CHAPTER 19 Decision Tree Classifier

Let the training dataset $\mathbf{D}$ consist of n points $\mathbf{x}_i$ in a d-dimensional space, with y_i being the corresponding class label. We assume that the dimensions or the attributes X_j are numeric or categorical, and that there are k distinct classes, so that $y_i \in \{c_1, c_2, \dots, c_k\}$. A decision tree classifier is a recursive, partition-based tree model that predicts the class $\hat{y}_i$ for each point $\mathbf{x}_i$. Let $\mathcal{R}$ denote the data space that encompasses the set of input points $\mathbf{D}$. A decision tree uses an axis-parallel hyperplane to split the data space $\mathcal{R}$ into two resulting half-spaces or regions, say $\mathcal{R}_1$ and $\mathcal{R}_2$, which also induces a partition of the input points into $\mathbf{D}_1$ and $\mathbf{D}_2$, respectively. Each of these regions is recursively split via axis-parallel hyperplanes until the points within an induced partition are relatively pure in terms of their class labels, that is, most of the points belong to the same class. The resulting hierarchy of split decisions constitutes the decision tree model, with the leaf nodes labeled with the majority class among points in those regions. To classify a new *test* point we have to recursively evaluate which half-space it belongs to until we reach a leaf node in the decision tree, at which point we predict its class as the label of the leaf.

Example 19.1. Consider the Iris dataset shown in Figure 19.1(a), which plots the attributes `sepal length` (X_1) and `sepal width` (X_2). The classification task is to discriminate between c_1, corresponding to `iris-setosa` (in circles), and c_2, corresponding to the other two types of Irises (in triangles). The input dataset $\mathbf{D}$ has $n = 150$ points that lie in the data space which is given as the rectangle, $\mathcal{R} = range(X_1) \times range(X_2) = [4.3, 7.9] \times [2.0, 4.4]$.

The recursive partitioning of the space $\mathcal{R}$ via axis-parallel hyperplanes is illustrated in Figure 19.1a. In two dimensions a hyperplane is simply a line. The first split corresponds to hyperplane h_0 shown as a black line. The resulting left and right half-spaces are further split via hyperplanes h_2 and h_3, respectively (shown as gray lines). The bottom half-space for h_2 is further split via h_4, and the top half-space for h_3 is split via h_5; these third level hyperplanes, h_4 and h_5, are shown as dashed lines. The set of hyperplanes and the set of six leaf regions, namely $\mathcal{R}_1, \dots, \mathcal{R}_6$, constitute the decision tree model. Note also the induced partitioning of the input points into these six regions.

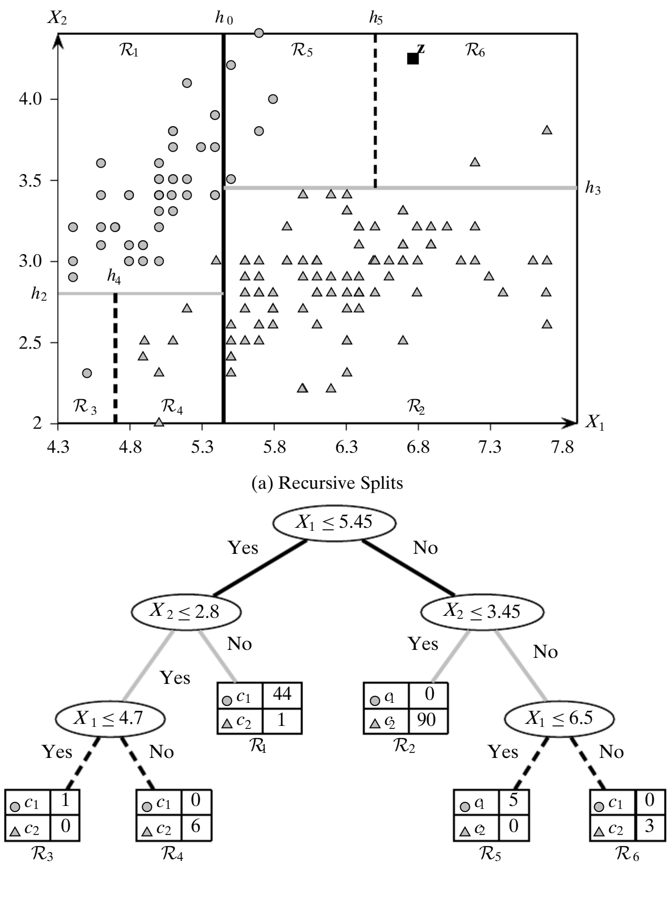

Figure 19.1. Decision trees: recursive partitioning via axis-parallel hyperplanes.

Consider the test point $\mathbf{z} = (6.75, 4.25)^T$ (shown as a white square). To predict its class, the decision tree first checks which side of h_0 it lies in. Because the point lies in the right half-space, the decision tree next checks h_3 to determine that $\mathbf{z}$ is in the top half-space. Finally, we check and find that $\mathbf{z}$ is in the right half-space of h_5, and we reach the leaf region $\mathcal{R}_6$. The predicted class is c_2, as that leaf region has all points (three of them) with class c_2 (triangles).

19.1 DECISION TREES

A decision tree consists of internal nodes that represent the decisions corresponding to the hyperplanes or split points (i.e., which half-space a given point lies in), and leaf nodes that represent regions or partitions of the data space, which are labeled with the majority class. A region is characterized by the subset of data points that lie in that region.

Axis-Parallel Hyperplanes

A hyperplane $h(\mathbf{x})$ is defined as the set of all points $\mathbf{x}$ that satisfy the following equation

$$\boxed{h(\mathbf{x}): \mathbf{w}^T\mathbf{x} + b = 0} \tag{19.1}$$

Here $\mathbf{w} \in \mathbb{R}^d$ is a *weight vector* that is normal to the hyperplane, and b is the offset of the hyperplane from the origin. A decision tree considers only *axis-parallel hyperplanes*, that is, the weight vector must be parallel to one of the original dimensions or axes X_j. Put differently, the weight vector $\mathbf{w}$ is restricted *a priori* to one of the standard basis vectors $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_d\}$, where $\mathbf{e}_i \in \mathbb{R}^d$ has a 1 for the jth dimension, and 0 for all other dimensions. If $\mathbf{x} = (x_1, x_2, \ldots, x_d)^T$ and assuming $\mathbf{w} = \mathbf{e}_j$, we can rewrite Eq. (19.1) as

$$h(\mathbf{x}): \mathbf{e}_j^T\mathbf{x} + b = 0, \text{ which implies that}$$
$$h(\mathbf{x}): x_j + b = 0$$

where the choice of the offset b yields different hyperplanes along dimension X_j.

