9

Recursive Partitioning and Tree-Based Methods

9.1 Introduction

An algorithm known as *recursive partitioning* is the key to the nonparametric statistical method of *classification and regression trees (CART)* (Breiman, Friedman, Olshen, and Stone, 1984). Recursive partitioning is the step-by-step process by which a *decision tree* is constructed by either splitting or not splitting each node on the tree into two daughter nodes. An attractive feature of the CART methodology (or the related C4.5 methodology; Quinlan, 1993) is that because the algorithm asks a sequence of hierarchical Boolean questions (e.g., is $X_i \leq \theta_j$?, where θ_j is a threshold value), it is relatively simple to understand and interpret the results.

As we described in previous chapters, classification and regression are both supervised learning techniques, but they differ in the way their output variables are defined. For binary classification problems, the output variable, Y, is binary-valued, whereas for regression problems, Y is a continuous variable. Such a formulation is particularly useful when assessing how well a classification or regression methodology does in predicting Y from a given set of input variables $X_1, X_2, \ldots, X_r$.

In the CART methodology, the input space, $\Re^r$, is partitioned into a number of nonoverlapping rectangular ($r = 2$) or cuboid ($r > 2$) regions,

A.J. Izenman, *Modern Multivariate Statistical Techniques*, Springer Texts in Statistics,
DOI 10.1007/978-0-387-78189-1_9, © Springer Science+Business Media New York 2013

each of which is viewed as homogeneous for the purpose of predicting Y. Each region, which has sides parallel to the axes of input space, is assigned a class (in a classification problem) or a constant value (in a regression problem). Such a partition corresponds to a classification or regression tree (as appropriate).

Tree-based methods, such as CART and C4.5, have been used extensively in a wide variety of fields. They have been found especially useful in biomedical and genetic research, marketing, political science, speech recognition, and other applied sciences.

9.2 Classification Trees

A classification tree is the result of asking an ordered sequence of questions, and the type of question asked at each step in the sequence depends upon the answers to the previous questions of the sequence. The sequence terminates in a prediction of the class.

The unique starting point of a classification tree is called the *root node* and consists of the entire learning set $\mathcal{L}$ at the top of the tree. A *node* is a subset of the set of variables, and it can be a terminal or nonterminal node. A *nonterminal* (or *parent*) *node* is a node that splits into two daughter nodes (a *binary split*). Such a binary split is determined by a Boolean condition on the value of a single variable, where the condition is either satisfied ("yes") or not satisfied ("no") by the observed value of that variable. All observations in $\mathcal{L}$ that have reached a particular (parent) node and satisfy the condition for that variable drop down to one of the two daughter nodes; the remaining observations at that (parent) node that do not satisfy the condition drop down to the other daughter node.

A node that does not split is called a *terminal node* and is assigned a class label. Each observation in $\mathcal{L}$ falls into one of the terminal nodes. When an observation of unknown class is "dropped down" the tree and ends up at a terminal node, it is assigned the class corresponding to the class label attached to that node. There may be more than one terminal node with the same class label. A single-split tree with only two terminal nodes is called a *stump*. The set of all terminal nodes is called a *partition* of the data.

Consider a simple example of recursive partitioning involving two input variables, X_1 and X_2. Suppose the tree diagram is given in the top panel of Figure 9.1. The possible stages of this tree are as follows: (1) Is $X_2 \leq \theta_1$? If the answer is yes, follow the left branch; if no, follow the right branch. (2) If the answer to (1) is yes, then we ask the next question: Is $X_1 \leq \theta_2$? An answer of yes yields terminal node τ_1 with corresponding region $R_1 = \{X_1 \leq \theta_2, X_2 \leq \theta_1\}$; an answer of no yields terminal node τ_2 with corresponding region $R_2 = \{X_1 > \theta_2, X_2 \leq \theta_1\}$. (3) If the answer to (1) is

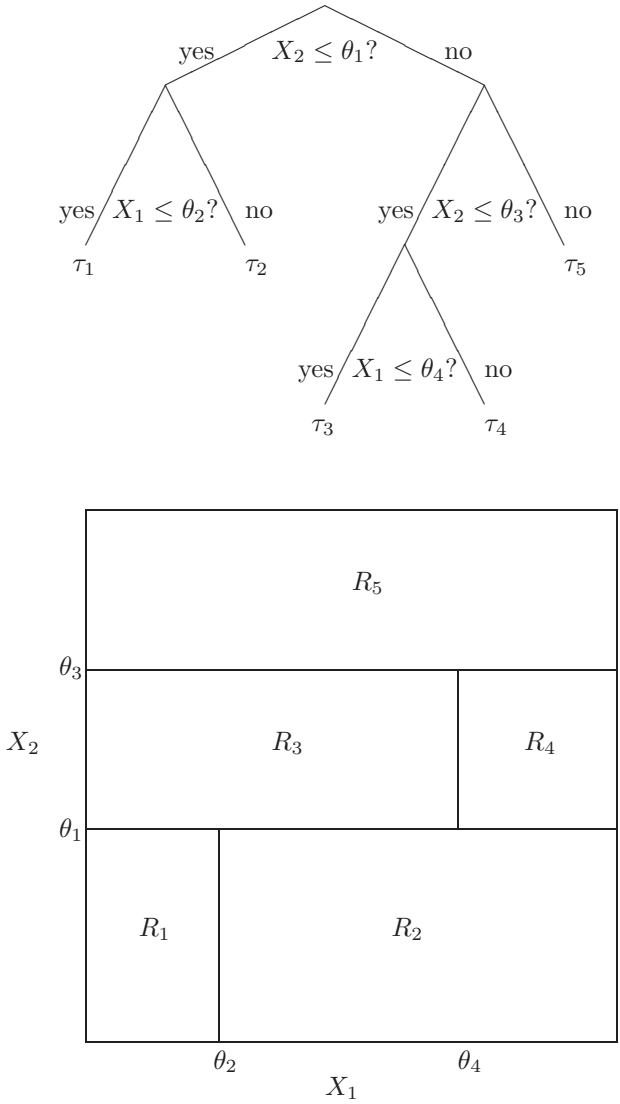

FIGURE 9.1. *Example of recursive partitioning with two input variables X_1 and X_2. Top panel shows a decision tree with five terminal nodes, $\tau_1 - \tau_5$, and four splits. Bottom panel shows the partitioning of $\Re^2$ into five regions, $R_1 - R_5$, corresponding to the five terminal nodes.*

no, we ask the next question: Is $X_2 \leq \theta_3$? If the answer to (3) is yes, then we ask the next question: Is $X_1 \leq \theta_4$? An answer of yes yields terminal node τ_3 with corresponding region $R_3 = \{X_1 \leq \theta_4, \theta_1 < X_2 \leq \theta_3\}$; if no, follow the right branch to terminal node τ_4 with corresponding region $R_4 = \{X_1 > \theta_4, \theta_1 < X_2 \leq \theta_3\}$. (4) If the answer to (3) is no, we arrive at terminal node τ_5 with corresponding region $R_5 = \{X_2 > \theta_3\}$. We have assumed that $\theta_2 < \theta_4$ and $\theta_1 < \theta_3$. The resulting 5-region partition of $\Re^2$ is given in the bottom panel of Figure 9.1. For a classification tree, each terminal node and corresponding region is assigned a class label.

9.2.1 Example: Cleveland Heart-Disease Data

These data[1] were obtained from a heart-disease study conducted by the Cleveland Clinic Foundation (Robert Detrano, principal investigator). For the study, the response variable is `diag` (diagnosis of heart disease: `buff` = healthy, `sick` = heart disease). There were 303 patients in the study, 164 of them healthy and 139 with heart disease.

The 13 input variables are `age` (age in years), `gender` (`male`, `fem`), `cp` (chest-pain type: `angina`=typical angina, `abnang`=atypical angina, `notang` =non-anginal pain, `asympt`=asymptomatic), `trestbps` (resting blood pressure), `chol` (serum cholesterol in mg/dl), `fbs` (fasting blood sugar < 120 mg/dl: `true`, `false`), `restecg` (resting electrocardiographic results: `norm` =normal, `abn`=having ST-T wave abnormality, `hyp`=showing probable or definite left ventricular hypertrophy by Estes's criteria), `thatach` (maximum heart rate achieved), `exang` (exercise-induced angina: `true`, `false`), `oldpeak` (ST depression induced by exercise relative to rest), `slope` (the slope of the peak exercise ST segment: `up`, `flat`, `down`), `ca` (number of major vessels (0–3) colored by flouroscopy), and `thal` (no description given: `norm`=normal, `fix`=fixed defect, `rev`=reversable defect). Of the 303 patients in the original data set, seven had missing data, and so we reduced the number of patients to 296 (160 healthy, 136 with heart disease).

The classification tree is displayed in Figure 9.2 (where we used the entropy measure as the impurity function for splitting). The root node with 296 patients is split according to whether `thal` = `norm` (163 patients) or `thal` = `fix` or `rev` (133 patients). The node with the 163 patients, which consists of 127 healthy patients and 36 patients with heart disease, is then split by whether `ca` < 0.5 (114 patients), or `ca` > 0.5 (49 patients). The node with 114 patients is declared a terminal node for `buff` because of the 102–12 majority in favor of `buff`. The node with 49 patients, which consists

[1]The data can be downloaded from file `cleveland.data` in the UCI repository `archive.ics.uci.edu/ml/datasets/Heart+Disease`.

of 25 healthy patients and 24 with heart disease, is split by whether `cp` = `abnang`, `angina`, `notang` (29 patients) or `cp` = `asympt` (20 patients). The node with 29 patients, which consists of 22 healthy patients and 7 with heart disease, is split by whether `age` $\leq$ 65.5 (7 patients) or `age` $<$ 65.5 (22 patients). The node with 7 patients is declared a terminal node for `buff` because of the 7–0 majority in favor of `buff`, and the node with 22 patients, which consists of 15 healthy patients and 7 with heart disease, is split by whether `age` $<$ 55.5 (13 patients) or `age` $\leq$ 55.5 (9 patients). The node with 13 patients is declared a terminal node for `buff` because of the 12–1 majority in favor of `buff`, and the node with 9 patients is declared a terminal node for `sick` because of the 6–3 majority in favor of `sick`. And so on.

Thus, we see that there are four paths (sequence of splits) through this tree for a patient to be declared healthy (`buff`) and five other paths for a patient to be diagnosed with heart disease (`sick`). In fact, there are 10 splits (and 11 terminal nodes) in this tree. The variables used in the tree construction are `thal`, `ca`, `cp`, `age`, `oldpeak`, `thatach`, and `exang`. The resubstitution (or apparent) error rate (i.e., the error rate obtained directly from the classification tree) is $37/296 = 0.125$ (12 `sick` patients who are classified as `buff` and 25 `buff` patients who are classified as `sick`).

9.2.2 Tree-Growing Procedure

In order to grow a classification tree, we need to answer four basic questions: (1) How do we choose the Boolean conditions for splitting at each node? (2) Which criterion should we use to split a parent node into its two daughter nodes? (3) How do we decide when a node becomes a terminal node (i.e., stop splitting)? (4) How do we assign a class to a terminal node?

9.2.3 Splitting Strategies

At each node, the tree-growing algorithm has to decide on which variable it is "best" to split. We need to consider every possible split over all variables present at that node, then enumerate all possible splits, evaluate each one, and decide which is best in some sense.

