Chapter 19

Decision Tree Classifier

Let the training dataset $\mathbf{D}$ consist of n points $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ in a d-dimensional space with categorical or numeric attributes X_j (with $j = 1, \ldots, d$). Further, let Y denote the (categorical) class attribute, with y_i denoting the class label of point $\mathbf{x}_i$. We assume that there are k distinct classes, so that $y_i \in \{c_1, c_2, \cdots, c_k\}$. The goal of decision tree classification is to build a *partition-based* model that predicts the class $\hat{y}_i$ for each point $\mathbf{x}_i$.

A decision tree uses axis-parallel hyperplanes to recursively partition the data space, so that the resulting set of partitions are as "pure" as possible in terms of the class labels. A decision tree consists of internal nodes that represent the decisions or split points, and leaf nodes that represent regions of the data space labeled with the majority class. A region $\mathcal{D} \subseteq \mathbf{D}$ is characterized by the subset of data points that belong to that partition. Figure 19.1 illustrates the recursive hyperplane partitioning, and the corresponding decision tree.

Purity The *purity* of a region is defined as the fraction of points with the majority label, i.e.,

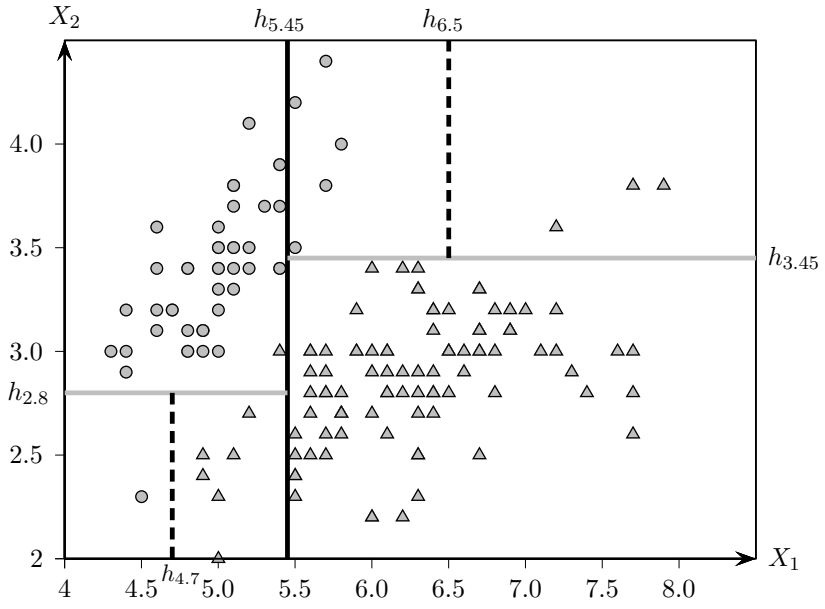
$$purity(\mathcal{D}) = \max_i \left\{ \frac{n_i}{n} \right\} \tag{19.1}$$

where n_i is the number of points $\mathbf{x}$ with class labels $y = c_i$, and n is the total number of points, in the region $\mathcal{D}$.

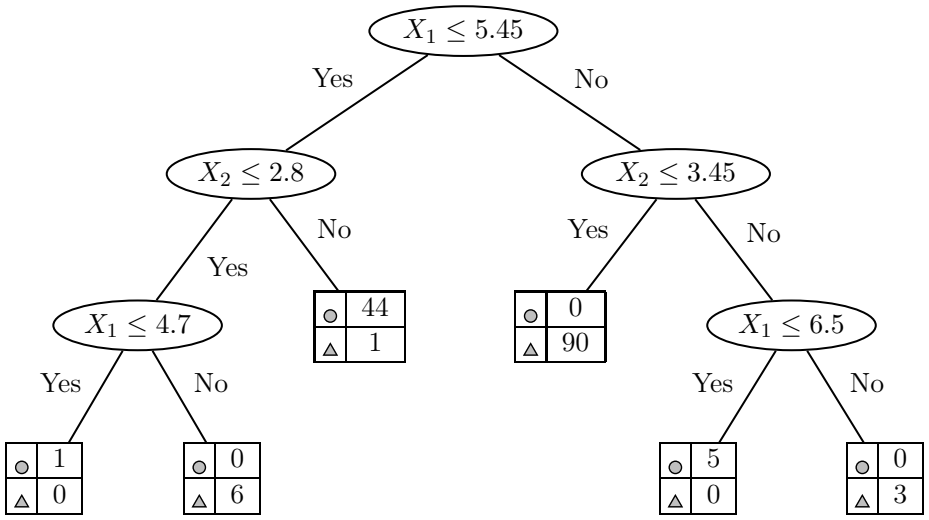
Axis-Parallel Hyperplanes Recall that a hyperplane $h(\mathbf{x})$ is given as

$$h(\mathbf{x}) \colon \mathbf{w}^T\mathbf{x} + b = 0 \tag{19.2}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the *weight vector* that is normal to the hyperplane, and b is the offset of the hyperplane from the origin. Since a decision tree considers only axis-parallel hyperplanes, the weight vector $\mathbf{w}$ is restricted *a priori* to the set of the standard