Split Points

A hyperplane specifies a decision or *split point* because it splits the data space $\mathcal{R}$ into two half-spaces. All points $\mathbf{x}$ such that $h(\mathbf{x}) \leq 0$ are on the hyperplane or to one side of the hyperplane, whereas all points such that $h(\mathbf{x}) > 0$ are on the other side. The split point associated with an axis-parallel hyperplane can be written as $h(\mathbf{x}) \leq 0$, which implies that $x_i + b \leq 0$, or $x_i \leq -b$. Because x_i is some value from dimension X_j and the offset b can be chosen to be any value, the generic form of a split point for a numeric attribute X_j is given as

$$X_j \leq v$$

where $v = -b$ is some value in the domain of attribute X_j. The decision or split point $X_j \leq v$ thus splits the input data space $\mathcal{R}$ into two regions $\mathcal{R}_Y$ and $\mathcal{R}_N$, which denote the set of *all possible points* that satisfy the decision and those that do not.

Data Partition

Each split of $\mathcal{R}$ into $\mathcal{R}_Y$ and $\mathcal{R}_N$ also induces a binary partition of the corresponding input data points $\mathbf{D}$. That is, a split point of the form $X_j \leq v$ induces the data partition

$$\mathbf{D}_Y = \left\{\mathbf{x}^T \mid \mathbf{x} \in \mathbf{D}, x_j \leq v\right\}$$
$$\mathbf{D}_N = \left\{\mathbf{x}^T \mid \mathbf{x} \in \mathbf{D}, x_j > v\right\}$$

where $\mathbf{D}_Y$ is the subset of data points that lie in region $\mathcal{R}_Y$ and $\mathbf{D}_N$ is the subset of input points that line in $\mathcal{R}_N$.

Purity

The purity of a region $\mathcal{R}_j$ is defined in terms of the mixture of classes for points in the corresponding data partition $\mathbf{D}_j$. Formally, purity is the fraction of points with the majority label in $\mathbf{D}_j$, that is,

$$\boxed{purity(\mathbf{D}_j) = \max_i \left\{ \frac{n_{ji}}{n_j} \right\}} \tag{19.2}$$

where $n_j = |\mathbf{D}_j|$ is the total number of data points in the region $\mathcal{R}_j$, and n_{ji} is the number of points in $\mathbf{D}_j$ with class label c_i.

Example 19.2. Figure 19.1(b) shows the resulting decision tree that corresponds to the recursive partitioning of the space via axis-parallel hyperplanes illustrated in Figure 19.1(a). The recursive splitting terminates when appropriate stopping conditions are met, usually taking into account the size and purity of the regions. In this example, we use a size threshold of 5 and a purity threshold of 0.95. That is, a region will be split further only if the number of points is more than five and the purity is less than 0.95.

The very first hyperplane to be considered is $h_1(\mathbf{x}) : x_1 - 5.45 = 0$ which corresponds to the decision

$$X_1 \leq 5.45$$

at the root of the decision tree. The two resulting half-spaces are recursively split into smaller half-spaces.

For example, the region $X_1 \leq 5.45$ is further split using the hyperplane $h_2(\mathbf{x}) : x_2 - 2.8 = 0$ corresponding to the decision

$$X_2 \leq 2.8$$

which forms the left child of the root. Notice how this hyperplane is restricted only to the region $X_1 \leq 5.45$. This is because each region is considered independently after the split, as if it were a separate dataset. There are seven points that satisfy the condition $X_2 \leq 2.8$, out of which one is from class c_1 (circle) and six are from class c_2 (triangles). The purity of this region is therefore $6/7 = 0.857$. Because the region has more than five points, and its purity is less than 0.95, it is further split via the hyperplane $h_4(\mathbf{x}) : x_1 - 4.7 = 0$ yielding the left-most decision node

$$X_1 \leq 4.7$$

in the decision tree shown in Figure 19.1(b).

Returning back to the right half-space corresponding to h_2, namely the region $X_2 > 2.8$, it has 45 points, of which only one is a triangle. The size of the region is 45, but the purity is $44/45 = 0.98$. Because the region exceeds the purity threshold it is not split further. Instead, it becomes a leaf node in the decision tree, and the entire region ($\mathcal{R}_1$) is labeled with the majority class c_1. The frequency for each class is also noted at a leaf node so that the potential error rate for that leaf can be computed. For example, we can expect that the probability of misclassification in region $\mathcal{R}_1$ is $1/45 = 0.022$, which is the error rate for that leaf.

Categorical Attributes

In addition to numeric attributes, a decision tree can also handle categorical data. For a categorical attribute X_j, the split points or decisions are of the $X_j \in V$, where $V \subset dom(X_j)$, and $dom(X_j)$ denotes the domain for X_j. Intuitively, this split can be considered to be the categorical analog of a hyperplane. It results in two "half-spaces," one region $\mathcal{R}_Y$ consisting of points $\mathbf{x}$ that satisfy the condition $x_i \in V$, and the other region $\mathcal{R}_N$ comprising points that satisfy the condition $x_i \notin V$.

Decision Rules

One of the advantages of decision trees is that they produce models that are relatively easy to interpret. In particular, a tree can be read as set of decision rules, with each rule's antecedent comprising the decisions on the internal nodes along a path to a leaf, and its consequent being the label of the leaf node. Further, because the regions are all disjoint and cover the entire space, the set of rules can be interpreted as a set of alternatives or disjunctions.

