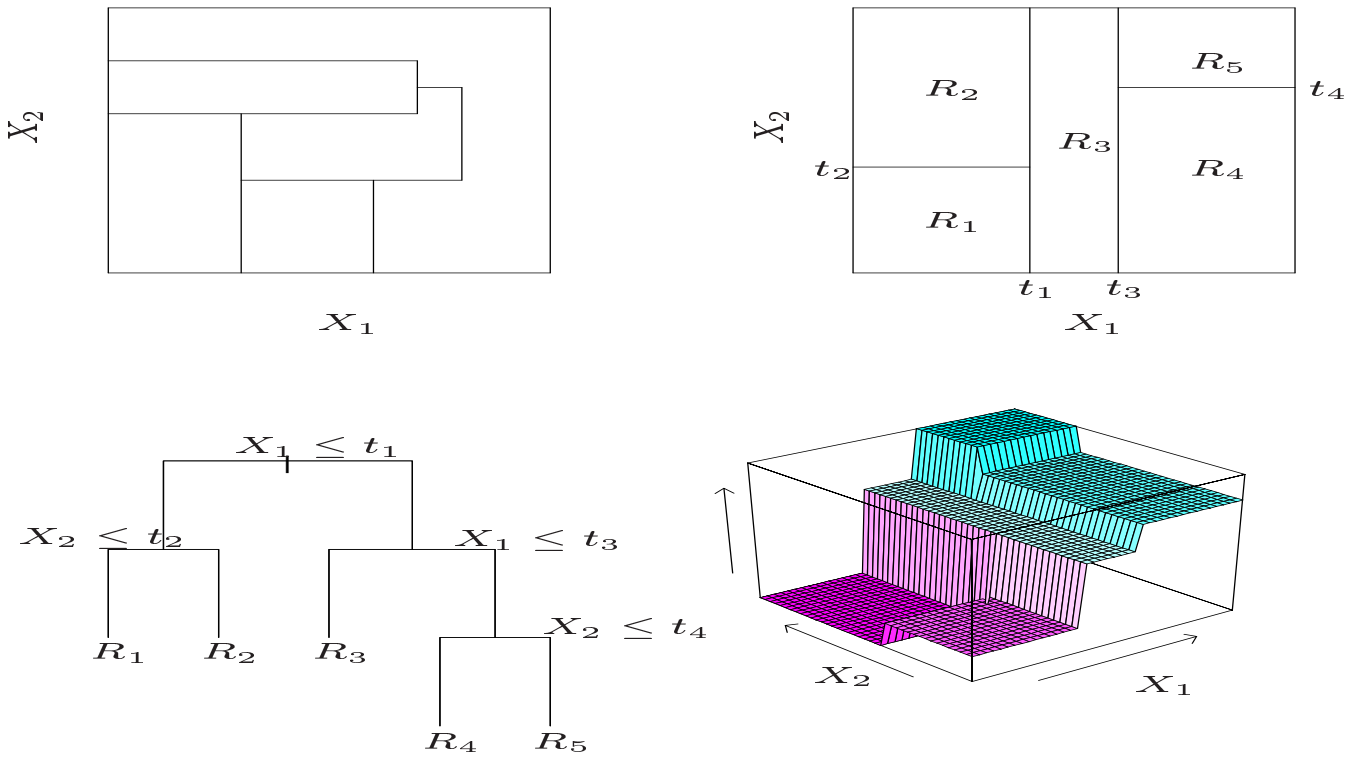


Figure 9.2: *Partitions and CART. Top right panel shows a partition of a two-dimensional feature space by recursive binary splitting, as used in CART, applied to some fake data. Top left panel shows a general partition that cannot be obtained from recursive binary splitting. Bottom left panel shows the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

Elements of A Tree: Nodes and Splits

T: a collection of nodes (t) and splits (s).

- root node
- terminal node (leaf node)
  - Each leaf node is assigned with a class fit.
- parent node
- child node
  - left node (t_L)
  - right node (t_R)
- The size of tree:

$$|T| = \text{the number of terminal nodes in } T$$

Three Processes to Grow A Tree

Three elements:

1. Split process
   - choose splitting variables and split points
   - goodness of split criterion $\Phi(s, t)$ 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 together

Step 1: Splitting Process

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

- If X_j is ordered (continuous), the splitting question is

$$\{\text{Is} \quad X_j \leq s?\}$$

for all real values s.

  - Since the training data set is finite, there are only finitely many distinct splits generated by the question.

- If X_j is categorical taking values $\{1, ..., M\}$, the splitting question is

$$\{\text{Is} \quad X_j \in A?\}$$

A is a subset of $\{1, ..., M\}$.