For a description of splitting rules, we need to make a distinction between ordinal (or continuous) and nominal (or categorical) variables.

Ordinal or Continuous Variable

For a continuous or ordinal variable, the number of possible splits at a given node is one fewer than the number of its distinctly observed values.

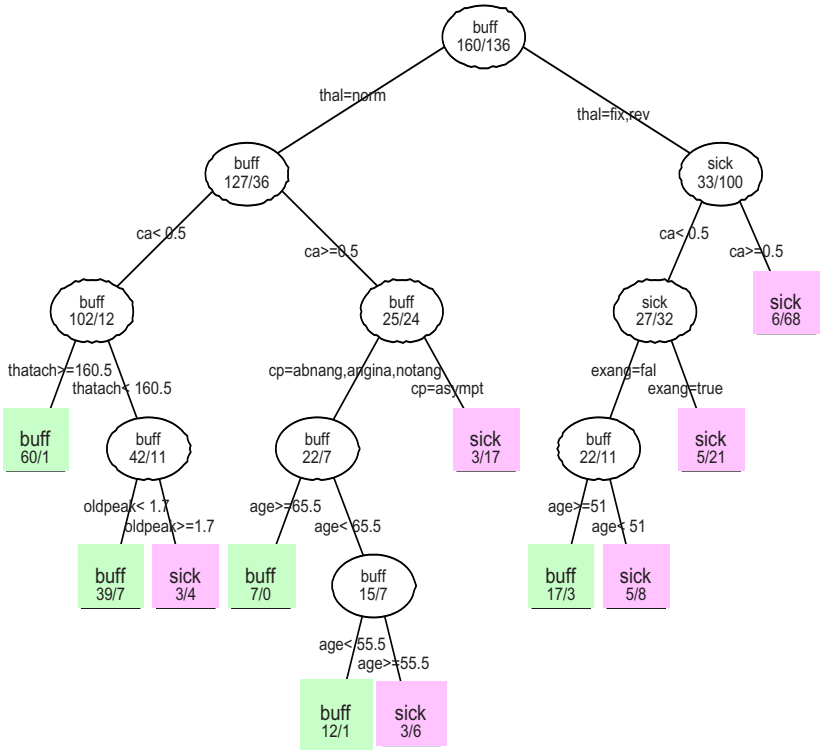

FIGURE 9.2. *Classification tree for the Cleveland heart-disease data, where the entropy measure has been used as the impurity function. The nodes (internal and terminal) are classified as* `buff` *(terminal nodes are colored green) or* `sick` *(terminal nodes are colored pink) according to the majority diagnosis of patients falling into that node. The splitting variables are displayed along the branches.*

In the Cleveland heart-disease data, we have six continuous or ordinal variables: `age` (40 possible splits), `treatbps` (48 possible splits), `chol` (151 possible splits), `thatach` (91 possible splits), `ca` (3 possible splits), and `oldpeak` (39 possible splits). The total number of possible splits from these continuous variables is, therefore, 372.

Nominal or Categorical Variable

Suppose that a particular categorical variable is defined by M distinct categories, $\ell_1, \ldots, \ell_M$. The set $\mathcal{S}$ of possible splits at that node for that variable is the set of all subsets of $\{\ell_1, \ldots, \ell_M\}$. Denote by τ_L and τ_R the left daughter-node and right daughter-node, respectively, emanating from

a (parent) node τ. If we let $M = 4$, then there are $2^M - 2 = 14$ possible splits (ignoring splits where one of the daughter-nodes is empty). However, half of those splits are redundant; for example, the split $\tau_L = \{\ell_1\}$ and $\tau_R = \{\ell_2, \ell_3, \ell_4\}$ is the reverse of the split $\tau_L = \{\ell_2, \ell_3, \ell_4\}$ and $\tau_R = \{\ell_1\}$. So, the set $\mathcal{S}$ of seven distinct splits is given by the following table:

τ_L	τ_R
ℓ_1	ℓ_2, ℓ_3, ℓ_4
ℓ_2	ℓ_1, ℓ_3, ℓ_4
ℓ_3	ℓ_1, ℓ_2, ℓ_4
ℓ_4	ℓ_1, ℓ_2, ℓ_3
ℓ_1, ℓ_2	ℓ_3, ℓ_4
ℓ_1, ℓ_3	ℓ_2, ℓ_4
ℓ_1, ℓ_4	ℓ_2, ℓ_3

In general, there are $2^{M-1} - 1$ distinct splits in $\mathcal{S}$ for an M-categorical variable.

In the Cleveland heart-disease data, there are seven categorical variables: `gender` (1 possible split), `cp` (7 possible splits), `fbs` (1 possible split), `restecg` (3 possible splits), `exang` (1 possible split), `slope` (3 possible splits), and `thal` (3 possible splits). The total number of possible splits from these categorical variables is, therefore, 19.

Total Number of Possible Splits

We now add the number of possible splits from categorical variables (19) to the total number of possible splits from continuous variables (372) to get 391 possible splits over all 13 variables at the root node. In other words, there are 391 possible splits of the root node into two daughter nodes. So, which split is "best"?

Node Impurity Functions

To choose the best split over all variables, we first need to choose the best split for a given variable. Accordingly, we define a measure of goodness of a split.

Let $\Pi_1, \ldots, \Pi_K$ be the $K \geq 2$ classes. For node τ, we define the *node impurity function* $i(\tau)$ as

$$i(\tau) = \phi(p(1|\tau), \cdots, p(K|\tau)), \tag{9.1}$$

where $p(k|\tau)$ is an estimate of $\mathrm{P}(\mathbf{X} \in \Pi_k|\tau)$, the conditional probability that an observation $\mathbf{X}$ is in Π_k given that it falls into node τ. In (9.1),

we require ϕ to be a symmetric function, defined on the set of all K-tuples of probabilities $(p_1, \cdots, p_K)$ with unit sum, minimized at the points $(1, 0, \cdots, 0)$, $(0, 1, 0, \cdots, 0)$, $\ldots$, $(0, 0, \cdots, 0, 1)$ and maximized at the point $(\frac{1}{K}, \cdots, \frac{1}{K})$. In the two-class case ($K = 2$), these conditions reduce to a symmetric $\phi(p)$ maximized at the point $p = 1/2$ with $\phi(0) = \phi(1) = 0$.

One such function ϕ is the *entropy function*,

$$i(\tau) = -\sum_{k=1}^{K} p(k|\tau) \log p(k|\tau), \tag{9.2}$$

which is a discrete version of (7.115). When there are two classes, the entropy function reduces to

$$i(\tau) = -p \log p - (1 - p) \log(1 - p), \tag{9.3}$$

where we set $p = p(1|\tau)$. Several other ϕ-functions have also been suggested, including the *Gini diversity index*,

$$i(\tau) = \sum_{k \neq k'} p(k|\tau) p(k'|\tau) = 1 - \sum_{k} \{p(k|\tau)\}^2. \tag{9.4}$$

In the two-class case, the Gini index reduces to

$$i(\tau) = 2p(1 - p). \tag{9.5}$$

This function can be motivated by considering which quadratic polynomial satisfies the above conditions for the two-class case.

In Figure 9.3, the entropy function and the Gini index are graphed for the two-class case. For practical purposes, there is not much difference between these two types of node impurity functions. The usual default in tree-growing software is the Gini index.

Choosing the Best Split for a Variable

Suppose, at node τ, we apply split s so that a proportion p_L of the observations drops down to the left daughter-node τ_L and the remaining proportion p_R drops down to the right daughter-node τ_R.

For example, suppose we have a data set in which the response variable Y has two possible values, 0 and 1. Suppose that one of the possible splits of the input variable X_j is $X_j \leq c$ vs. $X_j > c$, where c is some value of X_j. We can write down the 2×2 table in Table 9.1.

Consider, first, the parent node τ. We use the entropy function (9.3) as our impurity measure. Estimate p_L by n_{+1}/n_{++} and p_R by n_{+2}/n_{++}, and then the estimated impurity function is

$$i(\tau) = -\left(\frac{n_{+1}}{n_{++}}\right) \log_e \left(\frac{n_{+1}}{n_{++}}\right) - \left(\frac{n_{+2}}{n_{++}}\right) \log_e \left(\frac{n_{+2}}{n_{++}}\right). \tag{9.6}$$

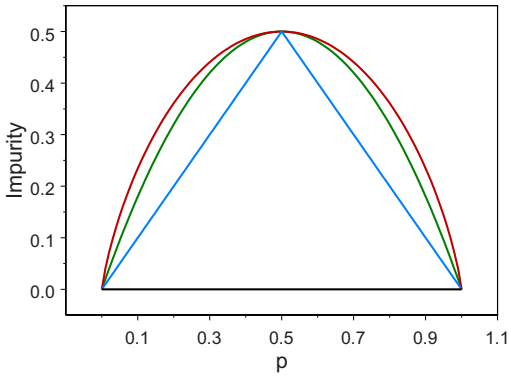

FIGURE 9.3. *Node impurity functions for the two-class case. The entropy function (rescaled) is the red curve, the Gini index is the green curve, and the resubstitution estimate of the misclassification rate is the blue curve.*

Note that $i(\tau)$ is completely independent of the type of proposed split. Now, for the daughter nodes, τ_L and τ_R. For $X_j \leq c$, we estimate p_L by n_{11}/n_{1+} and p_R by n_{12}/n_{1+}, and for $X_j > c$, we estimate p_L by n_{21}/n_{2+} and p_R by n_{22}/n_{2+}. We then compute

$$i(\tau_L) = -\left(\frac{n_{11}}{n_{1+}}\right)\log_e\left(\frac{n_{11}}{n_{1+}}\right) - \left(\frac{n_{12}}{n_{1+}}\right)\log_e\left(\frac{n_{12}}{n_{1+}}\right) \tag{9.7}$$

$$i(\tau_R) = -\left(\frac{n_{21}}{n_{2+}}\right)\log_e\left(\frac{n_{11}}{n_{2+}}\right) - \left(\frac{n_{22}}{n_{2+}}\right)\log_e\left(\frac{n_{22}}{n_{2+}}\right). \tag{9.8}$$

The *goodness of split* s *at node* τ is given by the reduction in impurity gained by splitting the parent node τ into its daughter nodes, τ_R and τ_L,

$$\Delta i(s,\tau) = i(\tau) - p_L i(\tau_L) - p_R i(\tau_R). \tag{9.9}$$

The best split for the single variable X_j is the one that has the largest value of $\Delta i(s,\tau)$ over all $s \in \mathcal{S}_j$, the set of possible distinct splits for X_j.

Example: Cleveland Heart-Disease Data (Continued)

Consider the first variable `age` as a possible splitting variable at the root node. There are 41 different values for `age`, and so there are 40 possible

TABLE 9.1. *Two-by-two table for a split on the variable* X_j*, where the response variable has value 1 or 0.*

	1	0	Row Total
$X_j \leq c$	n_{11}	n_{12}	n_{1+}
$X_j > c$	n_{21}	n_{22}	n_{2+}
Column Total	n_{+1}	n_{+2}	n_{++}

TABLE 9.2. *Two-by-two table for the split on the variable* `age` *in the Cleveland heart disease data: the left branch would be* `age` ≤ 65 *and the right branch would be* `age` > 65.