Example 19.3. Consider the decision tree in Figure 19.1(b). It can be interpreted as the following set of disjunctive rules, one per leaf region $\mathcal{R}_i$

$$\mathcal{R}_3 : \text{If } X_1 \leq 5.45 \text{ and } X_2 \leq 2.8 \text{ and } X_1 \leq 4.7, \text{ then class is } c_1, \text{ or}$$
$$\mathcal{R}_4 : \text{If } X_1 \leq 5.45 \text{ and } X_2 \leq 2.8 \text{ and } X_1 > 4.7, \text{ then class is } c_2, \text{ or}$$
$$\mathcal{R}_1 : \text{If } X_1 \leq 5.45 \text{ and } X_2 > 2.8, \text{ then class is } c_1, \text{ or}$$
$$\mathcal{R}_2 : \text{If } X_1 > 5.45 \text{ and } X_2 \leq 3.45, \text{ then class is } c_2, \text{ or}$$
$$\mathcal{R}_5 : \text{If } X_1 > 5.45 \text{ and } X_2 > 3.45 \text{ and } X_1 \leq 6.5, \text{ then class is } c_1, \text{ or}$$
$$\mathcal{R}_6 : \text{If } X_1 > 5.45 \text{ and } X_2 > 3.45 \text{ and } X_1 > 6.5, \text{ then class is } c_2$$

19.2 DECISION TREE ALGORITHM

The pseudo-code for decision tree model construction is shown in Algorithm 19.1. It takes as input a training dataset $\mathbf{D}$, and two parameters η and π, where η is the leaf size and π the leaf purity threshold. Different split points are evaluated for each attribute in $\mathbf{D}$. Numeric decisions are of the form $X_j \leq v$ for some value v in the value range for attribute X_j, and categorical decisions are of the form $X_j \in V$ for some subset of values in the domain of X_j. The best split point is chosen to partition the data into two subsets, $\mathbf{D}_Y$ and $\mathbf{D}_N$, where $\mathbf{D}_Y$ corresponds to all points $\mathbf{x} \in \mathbf{D}$ that satisfy the

Algorithm 19.1: Decision Tree Algorithm

```
DecisionTree ($\mathbf{D}, \eta, \pi$):
1  $n \leftarrow |\mathbf{D}|$ // partition size
2  $n_i \leftarrow |\{\mathbf{x}_j | \mathbf{x}_j \in \mathbf{D}, y_j = c_i\}|$ // size of class $c_i$
3  $purity(\mathbf{D}) \leftarrow \max_i \{\frac{n_i}{n}\}$
4  if $n \leq \eta$ or $purity(\mathbf{D}) \geq \pi$ then // stopping condition
5      $c^* \leftarrow \arg\max_{c_i} \{\frac{n_i}{n}\}$ // majority class
6      create leaf node, and label it with class $c^*$
7      return
8  $(split\ point^*, score^*) \leftarrow (\emptyset, 0)$ // initialize best split point
9  foreach (attribute $X_j$) do
10     if ($X_j$ is numeric) then
11         $(v, score) \leftarrow$ Evaluate-Numeric-Attribute($\mathbf{D}, X_j$)
12         if $score > score^*$ then $(split\ point^*, score^*) \leftarrow (X_j \leq v, score)$
13     else if ($X_j$ is categorical) then
14         $(V, score) \leftarrow$ Evaluate-Categorical-Attribute($\mathbf{D}, X_j$)
15         if $score > score^*$ then $(split\ point^*, score^*) \leftarrow (X_j \in V, score)$
   // partition $\mathbf{D}$ into $\mathbf{D}_Y$ and $\mathbf{D}_N$ using $split\ point^*$, and call
      recursively
16 $\mathbf{D}_Y \leftarrow \{\mathbf{x}^T \mid \mathbf{x} \in \mathbf{D}$ satisfies $split\ point^*\}$
17 $\mathbf{D}_N \leftarrow \{\mathbf{x}^T \mid \mathbf{x} \in \mathbf{D}$ does not satisfy $split\ point^*\}$
18 create internal node $split\ point^*$, with two child nodes, $\mathbf{D}_Y$ and $\mathbf{D}_N$
19 DecisionTree($\mathbf{D}_Y, \eta, \pi$); DecisionTree($\mathbf{D}_N, \eta, \pi$)
```

split decision, and $\mathbf{D}_N$ corresponds to all points that do not satisfy the split decision. The decision tree method is then called recursively on $\mathbf{D}_Y$ and $\mathbf{D}_N$. A number of stopping conditions can be used to stop the recursive partitioning process. The simplest condition is based on the size of the partition $\mathbf{D}$. If the number of points n in $\mathbf{D}$ drops below the user-specified size threshold η, then we stop the partitioning process and make $\mathbf{D}$ a leaf. This condition prevents over-fitting the model to the training set, by avoiding to model very small subsets of the data. Size alone is not sufficient because if the partition is already pure then it does not make sense to split it further. Thus, the recursive partitioning is also terminated if the purity of $\mathbf{D}$ is above the purity threshold π. Details of how the split points are evaluated and chosen are given next.

19.2.1 Split Point Evaluation Measures

Given a split point of the form $X_j \leq v$ or $X_j \in V$ for a numeric or categorical attribute, respectively, we need an objective criterion for scoring the split point. Intuitively, we want to select a split point that gives the best separation or discrimination between the different class labels.

Entropy

Entropy, in general, measures the amount of disorder or uncertainty in a system. In the classification setting, a partition has lower entropy (or low disorder) if it is relatively pure, that is, if most of the points have the same label. On the other hand, a partition has higher entropy (or more disorder) if the class labels are mixed, and there is no majority class as such.

The entropy of a set of labeled points $\mathbf{D}$ is defined as follows:

$$\boxed{H(\mathbf{D}) = -\sum_{i=1}^{k} P(c_i|\mathbf{D}) \log_2 P(c_i|\mathbf{D})} \tag{19.3}$$

where $P(c_i|\mathbf{D})$ is the probability of class c_i in $\mathbf{D}$, and k is the number of classes. If a region is pure, that is, has points from the same class, then the entropy is zero. On the other hand, if the classes are all mixed up, and each appears with equal probability $P(c_i|\mathbf{D}) = \frac{1}{k}$, then the entropy has the highest value, $H(\mathbf{D}) = \log_2 k$.