Choose Best Split

For the splitting variable j and split points s, define the pair of half-planes

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

- Each split produces two subnodes.
- We scan through all the inputs and all the possible splits quickly and determine the best pair (j,s) yielding two most "pure" nodes, i.e.,

$$\min_{j,s}[\phi_{R_1} + \phi_{R_2}],$$

here ϕ_{R_m} is the some purity measure of node R_m for $m = 1, 2$.

Purity Measure of Multiclass Problems

In multiclass classification problems,

- Assume each subnode R_m contains N_m observations, let

$$\hat{p}_{mk} = \Pr(k|m) = \frac{1}{N_m} \sum_{\mathbf{x}_i \in R^m} I(y_i = k),$$

  which is the proportion of class k observations in node m.

- We classify the observations in node m to class

$$k(m) = \arg \max_{k=1,\cdots,K} \hat{p}_{mk},$$

  by the majority class in node m.

What is the characteristic of a purity (or impurity) function?

- A node is more pure if one class dominates the node than if multiple classes equally present in the node.

Impurity Functions

An impurity function is a function ϕ defined on the set

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

satisfying

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

Popular Impurity Functions

Popular Examples:

- Misclassification error:$\phi = 1 - \max_{k=1,\ldots,K} p_k$
- Gini index: $\phi = \sum_{k=1}^{K} p_k(1 - p_k)$.
- Cross-entropy (deviance): $\phi = \sum_{k=1}^{K} p_k \log(p_k)$.

Two-class Example

Let p be the proportion of class 2. Three measures have the following expressions:

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

Default: $0\log(0) = 0$.

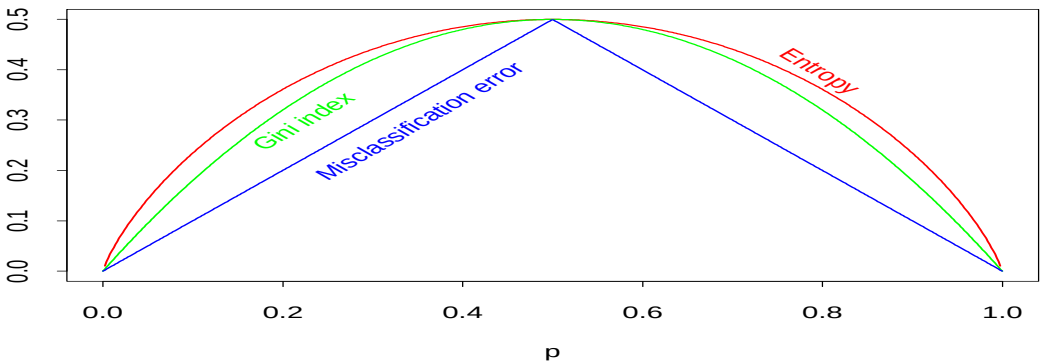


Figure 9.3: *Node impurity measures for two-class classification, as a function of the proportion p in class 2. Cross-entropy has been scaled to pass through* $(0.5, 0.5)$.

Three Commonly-used Impurity Functions

For each node R_m, its impurity ϕ_{R_m} can be estimated as following:

- Misclassification error

$$\phi_{R_m} = \frac{1}{N_m} \sum_{\mathbf{x}_i \in R^m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}$$

- Gini index:

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

- Cross-entropy (deviance):

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

Total Impurity of A Tree

Assume we use the impurity measure ϕ. Given a tree T containing m nodes R_m's.

- the size of tree is $|T|$
- the sample size of each node R_m is N_m.
- the impurity of for each node is ϕ_{R_m}.

The total impurity of the tree is

$$\sum_{m=1}^{|T|} N_m \phi_{R_m}.$$

Step 2: Partition Process

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 impurity due to splits exceeds some threshold
  - Drawback: A seemingly worthless split might lead to a very good split below it (short-sighted)
- **Preferred approach**: grow a large tree T_0, stopping the splitting process only when some minimum node size (say 5) is reached.

Step 3: Pruning Process

We use the *weakest link pruning* procedure:

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

This gives a finite sequence of subtrees. For each subtree T, we measure its cost complexity by

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

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.

- Large α results in smaller trees;
- Small α results in large trees.

Parameter Tuning for Tree

Breiman (1984) and Ripley (1996) have shown that

- For each α, there is a unique smallest subtree T_α that minimizes $C_\alpha(T)$.
- The sequence of subtrees obtained by pruning under the weakest link, must contain T_α.

In practice,

- One uses the five- or ten-fold cross validation to choose the best α.