	Buff	Sick	Row Total
`age` ≤ 65	143	120	263
`age` > 65	17	16	33
Column Total	160	136	296

splits. We set up the 2×2 table, Table 9.2, in which `age` is split, for example, at 65.

Using the two-class entropy function as the impurity measure, we compute (9.7) and (9.8), respectively, for the two possible daughter nodes:

$$i(\tau_L) = -(143/263)\log_e(143/263) - (120/263)\log_e(120/263), \quad (9.10)$$

$$i(\tau_R) = -(17/33)\log_e(17/33) - (16/33)\log_e(16/33), \quad (9.11)$$

whence, $i(\tau_L) = 0.6893$ and $i(\tau_R) = 0.6927$. Furthermore, from (9.6),

$$i(\tau) = -(160/296)\log_e(160/296) - (136/296)\log_e(136/296) = 0.6899. \quad (9.12)$$

Using (9.9), the goodness of this split is given by:

$$\Delta i(s,\tau) = 0.6899 - (263/296)(0.6893) - (33/296)(0.6927) = 0.000162. \quad (9.13)$$

If we repeat these computations for all 40 possible splits for the variable `age`, we arrive at Figure 9.4. In the left panel, we plot $i(\tau_L)$ (blue curve) and $i(\tau_R)$ (red curve) against each of the 40 splits; for comparison, we have the constant value of $i(\tau) = 0.6899$. Note the large drop in the plot of $i(\tau_R)$ at the split `age` > 70. In the right panel, we plot $\Delta i(s,\tau)$ against each of the 40 splits s. The largest value of $\Delta i(s,\tau)$ is 0.04305, which corresponds to the split `age` ≤ 54.

Recursive Partitioning

In order to grow a tree, we start with the root node, which consists of the learning set $\mathcal{L}$. Using the "goodness-of-split" criterion for a single variable, the tree algorithm finds the best split at the root node for each of the variables, X_1 to X_r. The best split s at the root node is then defined as the one that has the largest value of (9.9) over all r single-variable best splits at that node.

In the case of the Cleveland heart-disease data, the best split at the root node (and corresponding value of $\Delta i(s,\tau)$) for each of the 13 variables is listed in Table 9.3. The largest value is 0.147 corresponding to the variable `thal`. So, for these data, the best split at the root node is to split the

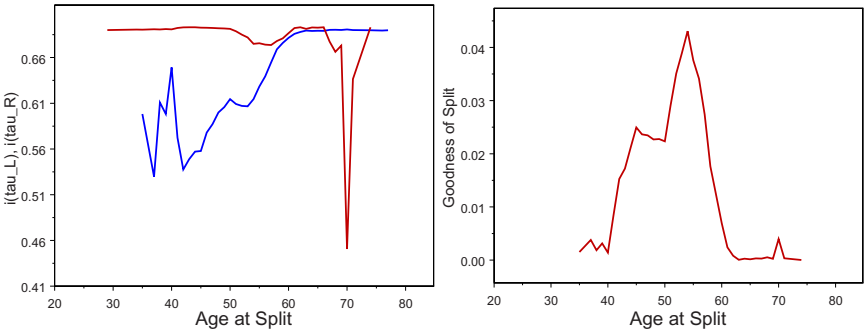

FIGURE 9.4. *Choosing the best split for the* `age` *variable in the Cleveland heart-disease study. The impurity measure is the entropy function. Left panel: Plots of* $i(\tau_L)$ *(blue curve), and* $i(\tau_R)$ *(red curve) against age at split. Note the sharp dip in the* $i(\tau_R)$ *plot at the split* `age` > 70*. Right panel: Plot of the goodness of split* s*,* $\Delta i(s, \tau)$*, against age at split. The peak of this curve corresponds to the split* `age` ≤ 54*.*

variable `thal` according to `norm` vs. (`fix`, `rev`); that is, first separate the 163 normal patients from the 133 patients who have (either fixed or reversible) defects for the variable `thal`.

We next split each of the daughter nodes of the root node in the same way. We repeat the above computations for the left daughter node, except that we consider only those 163 patients having `thal` = `norm`, and then consider the right daughter node, except we consider only those 133 patients having `thal` = `fix` or `rev`. When those splits are completed, we continue to split each of the subsequent nodes. This sequential splitting process of building a tree layer-by-layer is called *recursive partitioning*. If every parent node splits into two daughter nodes, the result is called a *binary tree*. If the binary tree is grown until none of the nodes can be split any further, we say the tree is *saturated*. It is very easy in a high-dimensional classification problem to let the tree get overwhelmingly large, especially if the tree is allowed to grow until saturation.

TABLE 9.3. *Determination of the best split at the root node for the Cleveland heart-disease data. The impurity measure is the entropy function. Each input variable is listed together with its maximum value of* $\Delta i(s, \tau)$ *over all possible splits of that variable.*

`age`	`gender`	`cp`	`trestbps`	`chol`	`fbs`	`restecg`
0.043	0.042	0.133	0.011	0.011	0.00001	0.015
`thatach`	`exang`	`oldpeak`	`slope`	`ca`	`thal`	
0.093	0.093	0.087	0.077	0.124	0.147	

One way to counter this type of situation is to restrict the growth of the tree. This was the philosophy of early tree-growers. For example, we can declare a node to be *terminal* if it fails to be larger than a certain critical size; that is, if $n(\tau) \leq n_{\min}$, where $n(\tau)$ is the number of observations in node τ and $n_{\min}$ is some previously declared minimum size of a node. Because a terminal node cannot be split into daughter nodes, it acts as a brake on tree growth; the larger the value of $n_{\min}$, the more severe the brake. Another early action was to stop a node from splitting if the largest goodness-of-split value at that node is smaller than a certain predetermined limit. These stopping rules, however, do not turn out to be such good ideas. A better approach (Breiman et al., 1984) is to let the tree grow to saturation and then "prune" it back; see Section 9.2.6.

How do we associate a class with a terminal node? Suppose at terminal node τ there are $n(\tau)$ observations, of which $n_k(\tau)$ are from class Π_k, $k = 1, 2, \ldots, K$. Then, the class which corresponds to the largest of the $\{n_k(\tau)\}$ is assigned to τ. This is called the *plurality rule*. This rule can be derived from the Bayes's rule classifier of Section 8.5.1, where we assign the node τ to class Π_i if $p(i|\tau) = \max_k p(k|\tau)$; if we estimate the prior probability π_k by $n_k(\tau)/n(\tau)$, $k = 1, 2, \ldots, K$, then this boils down to the plurality rule.

9.2.4 Example: Pima Indians Diabetes Study

This Indian population lives near Phoenix, Arizona. All patients listed in this data set[2] are females at least 21 years old of Pima Indian heritage. There are two classes: `diabetic`, if the patient shows signs of diabetes according to World Health Organization criteria (i.e., if the 2-hour post-load plasma glucose was at least 200 mg/dl at any survey examination, or if found during routine medical care), and `normal`. In the original data, there were 500 `normal` subjects and 268 `diabetic` subjects.

There are eight input variables: `npregnant` (number of times pregnant), `bmi` (body mass index, (weight in kg)/(height in m)^2), `glucose` (plasma glucose concentration at 2 hours in an oral glucose tolerance test), `pedigree` (diabetes pedigree function), `diastolic.bp` (diastolic blood pressure, mm Hg), `skinfold.thickness` (triceps skin fold thickness, mm), `insulin` (2-hour serum insulin, μU/ml), and `age` (age in years). We removed any subject with a nonsense value of zero for the variables `glucose`, `bmi`, `diastolic.bp`, `skinfold.thickness`; this reduced the data set to 532 patients (from 768), with 355 `normal` subjects and 177 `diabetic` subjects.

[2]These data are available on the book's website (file `pima`) and are also available from the UCI website `archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes`.

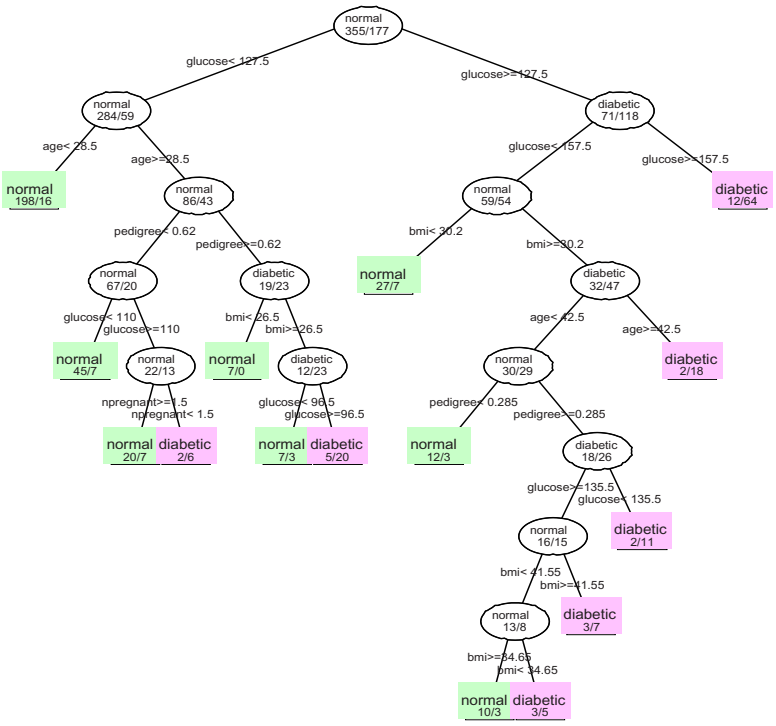

FIGURE 9.5. *A classification tree for the Pima Indians diabetes data, where the impurity measure is the Gini index. The terminal nodes are colored green for* `normal` *and pink for* `diabetic`*. The splitting variables are given on the branches of each split, and the number in each node is given as number of* `normal`*/number of* `diabetic`*, with the node classification given by the majority rule. Nodes were not split further unless they contained at least 10 subjects.*

We also did not use the variable `insulin` because it had so many zeros (374 in the original data).

A classification tree was grown for the Pima Indians diabetes data using Gini's impurity measure (9.5). The classification tree appears in Figure 9.5, where nodes are declared to be terminal if they contain fewer than 10 patients. We see 14 splits and 15 terminal nodes; a patient is declared to be normal at 8 terminal nodes and diabetic at 7 terminal nodes. The assignment of each terminal node into "normal" or "diabetic" depends upon the majority rule at that node; the numbers of normal and diabetic patients in the learning set that fall into each terminal node are displayed at that node.

9.2.5 Estimating the Misclassification Rate

Next, we compute an estimate of the within-node misclassification rate. The resubstitution estimate of the misclassification rate $R(\tau)$ of an observation in node τ is

$$r(\tau) = 1 - \max_k p(k|\tau), \tag{9.14}$$

which, for the two-class case, reduces to

$$r(\tau) = 1 - \max(p, 1-p) = \min(p, 1-p). \tag{9.15}$$

The resubstitution estimate (9.15) in the two-class case is graphed in Figure 9.3 (the blue curve). If $p < 1/2$, the resubstitution estimate increases linearly in p, and if $p > 1/2$, it decreases linearly in p. Because of its poor properties (e.g., nondifferentiability), (9.15) is not used much in practice.