Assume that a split point partitions $\mathbf{D}$ into $\mathbf{D}_Y$ and $\mathbf{D}_N$. Define the *split entropy* as the weighted entropy of each of the resulting partitions, given as

$$\boxed{H(\mathbf{D}_Y, \mathbf{D}_N) = \frac{n_Y}{n} H(\mathbf{D}_Y) + \frac{n_N}{n} H(\mathbf{D}_N)} \tag{19.4}$$

where $n = |\mathbf{D}|$ is the number of points in $\mathbf{D}$, and $n_Y = |\mathbf{D}_Y|$ and $n_N = |\mathbf{D}_N|$ are the number of points in $\mathbf{D}_Y$ and $\mathbf{D}_N$.

To see if the split point results in a reduced overall entropy, we define the *information gain* for a given split point as follows:

$$\boxed{Gain(\mathbf{D}, \mathbf{D}_Y, \mathbf{D}_N) = H(\mathbf{D}) - H(\mathbf{D}_Y, \mathbf{D}_N)} \tag{19.5}$$

The higher the information gain, the more the reduction in entropy, and the better the split point. Thus, given split points and their corresponding partitions, we can score each split point and choose the one that gives the highest information gain.

Gini Index

Another common measure to gauge the purity of a split point is the *Gini index*, defined as follows:

$$\boxed{G(\mathbf{D}) = 1 - \sum_{i=1}^{k} P(c_i|\mathbf{D})^2} \tag{19.6}$$

If the partition is pure, then the probability of the majority class is 1 and the probability of all other classes is 0, and thus, the Gini index is 0. On the other hand, when each class is equally represented, with probability $P(c_i|\mathbf{D}) = \frac{1}{k}$, then the Gini index has value $\frac{k-1}{k}$. Thus, higher values of the Gini index indicate more disorder, and lower values indicate more order in terms of the class labels.

We can compute the weighted Gini index of a split point as follows:

$$G(\mathbf{D}_Y, \mathbf{D}_N) = \frac{n_Y}{n} G(\mathbf{D}_Y) + \frac{n_N}{n} G(\mathbf{D}_N)$$

where n, n_Y, and n_N denote the number of points in regions $\mathbf{D}$, $\mathbf{D}_Y$, and $\mathbf{D}_N$, respectively. The lower the Gini index value, the better the split point.

Other measures can also be used instead of entropy and Gini index to evaluate the splits. For example, the Classification And Regression Trees (CART) measure is given as

$$\boxed{CART(\mathbf{D}_Y, \mathbf{D}_N) = 2\frac{n_Y}{n}\frac{n_N}{n}\sum_{i=1}^{k}\Big|P(c_i|\mathbf{D}_Y) - P(c_i|\mathbf{D}_N)\Big|} \tag{19.7}$$

This measure thus prefers a split point that maximizes the difference between the class probability mass function for the two partitions; the higher the CART measure, the better the split point.

19.2.2 Evaluating Split Points

All of the split point evaluation measures, such as entropy [Eq. (19.3)], Gini-index [Eq. (19.6)], and CART [Eq. (19.7)], considered in the preceding section depend on the class probability mass function (PMF) for $\mathbf{D}$, namely, $P(c_i|\mathbf{D})$, and the class PMFs for the resulting partitions $\mathbf{D}_Y$ and $\mathbf{D}_N$, namely $P(c_i|\mathbf{D}_Y)$ and $P(c_i|\mathbf{D}_N)$. Note that we have to compute the class PMFs for all possible split points; scoring each of them independently would result in significant computational overhead. Instead, one can incrementally compute the PMFs as described in the following paragraphs.

Numeric Attributes

If X is a numeric attribute, we have to evaluate split points of the form $X \le v$. Even if we restrict v to lie within the value range of attribute X, there are still an infinite number of choices for v. One reasonable approach is to consider only the midpoints between two successive distinct values for X in the sample $\mathbf{D}$. This is because split points of the form $X \le v$, for $v \in [x_a, x_b)$, where x_a and x_b are two successive distinct values of X in $\mathbf{D}$, produce the same partitioning of $\mathbf{D}$ into $\mathbf{D}_Y$ and $\mathbf{D}_N$, and thus yield the same scores. Because there can be at most n distinct values for X, there are at most $n-1$ midpoint values to consider.

Let $\{v_1, \ldots, v_m\}$ denote the set of all such midpoints, such that $v_1 < v_2 < \cdots < v_m$. For each split point $X \le v$, we have to estimate the class PMFs:

$$\hat{P}(c_i|\mathbf{D}_Y) = \hat{P}(c_i|X \le v) \tag{19.8}$$

$$\hat{P}(c_i|\mathbf{D}_N) = \hat{P}(c_i|X > v) \tag{19.9}$$

Let $I()$ be an indicator variable that takes on the value 1 only when its argument is true, and is 0 otherwise. Using the Bayes theorem, we have

$$\hat{P}(c_i|X \le v) = \frac{\hat{P}(X \le v|c_i)\hat{P}(c_i)}{\hat{P}(X \le v)} = \frac{\hat{P}(X \le v|c_i)\hat{P}(c_i)}{\sum_{j=1}^{k}\hat{P}(X \le v|c_j)\hat{P}(c_j)} \tag{19.10}$$

The prior probability for each class in $\mathbf{D}$ can be estimated as follows:

$$\hat{P}(c_i) = \frac{1}{n}\sum_{j=1}^{n} I(y_j = c_i) = \frac{n_i}{n} \tag{19.11}$$

where y_j is the class for point $\mathbf{x}_j$, $n = |\mathbf{D}|$ is the total number of points, and n_i is the number of points in $\mathbf{D}$ with class c_i. Define N_{vi} as the number of points $x_j \leq v$ with class c_i, where x_j is the value of data point $\mathbf{x}_j$ for the attribute X, given as

$$N_{vi} = \sum_{j=1}^{n} I(x_j \leq v \text{ and } y_j = c_i) \tag{19.12}$$

We can then estimate $P(X \leq v|c_i)$ as follows:

$$\begin{aligned}\hat{P}(X \leq v|c_i) &= \frac{\hat{P}(X \leq v \text{ and } c_i)}{\hat{P}(c_i)} = \left(\frac{1}{n}\sum_{j=1}^{n} I(x_j \leq v \text{ and } y_j = c_i)\right) \Big/ (n_i/n) \\ &= \frac{N_{vi}}{n_i}\end{aligned} \tag{19.13}$$