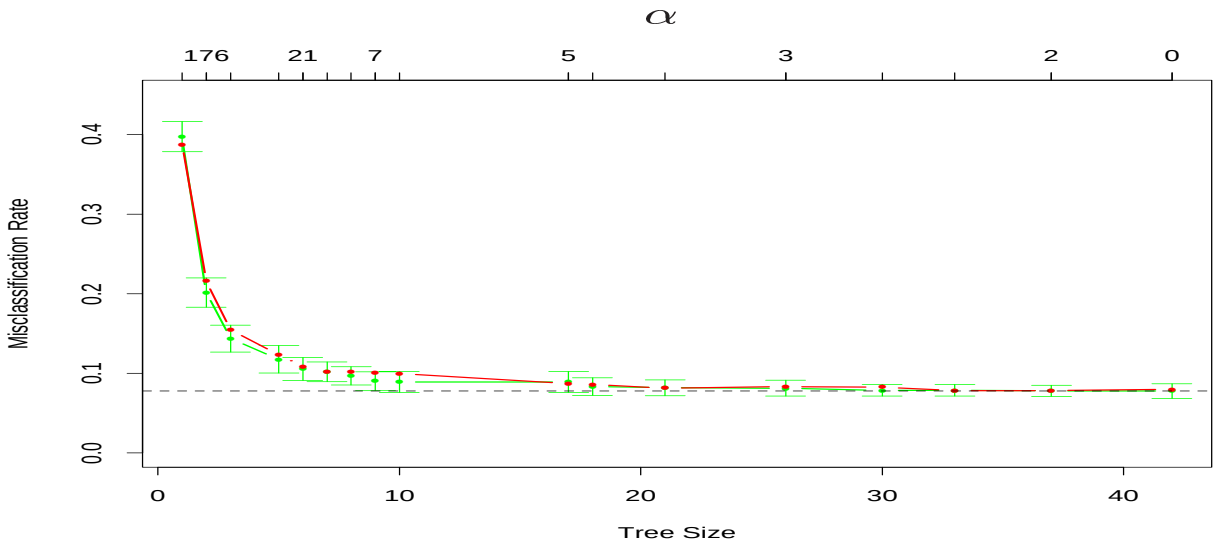


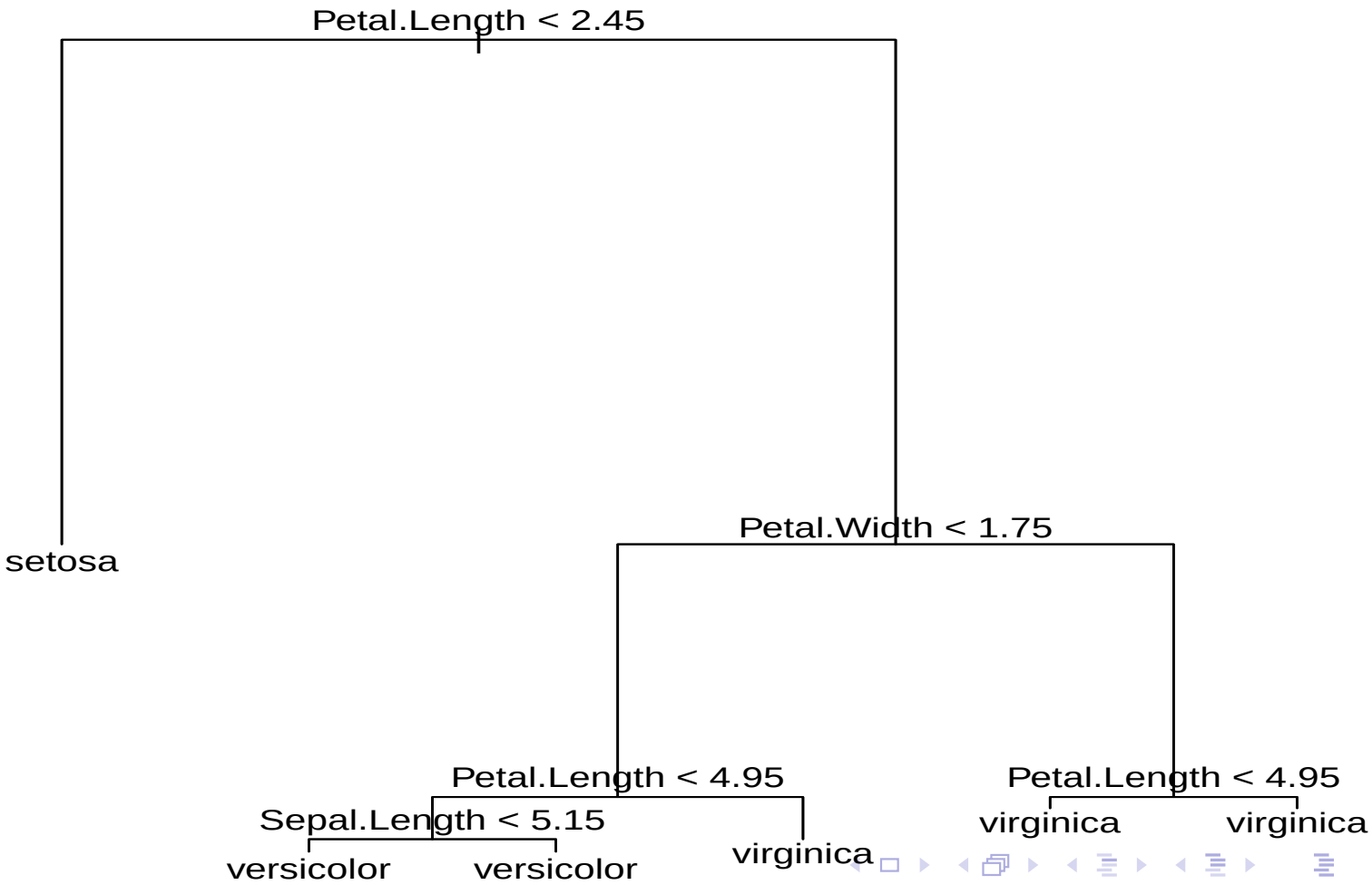
Figure 9.4: *Results for* **spam** *example. The green curve is the tenfold cross-validation estimate of misclassification rate as a function of tree size, with ± two standard error bars. The minimum occurs at a tree size with about 17 terminal nodes. The red curve is the test error, which tracks the CV error quite closely. The cross-validation was indexed by values of α, shown above. The tree sizes shown below refer to $|T_\alpha|$, the size of the original tree indexed by α.*