Let T be the tree classifier and let $\widetilde{T} = \{\tau_1, \tau_2, \ldots, \tau_L\}$ denote the set of all terminal nodes of T. We can now estimate the true misclassification rate,

$$R(T) = \sum_{\tau \in \widetilde{T}} R(\tau)P(\tau) = \sum_{\ell=1}^{L} R(\tau_\ell)P(\tau_\ell) \tag{9.16}$$

for T, where $P(\tau)$ is the probability that an observation falls into node τ. If we estimate $P(\tau_\ell)$ by the proportion $p(\tau_\ell)$ of all observations that fall into node τ_ℓ, then, the resubstitution estimate of $R(T)$ is

$$R^{re}(T) = \sum_{\ell=1}^{L} r(\tau_\ell)p(\tau_\ell) = \sum_{\ell=1}^{L} R^{re}(\tau_\ell), \tag{9.17}$$

where $R^{re}(\tau_\ell) = r(\tau_\ell)p(\tau_\ell)$.

Of the 532 subjects in the Pima Indians diabetes study, the classification tree in Figure 9.5 misclassifies 29 of the 355 `normal` subjects as `diabetic`, whereas of the 177 `diabetic` patients, 46 are misclassified as `normal`. So, the resubstitution estimate is $R^{re}(T) = 75/532 = 0.141$.

The resubstitution estimate $R^{re}(T)$, however, leaves much to be desired as an estimate of $R(T)$. First, bigger trees (i.e., more splitting) have smaller values of $R^{re}(T)$; that is, $R^{re}(T') \leq R^{re}(T)$, where T' is formed by splitting a terminal node of T. For example, if a tree is allowed to grow until every terminal node contains only a single observation, then that node is classified by the class of that observation and $R^{re}(T) = 0$. Second, using only the resubstitution estimate tends to generate trees that are too big for the given data. Third, the resubstitution estimate $R^{re}(T)$ is a much-too-optimistic estimate of $R(T)$. More realistic estimates of $R(T)$ are given below.

9.2.6 Pruning the Tree

The Breiman et al. (1984) philosophy of growing trees is to grow the tree "large" and then prune off branches (from the bottom up) until the tree is the "right size." A pruned tree is a *subtree* of the original large tree. How to prune a tree, then, is the crucial part of the process. Because there are many different ways to prune a large tree, we decide which is the "best" of those subtrees by using an estimate of $R(T)$.

The pruning algorithm is as follows:

1. Grow a large tree, say, $T_{\max}$, where we keep splitting until the nodes each contain fewer than $n_{\min}$ observations;
2. Compute an estimate of $R(\tau)$ at each node $\tau \in T_{\max}$;
3. Prune $T_{\max}$ upwards toward its root node so that at each stage of pruning, the estimate of $R(T)$ is minimized.

Instead of using the resubstitution measure $R^{re}(T)$ as our estimate of $R(T)$, we modify it for tree pruning by adopting a regularization approach. Let $\alpha \geq 0$ be a *complexity parameter*. For any node $\tau \in T$, set

$$R_\alpha(\tau) = R^{re}(\tau) + \alpha. \tag{9.18}$$

From (9.18), we define a *cost-complexity pruning measure* for a tree as follows:

$$R_\alpha(T) = \sum_{\ell=1}^{L} R_\alpha(\tau_\ell) = R^{re}(T) + \alpha|\widetilde{T}|, \tag{9.19}$$

where $|\widetilde{T}| = L$ is the number of terminal nodes in the subtree T of $T_{\max}$. Think of $\alpha|\widetilde{T}|$ as a penalty term for tree size, so that $R_\alpha(T)$ penalizes $R^{re}(T)$ for generating too large a tree. For each α, we then choose that subtree $T(\alpha)$ of $T_{\max}$ that minimizes $R_\alpha(T)$:

$$R_\alpha(T(\alpha)) = \min_T R_\alpha(T). \tag{9.20}$$

If $T(\alpha)$ satisfies (9.20), then it is called a *minimizing subtree* (or an *optimally-pruned subtree*) of $T_{\max}$. For any α, there may be more than one minimizing subtree of $T_{\max}$.

The value of α determines the tree size. When α is very small, the penalty term will be small, and so the size of the minimizing subtree $T(\alpha)$, which will essentially be determined by $R^{re}(T(\alpha))$, will be large. For example, suppose we set $\alpha = 0$ and grow the tree $T_{\max}$ so large that each terminal node contains only a single observation; then, each terminal node takes on the class of its solitary observation, every observation is classified correctly, and $R^{re}(T_{\max}) = 0$. So, $T_{\max}$ minimizes $R_0(T)$. As we increase α, the

minimizing subtrees $T(\alpha)$ will have fewer and fewer terminal nodes. When α is very large, we will have pruned the entire tree $T_{\max}$, leaving only the root node.

Note that although α is defined on the interval $[0, \infty)$, the number of subtrees of T is finite. Suppose that, for $\alpha = \alpha_1$, the minimizing subtree is $T_1 = T(\alpha_1)$. As we increase the value of α, T_1 continues to be the minimizing subtree until a certain point, say, $\alpha = \alpha_2$, is reached, and a new subtree, $T_2 = T(\alpha_2)$, becomes the minimizing subtree. As we increase α further, the subtree T_2 continues to be the minimizing subtree until a value of α is reached, $\alpha = \alpha_3$, say, when a new subtree $T_3 = T(\alpha_3)$ becomes the minimizing subtree. This argument is repeated a finite number of times to produce a sequence of minimizing subtrees $T_1, T_2, T_3, \ldots$.

How do we get from $T_{\max}$ to T_1? Suppose the node τ in the tree $T_{\max}$ has daughter nodes τ_L and τ_R, both of which are terminal nodes. Then,

$$R^{re}(\tau) \geq R^{re}(\tau_L) + R^{re}(\tau_R) \tag{9.21}$$

(Breiman et al., 1984, Proposition 4.2). For example, in the classification tree for the Pima Indians diabetes study (Figure 9.5), the lowest subtree has a root node with 13 normals and 8 diabetics, whereas its left daughter node has 10 normals and 3 diabetics and its right daughter node has 3 normals and 5 diabetics. Thus, $R^{re}(\tau) = 8/532 > R^{re}(\tau_L) + R^{re}(\tau_R) = (3+3)/532 = 6/532$. If equality occurs in (9.21) at node τ, then prune the terminal nodes τ_L and τ_R from the tree. Continue this pruning strategy until no further pruning of this type is possible. The resulting tree is T_1.

Next, we find T_2. Let τ be any nonterminal node of T_1, let T_τ be the subtree whose root node is τ, and let $\widetilde{T}_\tau = \{\tau'_1, \tau'_2, \ldots, \tau'_{L_\tau}\}$ be the set of terminal nodes of T_τ. Let

$$R^{re}(T_\tau) = \sum_{\tau' \in \widetilde{T}_\tau} R^{re}(\tau') = \sum_{\ell'=1}^{L_\tau} R^{re}(\tau'_{\ell'}). \tag{9.22}$$

Then, $R^{re}(\tau) > R^{re}(T_\tau)$ (Breiman et al., 1984, Proposition 3.8). For example, from Figure 9.5, let τ be the nonterminal node on the right-hand side of the tree near the center of the tree having 18 normals and 26 diabetics, and let T_τ be the subtree with τ as its root node. Then, $R^{re}(\tau) = 18/532 > R^{re}(T_\tau) = (3+3+3+2)/532 = 11/532$. Now, set

$$R_\alpha(T_\tau) = R^{re}(T_\tau) + \alpha|\widetilde{T}_\tau|. \tag{9.23}$$

As long as $R_\alpha(\tau) > R_\alpha(T_\tau)$, the subtree T_τ has a smaller cost-complexity than its root node τ, and, therefore, it pays to retain T_τ. For the previous example, we retain T_τ as long as $R^{re}_\alpha(\tau) = 18/532 + \alpha > 11/532 + 4\alpha = R^{re}_\alpha(T_\tau)$, or $\alpha < 7/(3 \cdot 532) = 0.0044$.

Substituting (9.18) and (9.23) into this condition and solving for α yields

$$\alpha < \frac{R^{re}(\tau) - R^{re}(T_\tau)}{|\widetilde{T}_\tau| - 1}. \tag{9.24}$$

So, the right-hand side of (9.24), which is positive, computes the reduction in R^{re} (due to going from a single node to the subtree with that node as root) relative to the increase in the number of terminal nodes. For $\tau \in T_1$, define

$$g_1(\tau) = \frac{R^{re}(\tau) - R^{re}(T_{1,\tau})}{|\widetilde{T}_{1,\tau}| - 1}. \quad \tau \notin \widetilde{T}(\alpha_1), \tag{9.25}$$

where $T_{1,\tau}$ is the same as T_τ. Then, $g_1(\tau)$ can be regarded as a critical value for α: as long as $g_1(\tau) > \alpha_1$, we do not prune the nonterminal nodes $\tau \in T_1$.

We define the *weakest-link node* $\widetilde{\tau}_1$ as the node in T_1 that satisfies

$$g_1(\widetilde{\tau}_1) = \min_{\tau \in T_1} g_1(\tau). \tag{9.26}$$

As α increases, $\widetilde{\tau}_1$ is the first node for which $R_\alpha(\tau) = R_\alpha(T_\tau)$, so that $\widetilde{\tau}_1$ is preferred to $T_{\widetilde{\tau}_1}$. Set $\alpha_2 = g_1(\widetilde{\tau}_1)$ and define the subtree $T_2 = T(\alpha_2)$ of T_1 by pruning away the subtree $T_{\widetilde{\tau}_1}$ (so that $\widetilde{\tau}_1$ becomes a terminal node) from T_1.

To find T_3, we find the weakest-link node $\widetilde{\tau}_2 \in T_2$ through the critical value

$$g_2(\tau) = \frac{R^{re}(\tau) - R^{re}(T_{2,\tau})}{|\widetilde{T}_{2,\tau}| - 1}, \quad \tau \in T(\alpha_2),\ \tau \notin \widetilde{T}(\alpha_2), \tag{9.27}$$

where $T_{2,\tau}$ is that part of T_τ which is contained in T_2. We set

$$\alpha_3 = g_2(\widetilde{\tau}_2) = \min_{\tau \in T_2} g_2(\tau), \tag{9.28}$$

and define the subtree T_3 of T_2 by pruning away the subtree $T_{\widetilde{\tau}_2}$ (so that $\widetilde{\tau}_2$ becomes a terminal node) from T_2. And so on for a finite number of steps.

As we noted above, there may be several minimizing subtrees for each α. How do we choose between them? For a given value of α, we call $T(\alpha)$ the *smallest minimizing subtree* if it is a minimizing subtree (i.e., satifies (9.20)) and satisfies the following condition:

$$\text{if } R_\alpha(T) = R_\alpha(T(\alpha)), \text{then } T \succ T(\alpha). \tag{9.29}$$

In (9.29), $T \succ T(\alpha)$ means that $T(\alpha)$ is a subtree of T and, hence, has fewer terminal nodes than T. This condition says that, in the event of any ties, $T(\alpha)$ is taken to be the smallest tree out of all those trees that minimize

R_α. Breiman et al. (1984, Proposition 3.7) showed that for every α, there exists a unique smallest minimizing subtree.