Plugging Eqs. (19.11) and (19.13) into Eq. (19.10), and using Eq. (19.8), we have

$$\hat{P}(c_i|\mathbf{D}_Y) = \hat{P}(c_i|X \leq v) = \frac{N_{vi}}{\sum_{j=1}^{k} N_{vj}} \tag{19.14}$$

We can estimate $\hat{P}(X > v|c_i)$ as follows:

$$\hat{P}(X > v|c_i) = 1 - \hat{P}(X \leq v|c_i) = 1 - \frac{N_{vi}}{n_i} = \frac{n_i - N_{vi}}{n_i} \tag{19.15}$$

Using Eqs. (19.11) and (19.15), the class PMF $\hat{P}(c_i|\mathbf{D}_N)$ is given as

$$\boxed{\hat{P}(c_i|\mathbf{D}_N) = \hat{P}(c_i|X > v) = \frac{\hat{P}(X > v|c_i)\hat{P}(c_i)}{\sum_{j=1}^{k} \hat{P}(X > v|c_j)\hat{P}(c_j)} = \frac{n_i - N_{vi}}{\sum_{j=1}^{k}(n_j - N_{vj})}} \tag{19.16}$$

Algorithm 19.2 shows the split point evaluation method for numeric attributes. The for loop on line 4 iterates through all the points and computes the midpoint values v and the number of points N_{vi} from class c_i such that $x_j \leq v$. The for loop on line 12 enumerates all possible split points of the form $X \leq v$, one for each midpoint v, and scores them using the gain criterion [Eq. (19.5)]; the best split point and score are recorded and returned. Any of the other evaluation measures can also be used. However, for Gini index and CART a lower score is better unlike for gain where a higher score is better.

In terms of computational complexity, the initial sorting of values of X (line 1) takes time $O(n \log n)$. The cost of computing the midpoints and the class-specific counts N_{vi} takes time $O(nk)$ (for loop on line 4). The cost of computing the score is also bounded by $O(nk)$, because the total number of midpoints v can be at most n (for loop on line 12). The total cost of evaluating a numeric attribute is therefore $O(n \log n + nk)$. Ignoring k, because it is usually a small constant, the total cost of numeric split point evaluation is $O(n \log n)$.

Example 19.4 (Numeric Attributes). Consider the 2-dimensional Iris dataset shown in Figure 19.1(a). In the initial invocation of Algorithm 19.1, the entire dataset $\mathbf{D}$ with

Algorithm 19.2: Evaluate Numeric Attribute (Using Gain)

```
EVALUATE-NUMERIC-ATTRIBUTE (D, X):
1  sort D on attribute $X$, so that $x_j \le x_{j+1}, \forall j = 1, \dots, n-1$
2  $\mathcal{M} \leftarrow \emptyset$ // set of midpoints
3  for $i = 1, \dots, k$ do $n_i \leftarrow 0$
4  for $j = 1, \dots, n-1$ do
5      if $y_j = c_i$ then $n_i \leftarrow n_i + 1$
       // running count for class $c_i$
6      if $x_{j+1} \ne x_j$ then
7          $v \leftarrow \frac{x_{j+1} + x_j}{2}$; $\mathcal{M} \leftarrow \mathcal{M} \cup \{v\}$ // midpoints
8          for $i = 1, \dots, k$ do
9              $N_{vi} \leftarrow n_i$ // Number of points such that $x_j \le v$ and
                   $y_j = c_i$
10 if $y_n = c_i$ then $n_i \leftarrow n_i + 1$
   // evaluate split points of the form $X \le v$
11 $v^* \leftarrow \emptyset$; $score^* \leftarrow 0$ // initialize best split point
12 forall $v \in \mathcal{M}$ do
13     for $i = 1, \dots, k$ do
14         $\hat{P}(c_i|\mathbf{D}_Y) \leftarrow \frac{N_{vi}}{\sum_{j=1}^{k} N_{vj}}$
15         $\hat{P}(c_i|\mathbf{D}_N) \leftarrow \frac{n_i - N_{vi}}{\sum_{j=1}^{k} n_j - N_{vj}}$
16     $score(X \le v) \leftarrow Gain(\mathbf{D}, \mathbf{D}_Y, \mathbf{D}_N)$ // use Eq. (19.5)
17     if $score(X \le v) > score^*$ then
18         $v^* \leftarrow v$; $score^* \leftarrow score(X \le v)$
19 return $(v^*, score^*)$
```

$n = 150$ points is considered at the root of the decision tree. The task is to find the best split point considering both the attributes, X_1 (`sepal length`) and X_2 (`sepal width`). Because there are $n_1 = 50$ points labeled c_1 (`iris-setosa`), the other class c_2 has $n_2 = 100$ points. We thus have

$$\hat{P}(c_1) = 50/150 = 1/3$$
$$\hat{P}(c_2) = 100/150 = 2/3$$

The entropy [Eq. (19.3)] of the dataset **D** is therefore

$$H(\mathbf{D}) = -\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) = 0.918$$

Consider split points for attribute X_1. To evaluate the splits we first compute the frequencies N_{vi} using Eq. (19.12), which are plotted in Figure 19.2 for both the classes. For example, consider the split point $X_1 \le 5.45$. From Figure 19.2, we see that

$$N_{v1} = 45 \qquad N_{v2} = 7$$

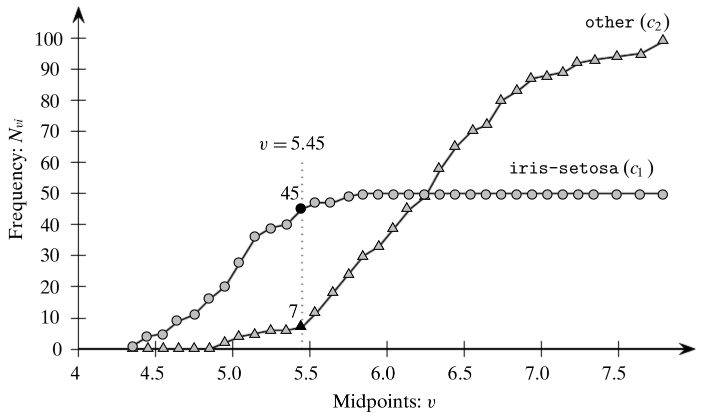

Figure 19.2. Iris: frequencies N_{vi} for classes c_1 and c_2 for attribute `sepal length`.