Iris Example

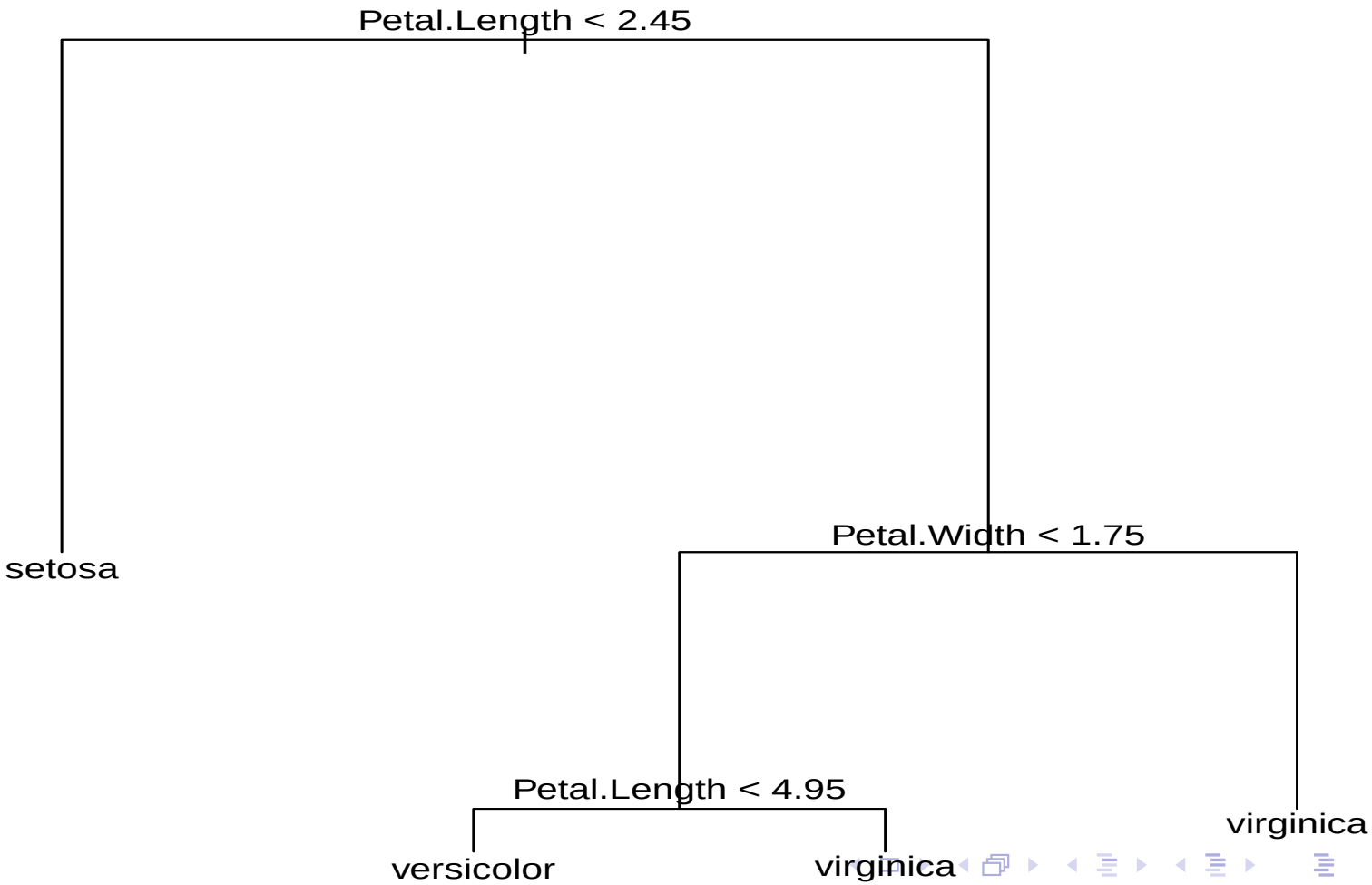
Data Information:

- $K = 3$: (three species)
  - setosa, versicolor, virginica.
- Four inputs:
  - Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
- Tree analysis
  - first, generate the tree using "Deviance" as the impurity measure
  - second, tune the cost-complexity parameter α by using 10-fold CV

Original Tree Using Deviance



After Parameter Tuning based on 10-CV



Comparison of Impurity Measures

- Gini index and cross-entropy are differentiable
- Gini index and cross-entropy are more sensitive to changes in the node probabilities
  - Example: Two-class problem $(400, 400)$.
  - One split $(300, 100), (100, 300)$; the other split $(200, 400)$, $(200, 0)$. Same misclassification rate 0.25.
  - Gini index and cross entropy prefer the second split, which is more pure.

Guidelines:

- When growing the tree, Gini index and cross entropy are preferred
- When pruning the tree, typically misclassification rate is used

Missing Values

Missing values for some variables are often encountered in high dimensional data, for example, in gene expression data.

- If each variable has 5% chance missing independently. With 50 variables, probability of missing some variable is as high as 92.3%.

Traditional approaches:

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

How Do Tree-Methods Handle Missing Values?

Two better approaches:

- For categorical variables, create the category "missing",
  - helps to discover that observations with missing values behave differently than those complete observations.
- Construct *surrogate* variables besides the best splitting variable
  - If the primary splitting variable is missing, use surrogate splits in order.

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 is increases significantly. Model becomes more complex.

Advantages of Tree-Based Methods

- 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.

Limitations of Trees

One major problem is their high variance, mainly due to the hierarchical nature of the process.

- Small change in data may result in a very different series of splits, making interpretations somewhat precautious.
- The effect of an error in the top split is propagated down to all the splits below it.

Remedy: Bagging averages many trees to reduce the variance

- Bagging Trees
- Random Forest

Other Problems of Trees

- The lack of smoothness of the prediction surface: it can degrade performance in the regression setting
  - MARS is a modification of CART to alleviate the lack of smoothness;
- The difficulty in modeling additive structure.
  - MARS gives us the tree structure in order to capture additive structure.

R code for Fitting Classification Trees (Iris Data)

```
library(tree)
Iris <- data.frame(rbind(iris3[,,1], iris3[,,2],
   iris3[,,3]), Sp = rep(c("s","c","v"), rep(50,3)))
train = Iris[c(1:25,51:75,101:125),]
test = Iris[-c(1:25,51:75,101:125),]
train=data.frame(train)
test=data.frame(test)
## fit the tree with split="deviance"
tree1=tree(Sp~., data=train, split="deviance")
## tree summary
summary(tree1)
## fit the tree with split="Gini"
tree2=tree(Sp~., data=train, split="gini")
summary(tree2)
```

```
# compute the training error
  train_pred1 <- predict(tree1, type="class")
  train_err1 <- mean(train_pred1!=train[,5])
# compute the test error
  test_pred1 <- predict(tree1, test, type="class")
  test_err1 <- mean(test_pred1!=test[,5])
  print(c(train_err1, test_err1))
# plot the tree
  par(mfrow=c(1,2))
  plot(tree1);
  text(tree1)
  plot(tree2);
  text(tree2)
```