The above construction gives us a finite increasing sequence of complexity parameters,

$$0 = \alpha_0 < \alpha_1 < \alpha_2 < \alpha_3 < \cdots < \alpha_M, \tag{9.30}$$

which corresponds to a finite sequence of nested subtrees of $T_{\max}$,

$$T_{\max} = T_0 \succ T_1 \succ T_2 \succ T_3 \succ \cdots \succ T_M, \tag{9.31}$$

where $T_k = T(\alpha_k)$ is the unique smallest minimizing subtree for $\alpha \in [\alpha_k, \alpha_{k+1})$, and T_M is the root-node subtree. We start with T_1 and increase α until $\alpha = \alpha_2$ determines the weakest-link node $\widetilde{\tau}_1$; we then prune the subtree $T_{\widetilde{\tau}_1}$ with that node as root. This gives us T_2. We repeat this procedure by finding $\alpha = \alpha_3$ and the weakest-link node $\widetilde{\tau}_2$ in T_2 and prune the subtree $T_{\widetilde{\tau}_2}$ with that node as root. This gives us T_3. This pruning process is repeated until we arrive at T_M.

Example: Pima Indians Diabetes Study (Continued)

The sequence of seven pruned classification trees, T_k, corresponding to their critical values, α_k, are listed in Table 9.4. The tree displayed in Figure 9.5 has 14 splits (and, hence, 15 terminal nodes).

Any value of $\alpha < 0.0038$ will produce a tree with 15 terminal nodes. When $\alpha = 0.0038$, the classification tree is pruned to have 11 splits (and 12 terminal nodes), which will remain the same for all $0.0038 \leq \alpha < 0.0047$. Increasing α to 0.0047 prunes the tree to 9 splits (and 10 terminal nodes). And so on, until α is increased above 0.0883 when the tree consists only of the root node.

9.2.7 Choosing the Best Pruned Subtree

Thus far, we have constructed a finite sequence of decreasing-size subtrees $T_1, T_2, T_3, \ldots, T_M$ by pruning more and more nodes from $T_{\max}$. When do we stop pruning? Which subtree of the sequence do we choose as the "best" pruned subtree?

Choice of the best subtree depends upon having a good estimate of the misclassification rate $R(T_k)$ corresponding to the subtree T_k. Breiman et al. (1984) offered two estimation methods: use an independent test sample or use cross-validation. When the data set is very large, use of an independent test set is straightforward and computationally efficient, and is, generally, the preferred estimation method. For smaller data sets, cross-validation is preferred.

TABLE 9.4. *Pruned classification trees for the Pima Indians diabetes study. The impurity function is the Gini index. By increasing the complexity parameter α, seven classification trees, T_k, $k = 1, 2, \ldots, 6$, are derived, where the tree details are listed so that $T_k \succ T_{k+1}$; i.e., largest tree to smallest tree. Also listed for each tree are the number of terminal nodes ($|\widetilde{T}_k|$), resubstitution error (R^{re}), and 10-fold cross-validation (CV) error ($R^{CV/10}$). The $\pm$ values on the CV error are the CV standard errors ($\widehat{SE}$). The CV error estimate and its estimated standard error produce random values according to the random CV-partition of the data.*

k	α_k	$\lvert\widetilde{T}_k\rvert$	$R^{re}(T_k)$	$R^{CV/10}(T_k)$
1		15	0.141	0.258 $\pm$ 0.019
2	0.0038	12	0.152	0.233 $\pm$ 0.018
3	0.0047	10	0.162	0.233 $\pm$ 0.018
4	0.0069	6	0.190	0.235 $\pm$ 0.018
5	0.0085	4	0.207	0.256 $\pm$ 0.019
6	0.0188	2	0.244	0.256 $\pm$ 0.019
7	0.0883	1	0.333	0.333 $\pm$ 0.020

Independent Test Set

Randomly assign the observations in the data set $\mathcal{D}$ into a learning set $\mathcal{L}$ and a test set $\mathcal{T}$, where $\mathcal{D} = \mathcal{L} \cup \mathcal{T}$ and $\mathcal{L} \cap \mathcal{T} = \emptyset$. Suppose there are $n_{\mathcal{T}}$ observations in the test set and that they are drawn independently from the same underlying distribution as the observations in $\mathcal{L}$. Grow the tree $T_{\max}$ from the learning set only, prune it from the bottom up to give the sequence of subtrees $T_1 \succ T_2 \succ T_3 \succ \cdots \succ T_M$, and assign a class to each terminal node.

Take each of the $n_{\mathcal{T}}$ test-set observations and drop it down the subtree T_k. Each observation in $\mathcal{T}$ is then classified into one of the different classes. Because the true class of each observation in $\mathcal{T}$ is known, we estimate $R(T_k)$ by $R^{ts}(T_k)$, which is (9.19) with $\alpha = 0$; that is, $R^{ts}(T_k) = R^{re}(T_k)$, the resubstitution estimate computed using the independent test set. When the costs of misclassification are identical for each class, $R^{ts}(T_k)$ is the proportion of all test set observations that are misclassified by T_k. These estimates are then used to select the best-pruned subtree T_* by the rule

$$R^{ts}(T_*) = \min_k R^{ts}(T_k), \tag{9.32}$$

and $R^{ts}(T_*)$ is its estimated misclassification rate.

We estimate the standard error of $R^{ts}(T)$ as follows. When we drop the test set $\mathcal{T}$ down a tree T, the chance that we misclassify any one of those observations is $p^* = R(T)$. Thus, we have a binomial sampling situation with $n_{\mathcal{T}}$ Bernoulli trials and probability of success p^*. If $p = R^{ts}(T)$ is

the proportion of misclassified observations in $\mathcal{T}$, then, p is unbiased for p^* and the variance of p is $p^*(1-p^*)/n_{\mathcal{T}}$. The standard error of $R^{ts}(T)$ is, therefore, estimated by

$$\widehat{\mathrm{SE}}(R^{ts}(T)) = \left\{ \frac{R^{ts}(T)(1-R^{ts}(T))}{n_{\mathcal{T}}} \right\}^{1/2}. \tag{9.33}$$

Cross-Validation

In V-fold cross-validation (CV/V), we randomly divide the data $\mathcal{D}$ into V roughly equal-size, disjoint subsets, $\mathcal{D} = \bigcup_{v=1}^{V} \mathcal{D}_v$, where $\mathcal{D}_v \cap \mathcal{D}_{v'} = \emptyset$, $v \neq v'$, and V is usually taken to be 5 or 10. We next create V different data sets from the $\{\mathcal{D}_v\}$ by taking $\mathcal{L}_v = \mathcal{D} - \mathcal{D}_v$ as the vth learning set and $\mathcal{T}_v = \mathcal{D}_v$ as the vth test set, $v = 1, 2, \ldots, V$. If the $\{\mathcal{D}_v\}$ each have the same number of observations, then each learning set will have $(\frac{V-1}{V}) \times 100$ percent of the original data set.

Grow the vth "auxilliary" tree $T_{\max}^{(v)}$ using the vth learning set $\mathcal{L}_v$, $v = 1, 2, \ldots, V$. Fix the value of the complexity parameter α. Let $T^{(v)}(\alpha)$ be the best pruned subtree of $T_{\max}^{(v)}$, $v = 1, 2, \ldots, V$. Now, drop each observation in the vth test set $\mathcal{T}_v$ down the tree $T^{(v)}(\alpha)$, $v = 1, 2, \ldots, V$. Let $n_{ij}^{(v)}(\alpha)$ denote the number of jth class observations in $\mathcal{T}_v$ that are classified as being from the ith class, $i, j = 1, 2, \ldots, K$, $v = 1, 2, \ldots, V$. Because $\mathcal{D} = \bigcup_{v=1}^{V} \mathcal{T}_v$ is a disjoint sum, the total number of jth class observations that are classified as being from the ith class is $n_{ij}(\alpha) = \sum_{v=1}^{V} n_{ij}^{(v)}(\alpha)$, $i, j = 1, 2, \ldots, K$. If we set n_j to be the number of observations in $\mathcal{D}$ that belong to the jth class, $j = 1, 2, \ldots, K$, and assume that misclassification costs are equal for all classes, then, for a given α,

$$R^{CV/V}(T(\alpha)) = n^{-1} \sum_{i=1}^{K} \sum_{j=1}^{K} n_{ij}(\alpha) \tag{9.34}$$

is the estimated misclassification rate over $\mathcal{D}$, where $T(\alpha)$ is a minimizing subtree of $T_{\max}$.

The final step in this process is to find the right-sized subtree. Breiman et al. (1984, p. 77) recommend evaluating (9.34) at the sequence of values $\alpha'_k = \sqrt{\alpha_k \alpha_{k+1}}$, where α'_k is the geometric midpoint of the interval $[\alpha_k, \alpha_{k+1})$ in which $T(\alpha) = T_k$. Set

$$R^{CV/V}(T_k) = R^{CV/V}(T(\alpha'_k)). \tag{9.35}$$

Then, select the best-pruned subtree T_* by the rule:

$$R^{CV/V}(T_*) = \min_k R^{CV/V}(T_k), \tag{9.36}$$

and use $R^{CV/V}(T_*)$ as its estimated misclassification rate.

Deriving an estimated standard error of the cross-validated estimate of the misclassification rate is more complicated than using a test set. The usual way of sidestepping issues of non-independence of the summands in (9.29) is to ignore them and pretend instead that independence holds. Actually, this approximation appears to work well in practice. See Breiman et al. (1984, Section 11.5) for details.

It is usual to take $V = 10$ for 10-fold CV. The leave-one-out CV method (i.e., $V = n$) is not recommended because the resulting auxilliary trees will be almost identical to the tree constructed from the full data set, and so nothing would be gained from this procedure.

The One-SE Rule

To overcome possible instability in selecting the best-pruned subtree, Breiman et al. (1984, Section 3.4.3) propose an alternative rule.

Let $\widehat{R}(T_*) = \min_k R(T_k)$ denote the estimated misclassification rate, calculated from either a test set (i.e., $R^{ts}(T_*)$) or cross-validation (i.e., $R^{CV/V}(T_*)$). Then, we choose the smallest tree T_{**} that satisfies the "1-SE rule," namely,

$$\widehat{R}(T_{**}) \leq \widehat{R}(T_*) + \widehat{\mathrm{SE}}(\widehat{R}(T_*)). \tag{9.37}$$

This rule appears to produce a better subtree than using T_* because it responds to the variability (through the standard error) of the cross-validation estimates.

Example: Pima Indians Diabetes Study (Continued)

For example, we apply the 1-SE rule to the Pima Indians diabetes study. From Table 9.4, the 1-SE rule yields a minimum of CV error + SE = 0.233 + 0.018 = 0.251, which leads to the choice of a classification tree with 9 splits (10 terminal nodes) based upon cross-validation. The corresponding pruned classification tree is displayed in Figure 9.6.

A diagnosis of diabetes is given to those subjects who have one of the following symptoms:

1. plasma glucose level at least 157.5;
2. plasma glucose level between 127.5 and 157.5, bmi at least 30.2, and age at least 42.5 years;
3. plasma glucose level between 127.5 and 157.6, bmi at least 30.2, age less than 42.5 years, and a pedigree at least 0.285;
4. plasma glucose level between 96.5 and 127.5, age at least 28.5 years, a pedigree at least 0.62, and bmi at least 26.5.