Plugging in these values into Eq. (19.14) we get

$$\hat{P}(c_1|\mathbf{D}_Y) = \frac{N_{v1}}{N_{v1}+N_{v2}} = \frac{45}{45+7} = 0.865$$

$$\hat{P}(c_2|\mathbf{D}_Y) = \frac{N_{v2}}{N_{v1}+N_{v2}} = \frac{7}{45+7} = 0.135$$

and using Eq. (19.16), we obtain

$$\hat{P}(c_1|\mathbf{D}_N) = \frac{n_1 - N_{v1}}{(n_1 - N_{v1}) + (n_2 - N_{v2})} = \frac{50-45}{(50-45)+(100-7)} = 0.051$$

$$\hat{P}(c_2|\mathbf{D}_N) = \frac{n_2 - N_{v2}}{(n_1 - N_{v1}) + (n_2 - N_{v2})} = \frac{(100-7)}{(50-45)+(100-7)} = 0.949$$

We can now compute the entropy of the partitions $\mathbf{D}_Y$ and $\mathbf{D}_N$ as follows:

$$H(\mathbf{D}_Y) = -(0.865\log_2 0.865 + 0.135\log_2 0.135) = 0.571$$

$$H(\mathbf{D}_N) = -(0.051\log_2 0.051 + 0.949\log_2 0.949) = 0.291$$

The entropy of the split point $X \le 5.45$ is given via Eq. (19.4)

$$H(\mathbf{D}_Y, \mathbf{D}_N) = \frac{52}{150}H(\mathbf{D}_Y) + \frac{98}{150}H(\mathbf{D}_N) = 0.388$$

where $n_Y = |\mathbf{D}_Y| = 52$ and $n_N = |\mathbf{D}_N| = 98$. The information gain for the split point is therefore

$$Gain = H(\mathbf{D}) - H(\mathbf{D}_Y, \mathbf{D}_N) = 0.918 - 0.388 = 0.53$$

In a similar manner, we can evaluate all of the split points for both attributes X_1 and X_2. Figure 19.3 plots the gain values for the different split points for the two

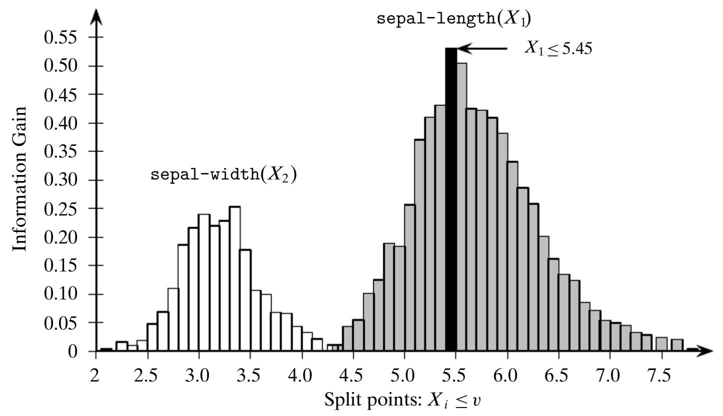

Figure 19.3. Iris: gain for different split points, for `sepal length` and `sepal width`.

attributes. We can observe that $X \le 5.45$ is the best split point and it is thus chosen as the root of the decision tree in Figure 19.1(b).

The recursive tree growth process continues and yields the final decision tree and the split points as shown in Figure 19.1(b). In this example, we use a leaf size threshold of 5 and a purity threshold of 0.95.

Categorical Attributes

If X is a categorical attribute we evaluate split points of the form $X \in V$, where $V \subset dom(X)$ and $V \neq \emptyset$. In words, all distinct partitions of the set of values of X are considered. Because the split point $X \in V$ yields the same partition as $X \in \overline{V}$, where $\overline{V} = dom(X) \setminus V$ is the complement of V, the total number of distinct partitions is given as

$$\sum_{i=1}^{\lfloor m/2 \rfloor} \binom{m}{i} = O(2^{m-1}) \tag{19.17}$$

where m is the number of values in the domain of X, that is, $m = |dom(X)|$. The number of possible split points to consider is therefore exponential in m, which can pose problems if m is large. One simplification is to restrict V to be of size one, so that there are only m split points of the form $X_j \in \{v\}$, where $v \in dom(X_j)$.

To evaluate a given split point $X \in V$ we have to compute the following class probability mass functions:

$$P(c_i|\mathbf{D}_Y) = P(c_i|X \in V) \qquad\qquad P(c_i|\mathbf{D}_N) = P(c_i|X \notin V)$$

Making use of the Bayes theorem, we have

$$P(c_i|X \in V) = \frac{P(X \in V|c_i)P(c_i)}{P(X \in V)} = \frac{P(X \in V|c_i)P(c_i)}{\sum_{j=1}^{k} P(X \in V|c_j)P(c_j)}$$

However, note that a given point $\mathbf{x}$ can take on only one value in the domain of X, and thus the values $v \in dom(X)$ are mutually exclusive. Therefore, we have

$$P(X \in V|c_i) = \sum_{v \in V} P(X = v|c_i)$$

and we can rewrite $P(c_i|\mathbf{D}_Y)$ as

$$P(c_i|\mathbf{D}_Y) = \frac{\sum_{v \in V} P(X = v|c_i)P(c_i)}{\sum_{j=1}^{k}\sum_{v \in V} P(X = v|c_j)P(c_j)} \tag{19.18}$$

Define n_{vi} as the number of points $\mathbf{x}_j \in \mathbf{D}$, with value $x_j = v$ for attribute X and having class $y_j = c_i$:

$$n_{vi} = \sum_{j=1}^{n} I(x_j = v \text{ and } y_j = c_i) \tag{19.19}$$

The class conditional empirical PMF for X is then given as

$$\begin{aligned}\hat{P}(X = v|c_i) &= \frac{\hat{P}\big(X = v \text{ and } c_i\big)}{\hat{P}(c_i)} \\ &= \left(\frac{1}{n}\sum_{j=1}^{n} I(x_j = v \text{ and } y_j = c_i)\right) \Big/ \big(n_i/n\big) \\ &= \frac{n_{vi}}{n_i} \end{aligned} \tag{19.20}$$

Note that the class prior probabilities can be estimated using Eq. (19.11) as discussed earlier, that is, $\hat{P}(c_i) = n_i/n$. Thus, substituting Eq. (19.20) in Eq. (19.18), the class PMF for the partition $\mathbf{D}_Y$ for the split point $X \in V$ is given as

$$\boxed{\hat{P}(c_i|\mathbf{D}_Y) = \frac{\sum_{v \in V} \hat{P}(X = v|c_i)\hat{P}(c_i)}{\sum_{j=1}^{k}\sum_{v \in V} \hat{P}(X = v|c_j)\hat{P}(c_j)} = \frac{\sum_{v \in V} n_{vi}}{\sum_{j=1}^{k}\sum_{v \in V} n_{vj}}} \tag{19.21}$$

In a similar manner, the class PMF for the partition $\mathbf{D}_N$ is given as

$$\boxed{\hat{P}(c_i|\mathbf{D}_N) = \hat{P}(c_i|X \notin V) = \frac{\sum_{v \notin V} n_{vi}}{\sum_{j=1}^{k}\sum_{v \notin V} n_{vj}}} \tag{19.22}$$

Algorithm 19.3 shows the split point evaluation method for categorical attributes. The for loop on line 4 iterates through all the points and computes n_{vi}, that is, the number of points having value $v \in dom(X)$ and class c_i. The for loop on line 7 enumerates all possible split points of the form $X \in V$ for $V \subset dom(X)$, such that $|V| \leq l$, where l is a user specified parameter denoting the maximum cardinality of V. For

Algorithm 19.3: Evaluate Categorical Attribute (Using Gain)

```
EVALUATE-CATEGORICAL-ATTRIBUTE ($\mathbf{D}, X, l$):
1  for $i = 1, \dots, k$ do
2      $n_i \leftarrow 0$
3      forall $v \in dom(X)$ do $n_{vi} \leftarrow 0$
4  for $j = 1, \dots, n$ do
5      if $x_j = v$ and $y_j = c_i$ then $n_{vi} \leftarrow n_{vi} + 1$ // frequency statistics
   // evaluate split points of the form $X \in V$
6  $V^* \leftarrow \emptyset$; $score^* \leftarrow 0$ // initialize best split point
7  forall $V \subset dom(X)$, such that $1 \le |V| \le l$ do
8      for $i = 1, \dots, k$ do
9          $\hat{P}(c_i|\mathbf{D}_Y) \leftarrow \frac{\sum_{v \in V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \in V} n_{vj}}$
10         $\hat{P}(c_i|\mathbf{D}_N) \leftarrow \frac{\sum_{v \notin V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \notin V} n_{vj}}$
11     $score(X \in V) \leftarrow Gain(\mathbf{D}, \mathbf{D}_Y, \mathbf{D}_N)$ // use Eq. (19.5)
12     if $score(X \in V) > score^*$ then
13         $V^* \leftarrow V$; $score^* \leftarrow score(X \in V)$
14 return $(V^*, score^*)$
```

example, to control the number of split points, we can also restrict V to be a single item, that is, $l = 1$, so that splits are of the form $V \in \{v\}$, with $v \in dom(X)$. If $l = \lfloor m/2 \rfloor$, we have to consider all possible distinct partitions V. Given a split point $X \in V$, the method scores it using information gain [Eq. (19.5)], although any of the other scoring criteria can also be used. The best split point and score are recorded and returned.

In terms of computational complexity the class-specific counts for each value n_{vi} takes $O(n)$ time (for loop on line 4). With $m = |dom(X)|$, the maximum number of partitions V is $O(2^{m-1})$, and because each split point can be evaluated in time $O(mk)$, the for loop in line 7 takes time $O(mk2^{m-1})$. The total cost for categorical attributes is therefore $O(n + mk2^{m-1})$. If we make the assumption that $2^{m-1} = O(n)$, that is, if we bound the maximum size of V to $l = O(\log n)$, then the cost of categorical splits is bounded as $O(n \log n)$, ignoring k.

Example 19.5 (Categorical Attributes). Consider the 2-dimensional Iris dataset comprising the `sepal length` and `sepal width` attributes. Let us assume that `sepal length` has been discretized as shown in Table 19.1. The class frequencies n_{vi} are also shown. For instance $n_{a_1 2} = 6$ denotes the fact that there are 6 points in $\mathbf{D}$ with value $v = a_1$ and class c_2.

Consider the split point $X_1 \in \{a_1, a_3\}$. From Table 19.1 we can compute the class PMF for partition $\mathbf{D}_Y$ using Eq. (19.21)

$$\hat{P}(c_1|\mathbf{D}_Y) = \frac{n_{a_1 1} + n_{a_3 1}}{(n_{a_1 1} + n_{a_3 1}) + (n_{a_1 2} + n_{a_3 2})} = \frac{39 + 0}{(39 + 0) + (6 + 43)} = 0.443$$

Table 19.1. Discretized `sepal length` attribute: class frequencies

Bins	v: values	Class frequencies (n_{vi})	
		c_1:`iris-setosa`	c_2:`other`
$[4.3, 5.2]$	Very Short (a_1)	39	6
$(5.2, 6.1]$	Short (a_2)	11	39
$(6.1, 7.0]$	Long (a_3)	0	43
$(7.0, 7.9]$	Very Long (a_4)	0	12

$$\hat{P}(c_2|\mathbf{D}_Y) = 1 - \hat{P}(c_1|\mathbf{D}_Y) = 0.557$$

with the entropy given as

$$H(\mathbf{D}_Y) = -(0.443 \log_2 0.443 + 0.557 \log_2 0.557) = 0.991$$

To compute the class PMF for $\mathbf{D}_N$ [Eq. (19.22)], we sum up the frequencies over values $v \notin V = \{a_1, a_3\}$, that is, we sum over $v = a_2$ and $v = a_4$, as follows:

$$\hat{P}(c_1|\mathbf{D}_N) = \frac{n_{a_2 1} + n_{a_4 1}}{(n_{a_2 1} + n_{a_4 1}) + (n_{a_2 2} + n_{a_4 2})} = \frac{11 + 0}{(11 + 0) + (39 + 12)} = 0.177$$

$$\hat{P}(c_2|\mathbf{D}_N) = 1 - \hat{P}(c_1|\mathbf{D}_N) = 0.823$$

with the entropy given as

$$H(\mathbf{D}_N) = -(0.177 \log_2 0.177 + 0.823 \log_2 0.823) = 0.673$$

We can see from Table 19.1 that $V \in \{a_1, a_3\}$ splits the input data $\mathbf{D}$ into partitions of size $|\mathbf{D}_Y| = 39 + 6 + 43 = 88$, and $|\mathbf{D}_N| = 150 - 88 = 62$. The entropy of the split is therefore given as

$$H(\mathbf{D}_Y, \mathbf{D}_N) = \frac{88}{150} H(\mathbf{D}_Y) + \frac{62}{150} H(\mathbf{D}_N) = 0.86$$