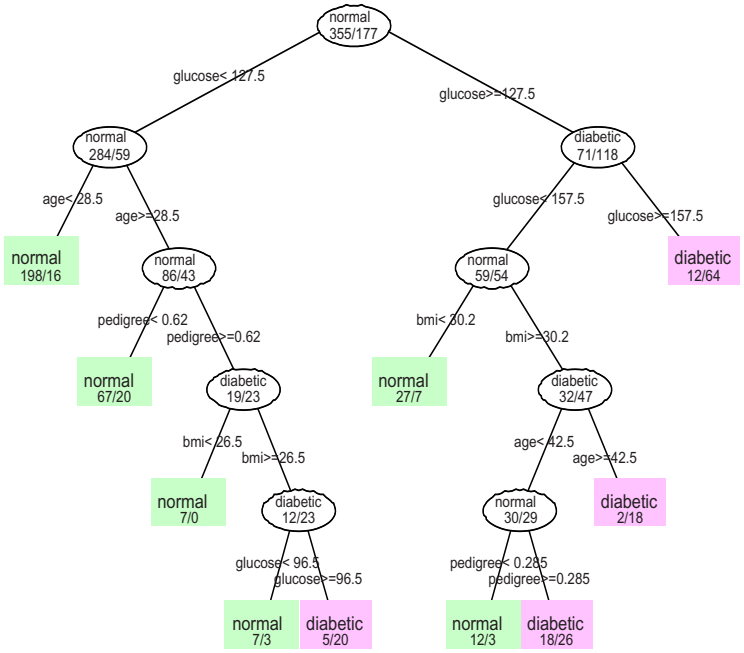

FIGURE 9.6. *A pruned classification tree for the Pima Indians diabetes data, with 9 splits and 10 terminal nodes, where the impurity measure is the Gini index. The terminal nodes are colored green for* `normal` *and pink for* `diabetic`.

This tree has a resubstitution error rate of $86/532 = 0.162$ and 10-fold CV misclassification rate of 0.233 ± 0.018.

9.2.8 Example: Vehicle Silhouettes

Consider the `vehicle` data[3] of Section 8.7, which were collected to study how well 3D objects could be distinguished by their 2D silhouette images. There are four classes of objects, each of which was a CORGI model vehicle: an Opel Manta car (`opel`, 212 images), a Saab 9000 car (`saab`, 217 images), a double-decker bus (`bus`, 218 images), and a Chevrolet van (`van`, 199 images), giving a total of 846 images. Each object was viewed by a camera from many different angles and elevations. The variables are `scaled variance`, `skewness`, and `kurtosis` about the major/minor axes, and

[3]These data can be found in the UCI Machine Learning Repository website `archive.ics.uci.edu/ml/datasets/Statlog+(Vehicle+Silhouettes)`.

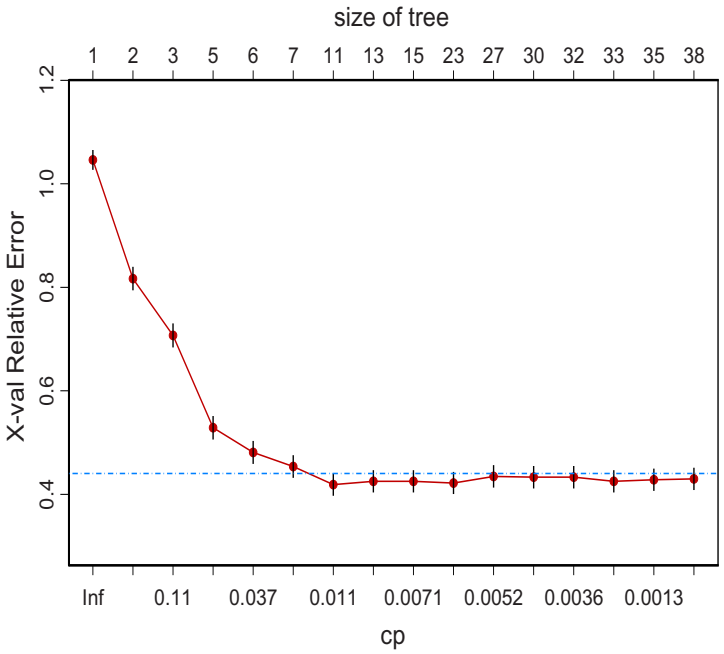

FIGURE 9.7. *Plot of 10-fold CV results of different size classification trees for the* `vehicle` *data. The cp-value is* α *divided by the resubstitution error rate estimate,* $R^{re}(T_0) = 628/846 = 0.742$*, for the root tree, and the vertical axis is the corresponding CV error rate also divided by* $R^{re}(T_0)$*. The vertical lines indicate* $\pm$ *two SE for each CV error estimate. The recommended tree size has cp equal to the smallest tree with the minimum CV error; in this case, 11 terminal nodes.*

heuristic measures such as `hollows ratio`, `circularity`, `elongatedness`, `rectangularity`, and `compactness` of the silhouettes.

Based upon the One–SE rule, and the resulting complexity-parameter plot in Figure 9.7, the most appropriate classification tree has 10 splits with 11 terminal nodes, with a resubstitution error rate of $0.3535 \times 0.74232 = 0.262$, and CV error rate of 0.299 ± 0.0157. In Figure 9.8, we have displayed the pruned classification tree with 10 splits and 11 terminal nodes.

9.3 Regression Trees

Suppose $\mathcal{L} = \{(\mathbf{x}_i, y_i), i = 1, 2, \ldots, n\}$, where the y_i are measurements made on a continuous response variable Y, and the $\mathbf{x}_i$ are measurements

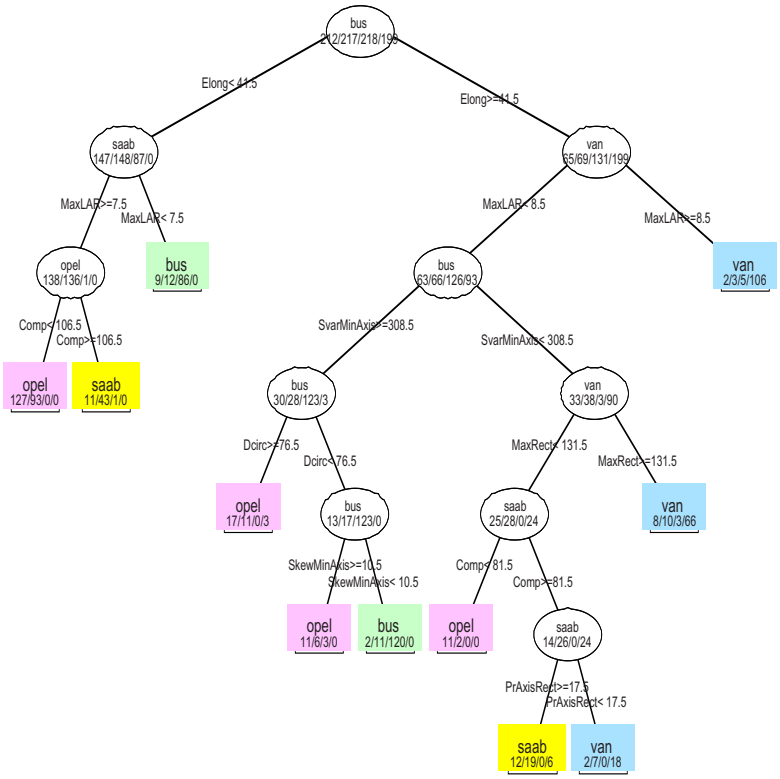

FIGURE 9.8. *A pruned classification tree for the* `vehicle` *data. There are 12 input variables, 846 observations, and four classes of vehicle models: opel (pink), saab (yellow), bus (green), and van (blue), whose numbers at each node are given by a/b/c/d, respectively, There are 10 splits and 11 terminal nodes in this tree. The resubstitution error rate is 0.262.*

on an input r-vector $\mathbf{X}$. We assume that Y is related to $\mathbf{X}$ as in multiple regression (see Chapter 5), and we wish to use a tree-based method to predict Y from $\mathbf{X}$.

Regression trees are constructed in a similar way as are classification trees, and the method is generally referred to as *recursive-partitioning regression*. In a classification tree, the class of a terminal node is defined as that class that commands a plurality (a majority in the two-class case) of all the observations in that node, where ties are decided at random. In a regression tree, the output variable is set to have the constant value $Y(\tau)$ at terminal node τ. Hence, the tree can be represented as an r-dimensional histogram estimate of the regression surface, where r is the number of input variables, $X_1, X_2, \ldots, X_r$.

9.3.1 The Terminal-Node Value

How do we find $y(\tau)$? Recall (from Chapter 5) that the resubstitution estimate of prediction error is

$$R^{re}(\widehat{\mu}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2, \tag{9.38}$$

where $\widehat{y}_i = \widehat{\mu}(\mathbf{x}_i)$ is the estimated value of the predictor at $\mathbf{x}_i$. For $\widehat{y}_i$ to be constant at each node, the predictor has to have the form

$$\widehat{\mu}(\mathbf{x}) = \sum_{\tau\in\widetilde{T}} y(\tau) I_{[\mathbf{x}\in\tau]} = \sum_{\ell=1}^{L} y(\tau_\ell) I_{[\mathbf{x}\in\tau_\ell]}, \tag{9.39}$$

where $I_{[\mathbf{x}\in\tau_\ell]}$ is equal to one if $\mathbf{x} \in \tau_\ell$ and zero otherwise. For $\mathbf{x}_i \in \tau_\ell$, $R^{re}(\widehat{\mu})$ is minimized by taking $\widehat{y}_i = \bar{y}(\tau_\ell)$ as the constant value $y(\tau_\ell)$, where $\bar{y}(\tau_\ell)$ is the average of the $\{y_i\}$ for all observations assigned to node τ_ℓ; that is,

$$\bar{y}(\tau_\ell) = \frac{1}{n(\tau_\ell)} \sum_{\mathbf{x}_i\in\tau_\ell} y_i, \tag{9.40}$$

where $n(\tau_\ell)$ is the total number of observations in node τ_ℓ. Changing notation slightly to reflect the tree structure, the resubstitution estimate is

$$R^{re}(T) = \frac{1}{n}\sum_{\ell=1}^{L}\sum_{\mathbf{x}_i\in\tau_\ell}(y_i - \bar{y}(\tau_\ell))^2 = \sum_{\ell=1}^{L} R^{re}(\tau_\ell), \tag{9.41}$$

where

$$R^{re}(\tau_\ell) = \frac{1}{n}\sum_{\mathbf{x}_i\in\tau_\ell}(y_i - \bar{y}(\tau_\ell))^2 = p(\tau_\ell)s^2(\tau_\ell), \tag{9.42}$$

$s^2(\tau_\ell)$ is the (biased) sample variance of all the y_i values in node τ_ℓ, and $p(\tau_\ell) = n(\tau_\ell)/n$ is the proportion of observations in node τ_ℓ. Hence, $R^{re}(T) = \sum_{\ell=1}^{L} p(\tau_\ell)s^2(\tau_\ell)$.

9.3.2 Splitting Strategy

How do we determine the type of split at any given node of the tree? We take as our splitting strategy at node $\tau \in \widetilde{T}$ the split that provides the biggest reduction in the value of $R^{re}(T)$. The reduction in $R^{re}(\tau)$ due to a split into τ_L and τ_R is given by

$$\Delta R^{re}(\tau) = R^{re}(\tau) - R^{re}(\tau_L) - R^{re}(\tau_R); \tag{9.43}$$

the best split at τ is then the one that maximizes $\Delta R^{re}(\tau)$. The result of employing such a splitting strategy is that the best split will divide up observations according to whether Y has a small or large value; in general, where splits occur, we see either $\bar{y}(\tau_L) < \bar{y}(\tau) < \bar{y}(\tau_R)$ or its reverse with $\bar{y}(\tau_L)$ and $\bar{y}(\tau_R)$ interchanged.

We note that finding τ_L and τ_R to maximize $\Delta R^{re}(\tau)$ is equivalent to minimizing $R^{re}(\tau_L) + R^{re}(\tau_R)$. From (9.42), this boils down to finding τ_L and τ_R to solve

$$\min_{\tau_L, \tau_R} \{p(\tau_L)s^2(\tau_L) + p(\tau_R)s^2(\tau_R)\}, \tag{9.44}$$

where $p(\tau_L)$ and $p(\tau_R)$ are the proportions of observations in τ that split to τ_L and τ_R, respectively.

9.3.3 Pruning the Tree

The method for pruning a regression tree incorporates the same ideas as is used to prune a classification tree.

As before, we first grow a large tree, $T_{\max}$, by splitting nodes repeatedly until each node contains fewer than a given number of observations; that is, until $n(\tau) \leq n_{\min}$ for each $\tau \in \widetilde{T}$, where we typically set $n_{\min} = 5$.

Next, we set up an *error-complexity measure*,

$$R_\alpha(T) = R^{re}(T) + \alpha|\widetilde{T}|, \tag{9.45}$$

where $\alpha \geq 0$ is a complexity parameter. Use $R_\alpha(T)$ as the criterion for deciding when and how to split, just as we did in pruning classification trees. The result is a sequence of subtrees,

$$T_{\max} = T_0 \succ T_1 \succ T_2 \succ T_3 \succ \cdots \succ T_M, \tag{9.46}$$

and an associated sequence of complexity parameters,

$$0 = \alpha_0 < \alpha_1 < \alpha_2 < \alpha_3 < \cdots < \alpha_M, \tag{9.47}$$

such that for $\alpha \in [\alpha_k, \alpha_{k+1})$, T_k is the smallest minimizing subtree of $T_{\max}$.

9.3.4 Selecting the Best Pruned Subtree

We estimate $R(T_k)$ by using an independent test set or by cross-validation. The details follow those in Section 9.2.6.

For an independent test set, $\mathcal{T}$, an estimate of $R(T_k)$ is given by

$$R^{ts}(T_k) = \frac{1}{n_{\mathcal{T}}} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{T}} (y_i - \widehat{\mu}_k(\mathbf{x}_i))^2, \tag{9.48}$$

where $n_{\mathcal{T}}$ is the number of observations in the test set and $\widehat{\mu}_k(\mathbf{x})$ is the estimated prediction function associated with subtree T_k.

For a V-fold cross-validated estimate of $R(T_k)$, we first construct the minimal error-complexity subtrees $T^{(v)}(\alpha)$, $v = 1, 2, \ldots, V$, parameterized by α. Set $\alpha'_k = \sqrt{\alpha_k \alpha_{k+1}}$ and let $\widehat{\mu}_k^{(v)}(\mathbf{x})$ denote the estimated prediction function associated with the subtree $T^{(v)}(\alpha'_k)$. The V-fold CV estimate of $R(T_k)$ is given by

$$R^{CV/V}(T_k) = n^{-1} \sum_{v=1}^{V} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{T}_v} (y_i - \widehat{\mu}_k^{(v)}(\mathbf{x}_i))^2. \tag{9.49}$$

We usually select $V = 10$ for a 10-fold CV estimate in which we split the learning set into 10 subsets, use 9 of those 10 subsets to grow and prune the tree, and then use the omitted subset to test the results of the tree.

Given the sequence of subtrees $\{T_k\}$, we select the smallest subtree T_{**} for which

$$\widehat{R}(T_{**}) \leq \widehat{R}(T_*) + \widehat{\mathrm{SE}}(\widehat{R}(T_*)), \tag{9.50}$$

where $\widehat{R}(T_*) = \min_k \widehat{R}(T_k)$ is the estimated prediction error calculated using using either an independent test set (i.e., $R^{ts}(T_*)$) or cross-validation (i.e., $R^{CV/V}(T_*)$).

9.3.5 Example: 1992 Major League Baseball Salaries

As an example of a regression tree, we use data on the salaries of Major League Baseball (MLB) players for 1992 (Watnik, 1998).[4] The data consist of $n = 337$ MLB players who played at least one game in both the 1991 and 1992 seasons, excluding pitchers. The interesting aspect of these data is that a player's "value" is judged by his performance measures, which in turn could be used to determine his salary the next year or possibly to enable him to change his employer.

The output variable is the 1992 salaries (in thousands of dollars) of these players, and the input variables are the following performance measures from 1991: `BA` (batting average), `OBP` (on-base percentage), `Runs` (number of runs scored), `Hits` (number of hits), `2B` (number of doubles), `3B` (number of triples), `HR` (number of home runs), `RBI` (number of runs batted in), `BB` (number of bases on balls or walks), `SO` (number of strikeouts), `SB` (number of stolen bases), and `E` (number of errors made). Also included as input

[4]These data can be found at the website of the *Journal of Statistics Education*, `www.amstat.org/publications/jse/jse_data_archive.html`. Sources for these data are *CNN/Sports Illustrated*, *Sacramento Bee* (15th October 1991), *The New York Times* (19th November 1992), and the Society for American Baseball Research.

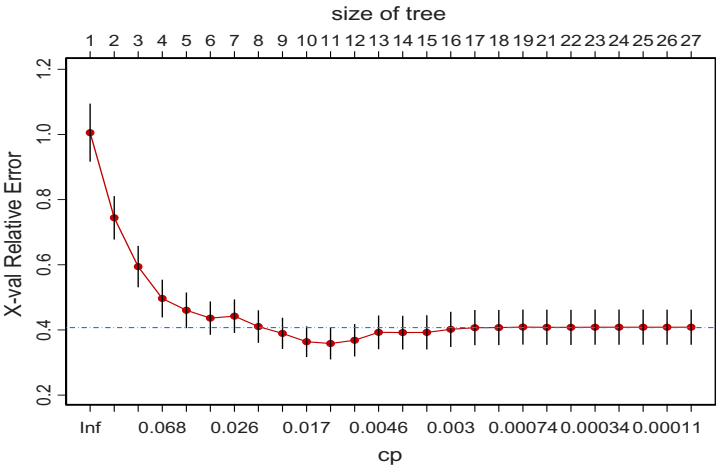

FIGURE 9.9. *Plot of 10-fold CV results of different size regression trees for 1992 baseball salary data. The cp-value is* α *divided by the resubstitution estimate,* $R^{re}(T_0)$*, for the root tree, and the vertical axis is the CV error also divided by* $R^{re}(T_0)$*. The vertical lines indicate* $\pm$ *two SE for each CV error estimate. The recommended amount of pruning is to set cp equal to the smallest tree with the minimum CV error; in this case, 11 terminal nodes.*

variables are the following four indicator variables: `FAE` (indicator of free-agent eligibility), `FA` (indicator of free agent in 1991/92), `AE` (indicator of arbitration eligibility), `A` (indicator of arbitration in 1991/92). These four variables indicated how free each player was to move to other teams. A player's `BA` is the ratio of number of hits to the total number of "at-bats" for that player (whether resulting in a hit or an out). The `OBP` is the ratio of number of hits plus the number of walks to the number of hits plus the number of walks plus the number of outs. For reference, a BA above 0.3 is very good, and an OBP above 0.4 is excellent. An RBI occurs when a runner scores as a direct result of a player's at-bat.

The plot of the CV results for this example is given in Figure 9.9, where the minimum value of the CV error occurs for a tree size of 10 terminal nodes. The pruned regression tree with 10 splits and 11 terminal nodes corresponding to the minimum 1–SE rule is given in Figure 9.10. We see from the terminal node on the right-hand side of the tree that the 14 players who score at least 46.5 runs have at least 94.5 RBIs, and are eligible for free-agency to earn the highest average salary ($3,897,214). The lowest average salary ($232,898), which is made by 108 players, is located at the terminal node on the left-hand side of the tree. We also see that performing well on at least one measure produces substantial differences in average salary. The resubstitution estimate (9.41) of prediction error for

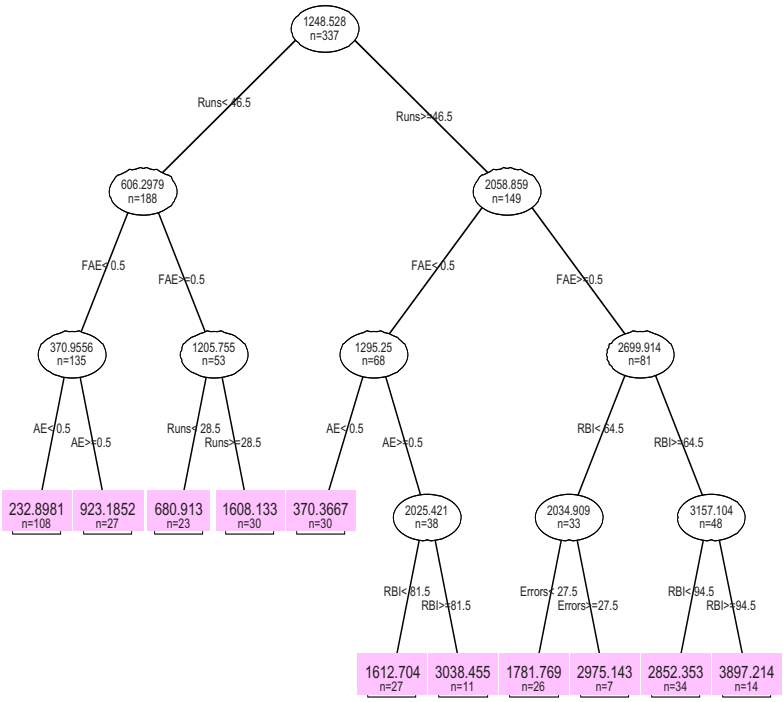

FIGURE 9.10. *Pruned regression tree for 1992 baseball salary data. The label of each node indicates the mean* `salary`*, in thousands of dollars, for the number* n *of players who fall into that node.*

this regression tree is $R^{re}(T) = \$341,841$, the cross-validation estimate of prediction error is \$549,217, and the cross-validation standard deviation is \$74,928. By comparison, regressing `Salary` on the 15 input variables in a multiple regression yields a residual sum of squares of \$155,032,181 and a residual mean square of \$482,966 based upon 321 df.