As noted in Example 19.4, the entropy of the whole dataset $\mathbf{D}$ is $H(\mathbf{D}) = 0.918$. The gain is then given as

$$Gain = H(\mathbf{D}) - H(\mathbf{D}_Y, \mathbf{D}_N) = 0.918 - 0.86 = 0.058$$

The split entropy and gain values for all the categorical split points are given in Table 19.2. We can see that $X_1 \in \{a_1\}$ is the best split point on the discretized attribute X_1.

19.2.3 Computational Complexity

To analyze the computational complexity of the decision tree method in Algorithm 19.1, we assume that the cost of evaluating all the split points for a numeric or categorical attribute is $O(n \log n)$, where $n = |\mathbf{D}|$ is the size of the dataset. Given $\mathbf{D}$, the decision

Table 19.2. Categorical split points for `sepal length`

V	Split entropy	Info. gain
$\{a_1\}$	0.509	0.410
$\{a_2\}$	0.897	0.217
$\{a_3\}$	0.711	0.207
$\{a_4\}$	0.869	0.049
$\{a_1, a_2\}$	0.632	0.286
$\{a_1, a_3\}$	0.860	0.058
$\{a_1, a_4\}$	0.667	0.251
$\{a_2, a_3\}$	0.667	0.251
$\{a_2, a_4\}$	0.860	0.058
$\{a_3, a_4\}$	0.632	0.286

tree algorithm evaluates all d attributes, with cost $(dn \log n)$. The total cost depends on the depth of the decision tree. In the worst case, the tree can have depth n, and thus the total cost is $O(dn^2 \log n)$.

19.3 FURTHER READING

Among the earliest works on decision trees are Hunt, Marin, and Stone (1966); Breiman et al. (1984); and Quinlan (1986). The description in this chapter is largely based on the C4.5 method described in Quinlan (1993), which is an excellent reference for further details, such as how to prune decision trees to prevent overfitting, how to handle missing attribute values, and other implementation issues. A survey of methods for simplifying decision trees appears in Breslow and Aha (1997). Scalable implementation techniques are described in Mehta, Agrawal, and Rissanen (1996) and Gehrke et al. (1999).

Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984). *Classification and Regression Trees*. Boca Raton, FL: Chapman and Hall/CRC Press.

Breslow, L. A. and Aha, D. W. (1997). Simplifying decision trees: A survey. *Knowledge Engineering Review*, 12 (1), 1–40.

Gehrke, J., Ganti, V., Ramakrishnan, R., and Loh, W.-Y. (1999). BOAT–Optimistic decision tree construction. *ACM SIGMOD Record*, 28 (2), 169–180.

Hunt, E. B., Marin, J., and Stone, P. J. (1966). *Experiments in Induction*. New York: Academic Press.

Mehta, M., Agrawal, R., and Rissanen, J. (1996). SLIQ: A fast scalable classifier for data mining. *Proceedings of the International Conference on Extending Database Technology*. New York: Springer-Verlag, pp. 18–32.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1 (1), 81–106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. New York: Morgan Kaufmann.

19.4 EXERCISES

Q1. True or False:

(a) High entropy means that the partitions in classification are "pure."

(b) Multiway split of a categorical attribute generally results in more pure partitions than a binary split.

Q2. Given Table 19.3, construct a decision tree using a purity threshold of 100%. Use information gain as the split point evaluation measure. Next, classify the point (Age=27,Car=Vintage).

Table 19.3. Data for Q2: `Age` is numeric and `Car` is categorical. `Risk` gives the class label for each point: high (H) or low (L)

	Age	Car	**Risk**
$\mathbf{x}_1^T$	25	Sports	L
$\mathbf{x}_2^T$	20	Vintage	H
$\mathbf{x}_3^T$	25	Sports	L
$\mathbf{x}_4^T$	45	SUV	H
$\mathbf{x}_5^T$	20	Sports	H
$\mathbf{x}_6^T$	25	SUV	H

Table 19.4. Data for Q4

Instance	a_1	a_2	a_3	Class
1	T	T	5.0	Y
2	T	T	7.0	Y
3	T	F	8.0	N
4	F	F	3.0	Y
5	F	T	7.0	N
6	F	T	4.0	N
7	F	F	5.0	N
8	T	F	6.0	Y
9	F	T	1.0	N

Q3. What is the maximum and minimum value of the CART measure [Eq. (19.7)] and under what conditions?

Q4. Given the dataset in Table 19.4. Answer the following questions:

(a) Show which decision will be chosen at the root of the decision tree using information gain [Eq. (19.5)], Gini index [Eq. (19.6)], and CART [Eq. (19.7)] measures. Show all split points for all attributes.

(b) What happens to the purity if we use Instance as another attribute? Do you think this attribute should be used for a decision in the tree?

Q5. Consider Table 19.5. Let us make a nonlinear split instead of an axis parallel split, given as follows: $AB - B^2 \leq 0$. Compute the information gain of this split based on entropy (use $\log_2$, i.e., log to the base 2).

Table 19.5. Data for Q5

	A	B	Class
$\mathbf{x}_1^T$	3.5	4	H
$\mathbf{x}_2^T$	2	4	H
$\mathbf{x}_3^T$	9.1	4.5	L
$\mathbf{x}_4^T$	2	6	H
$\mathbf{x}_5^T$	1.5	7	H
$\mathbf{x}_6^T$	7	6.5	H
$\mathbf{x}_7^T$	2.1	2.5	L
$\mathbf{x}_8^T$	8	4	L