9.4 Extensions and Adjustments

9.4.1 Multivariate Responses

Some work has been carried out on constructing classification trees for multivariate responses, especially where each response is binary (Zhang, 1998). In such cases, the measure of within-node homogeneity at node τ for a single binary variable is generalized to a scalar-valued function of a matrix argument. Examples include $-\log|\mathbf{V}_\tau|$, where $\mathbf{V}_\tau$ is the within-node sample covariance matrix of the s binary responses at node τ, and

a node-based quadratic form in $\mathbf{V}$, the covariance matrix derived from the root node. The cost-complexity of tree T is then defined as $R_\alpha(T)$ in (9.19), where $R^{re}(T)$ is a within-node homogeneity measure summed over all terminal nodes. When dealing with multivariate responses, it is clear from an applied point of view that the amount of data available for tree construction has to be very large.

9.4.2 Survival Trees

Tree-based methods for analyzing censored survival data have become very useful tools in biomedical research, where they can identify prognostic factors for predicting survival (see, e.g., Intrator and Kooperberg, 1995). The resulting trees are called *survival trees* (or *conditional inference trees*). Survival data usually take the form of time-to-death but can be more general than that, such as time to a particular event to occur. Censored survival data occur when patients live past the conclusion of the study, leave the study prematurely, or die during the period of the study from a disease not connected to the one being studied, and survival analysis has to take such conditions into account in the inference process.

When using tree-based methods to analyze censored survival data, it is necessary to choose a criterion for making splitting decisions. There are several splitting criteria, which can be divided into two types depending upon whether one prefers to use a "within-node homogeneity" measure or a "between-node heterogeneity" measure. Most applications of the former method (see, e.g., Davis and Anderson, 1989) are parametrically based; they typically incorporate a version of minus the log-likelihood loss function, where the versions differ in the loss function used and, thus, how they represent the model for the observed data likelihood within the nodes.

The first application of recursive partitioning to the analysis of censored survival data (Gordon and Olshen, 1985) used a more nonparametric approach, basing their tree-construction on within-node Kaplan-Meier estimates of the survival distribution, and then comparing those curve estimates to within-node Kaplan-Meier estimates of truly homogeneous nodes. An example of the latter method (Segal, 1988) computes the within-node Kaplan-Meier curves for the censored survival data corresponding to each of the two daughter nodes of a possible split and then applies the two-sample log-rank statistic to the Kaplan-Meier curves to measure the goodness of that split; the largest value of the log-rank statistic over all possible splits determines which split is best.

Data that fall into a particular terminal node tend to have similar experiences of survival (based upon a measure of within-node homogeneity). Survival trees can be used to partition patients into groups having similar survival results and, hence, identify common characteristics within these

groups. At each terminal node of a survival tree, we compute a Kaplan-Meier estimate of the survival curve using the survival information for all patients who are members of that node and then compare the survival curves from different terminal nodes.

9.4.3 MARS

Recursive partitioning used in constructing regression trees has been generalized to a flexible class of nonparametric regression models called *multivariate adaptive regression splines (MARS)* (Friedman, 1991).

In the MARS approach, Y is related to $\mathbf{X}$ via the model $Y = \mu(\mathbf{X}) + \epsilon$, where the error term ϵ has mean zero. The regression function, $\mu(\mathbf{X})$, is taken to be a weighted sum of L basis functions,

$$\mu(\mathbf{X}) = \beta_0 + \sum_{\ell=1}^{L} \beta_\ell B_\ell(\mathbf{X}). \tag{9.51}$$

The ℓth basis function,

$$B_\ell(\mathbf{X}) = \prod_{m=1}^{M_\ell} \phi_{\ell m}(X_{q(\ell,m)}), \tag{9.52}$$

is the product of M_ℓ univariate spline functions $\{\phi_{\ell m}(X)\}$, where M_ℓ is a finite number and $q(\ell, m)$ is an index depending upon the ℓth basis function and the mth spline function. Thus, for each ℓ, $B_\ell(\mathbf{X})$ can consist of a single spline function or a product of two or more spline functions, and no input variable can appear more than once in the product. These spline functions (for ℓ odd) are often taken to be linear of the form,

$$\phi_{\ell m}(X) = (X - t_{\ell m})_+, \quad \phi_{\ell+1,m}(X) = (t_{\ell m} - X)_+, \tag{9.53}$$

where $t_{\ell m}$ is a *knot* of $\phi_{\ell m}(X)$ occurring at one of the observed values of $X_{q(\ell,m)}$, $m = 1, 2, \ldots, M_\ell$, $\ell = 1, 2, \ldots, L$. In (9.53), $(x)_+ = \max(0, x)$. If, at $\mathbf{X} = \mathbf{x}$, $B_\ell(\mathbf{x}) = I_{[\mathbf{x}\in\tau_\ell]}$, and, at $Y = y$, $\beta_\ell = y(\tau_\ell)$, then the regression function (9.51) is equivalent to the regression-tree predictor (9.39). Thus, whereas regression trees fit a constant at each terminal node, MARS fits more complicated piecewise linear basis functions.

Basis function are first introduced into the model (9.51) in a forwards-stepwise manner. The process starts by entering the intercept β_0 (i.e., $B_0(\mathbf{X}) = 1$) into the model, and then at each step adding one pair of terms of the form (9.53) (i.e., choosing an input variable and a knot) by minimizing an error sum of squares criterion,

$$ESS(L) = \sum_{i=1}^{n} (y_i - \mu_L(\mathbf{x}_i))^2, \tag{9.54}$$

where, for a given L, $\mu_L(\mathbf{x}_i)$ is (9.51) evaluated at $\mathbf{X} = \mathbf{x}_i$. Suppose the forwards-stepwise procedure terminates at M terms. This model is then "pruned back" by using a backwards-stepwise procedure to prevent possibly overfitting the data. At each step in the backwards-stepwise procedure, we remove one term from the model. This yields M different nested models. To choose between these M models, MARS uses a version of generalized cross-validation (GCV),

$$GCV(m) = \frac{n^{-1}\sum_{i=1}^{n}(y_i - \widehat{\mu}_m(\mathbf{x}_i))^2}{\left(1 - \frac{C(m)}{n}\right)^2}, \quad m = 1, 2, \ldots, M, \tag{9.55}$$

where $\widehat{\mu}_m(\mathbf{x})$ is the fitted value of $\mu(\mathbf{x})$ based upon m terms, the numerator is the apparent error rate (or resubstitution error rate), and $C(m)$ is a *complexity cost function* that represents the *effective number of parameters* in the model (Craven and Wahba, 1979). The best choice of model has $m^* = \arg\min_m GCV(m)$ terms.

9.4.4 Missing Data

In some classification and regression problems, there may be missing values in the test set. Fortunately, there are a number of ways of dealing with missing data when using tree-based methods.

One obvious way is to drop a future observation with a missing data value (or values) down the tree constructed using only complete-data observations and see how far it goes. If the variable with the missing value is not involved in the construction of the tree, then the observation will drop to its appropriate terminal node, and we can then classify the observation or predict its Y value. If, on the other hand, the observation cannot drop any further than a particular internal node τ (because the next split at τ involves the variable with the missing value), we can either stop the observation at τ (Clark and Pregibon, 1992, Section 9.4.1) or force all the observations with a missing value for that variable to drop down to the same daughter node (Zhang and Singer, 1999, Section 4.8).

A method of *surrogate splits* has been proposed (Breiman et al., 1984, Section 5.3) to deal with missing data. The idea of a surrogate split at a given node τ is that we use a variable that best predicts the desired split as a substitute variable on which to split at node τ. If the best-splitting variable for a future observation at τ has a missing value at that split, we use a surrogate split at τ to force that observation further down the tree, assuming, of course, that the variable defining the surrogate split has complete data.

If the missing data occur for a nominal input variable with L levels, then we could introduce an additional level of "`missing`" or "`NA`" so that the variable now has $L + 1$ levels (Kass, 1980).

9.5 Software Packages

The original CART software is commercially available from Salford Systems. S-PLUS and R commands (such as `rpart`) for classification and regression trees are discussed in Venables and Ripley (2002, Chapter 9). For the `rpart` library manual, which we used for the examples in this chapter, see Therneau and Atkinson (1997). Alternative software packages for carrying out tree-based classification and regression are available; they have been implemented in SAS DATA MINING, SPSS CLASSIFICATION TREES, STATISTICA, and SYSTAT, version 7. These versions differ in several aspects, including the impurity measure (typical default is the entropy function), splitting criterion, and the stopping rule.

The original MARS software is also commercially available from Salford Systems. The `mars` command in the `mda` library (Venables and Ripley, 2002, Section 8.8) in S-PLUS and R is available for fitting MARS models.

Bibliographical Notes

This chapter follows the pioneering development of CART (Classification and Regression Trees) by Breiman, Friedman, Olshen, and Stone (1984). Other treatments of the same material can be found in Clark and Pregibon (1992, Chapter 9), Ripley (1996, Chapter 7), Zhang and Singer (1999), and Hastie, Tibshirani, and Friedman (2001, Section 9.2).

Regression trees were introduced by Morgan and Sonquist (1963) using a computer program they named *Automatic Interaction Detection (AID)*. Versions of AID followed: THAID in 1973 and CHAID in 1980; CHAID is used in several computer packages that carry out tree-based methods. Comments and references on the historical development of tree-based methods are given in Ripley (1996, Section 7.4). An excellent discussion of survival trees is given by Zhang and Singer (1999). For discussions of MARS, see Hastie, Tibshirani, and Friedman (2001, Section 9.4) and Zhang and Singer (1999, Chapter 9).

Exercises

9.1 The development of classification trees in this chapter assumes that misclassifying any observation has a cost independent of the classes in-

volved. In many circumstances, this may be unrealistic. For example, a civilized society usually considers convicting an innocent person to be more egregious than finding a guilty person to be not guilty. Define the *misclassification cost* $c(i|j)$ as the cost of misclassifying an observation from the jth class into the ith class. Assume that $c(i|j)$ is nonnegative for $i \neq j$ and zero when $i = j$. Rewrite Sections 9.2.4, 9.2.5, and 9.2.6, taking into account the costs of misclassification.

9.2 The discussion of the way to choose the best split for a classification tree in Section 9.2 used the entropy function as the impurity measure. Use the Gini index as an impurity measure on the Cleveland heart-disease data and determine the best split for the `age` variable (see Table 9.2); draw the graphs of $i(\tau_l)$ and $i(\tau_R)$ for the `age` variable and the goodness of split (see Figure 9.3). Determine the best split for all the variables in the data set (see Table 9.3).

9.3 The full Pima Indians data (768 subjects) has a large number of missing data. In the data set, missing values are designated by zero values. How could you use those subjects having missing values for one or more variables to enhance the classification results discussed in the text?

9.4 Consider the following two examples. Both examples start out with a root node with 800 subjects of which 400 have a given disease and the other 400 do not. The first example splits the root node as follows: the left node has 300 with the disease and 100 without, and the right node has 100 with the disease and 300 without. The second example splits the root node as follows: the left node has 200 with the disease and 400 without, and the right node has 200 with the disease and 0 without. Compute the resubstitution error rate for both examples and show they are equal. Which example do you view as more useful for the future growth of the tree?

9.5 Construct the appropriate-size classification tree for the `BUPA liver disorders` data (see Section 8.4).

9.6 Construct the appropriate-size classification tree for the `spambase` data (see Section 8.4).

9.7 Construct the appropriate-size classification tree for the `forensic glass` data (see Section 8.7).

9.8 Construct the appropriate-size classification tree for the `vehicle` data (see Section 8.7).

9.9 Construct the appropriate-size classification tree for the `wine` data (see Section 8.7).