(a)



(b)

Figure 19.1: Decision Tree Classifier: Recursive Partitioning

basis vectors $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_d\}$, where $\mathbf{e}_i$ has a 1 for the i-th element, and 0 for all other elements. Thus, assuming that $\mathbf{w} = \mathbf{e}_i$, we can rewrite (19.2) as

$$\begin{aligned} & h(\mathbf{x}) \colon \mathbf{e}_i^T \mathbf{x} + b_i = 0 \\ \implies & h(\mathbf{x}) \colon x_i + b_i = 0 \end{aligned} \tag{19.3}$$

where the offset b_i yields different hyperplanes along the i-th dimension.

Split-point A hyperplane specifies a decision or *split-point*, since it splits the data space 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

$$h(\mathbf{x}) \leq 0 \implies x_i + b_i \leq 0 \implies x_i \leq -b_i \tag{19.4}$$

Thus for numeric attributes, a decision tree considers split-points of the form $X_i \leq v$ for each attribute X_i and different values of v.

Example 19.1: Consider the Iris dataset shown in Figure 19.1, that plots `sepal length` (X_1) versus `sepal width` (X_2). The classification task is to discriminate between `iris-setosa` (in circles) and the other two types of irises (in triangles).

The recursive partitioning of the space via axis-parallel hyperplanes is illustrated in part (a), and the corresponding decision tree is shown in part (b). 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 5 and the purity is less than 0.95.

The very first hyperplane to be considered is

$$h_{5.45}(\mathbf{x}) \colon x_1 - 5.45$$

It splits the entire dataset into the two regions $X_1 \leq 5.45$ and $X_1 > 5.45$, and thus corresponds to the decision $X_1 \leq 5.45$ at the root of the decision tree. The two half-spaces are recursively split into smaller half-spaces. For example, the region $X_1 \leq 5.45$ is further split using the hyperplane

$$h_{2.8}(\mathbf{x}) \colon x_2 - 2.8$$

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.

Continuing recursively with $h_{2.8}(\mathbf{x}) \colon x_2 - 2.8$, there are 7 points that satisfy the condition $X_2 \leq 2.8$. Out of these 7 points, 1 is a circle, and 6 are triangles. The size of the region is 7, and the purity is $6/7 = 0.857$. Since the region has more than 5 points, and its purity is less than 0.95, it is further split via

$$h_{4.7}(\mathbf{x}) \colon x_1 - 4.7$$

yielding the left-most decision node $X_1 \leq 4.7$. On the other hand the other half-space $X_2 > 2.8$ has 45 points, out of which only 1 is a triangle. The size of the

region is 45, but the purity is $44/45 = 0.98$. Since 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 is considered to be labeled with the majority class. The frequency for each class is also noted so that the error rate for each leaf can be computed.

In addition to numeric attributes, a decision tree can also handle categorical attributes. Let $dom(X_j)$ denote the domain of values for a categorical attribute X_j. For a categorical attribute, we use a split-point of the form $X_j \in V$, where $V \subset dom(X_j)$. Intuitively, this split can be considered to be the categorical analog of a hyperplane. It results in two "half-spaces" – one region consisting of points $\mathbf{x}$ that satisfy the condition $x_i \in V$, and the other region consisting of points that satisfy the condition $x_i \notin V$.

19.1 Decision Tree Algorithm

Algorithm 19.1: Decision Tree Algorithm

```
DECISIONTREE (D):
1  if (stop condition met) then
2      choose majority class in D as leaf label
3  initialize (split-point*, score*) = ∅ // best split-point
4  foreach (attribute X in D) do
5      if (X is numeric) then
6          (v, score) = Evaluate-Numeric-Attribute(D, X)
7          if score > score* then (split-point*, score*) = (X ≤ v, score)
8      if (X is categorical) then
9          (V, score) = Evaluate-Categorical-Attribute(D, X)
10         if score > score* then (split-point*, score*) = (X ∈ V, score)
11 partition D into D_Y and D_N on split-point*
12 create internal decision node: split-point*
13 create two children nodes: D_Y and D_N
14 DecisionTree(D_Y)
15 DecisionTree(D_N)
```

The decision tree method is shown in Algorithm 19.1. It takes as input a data region $\mathcal{D}$, which is initially the complete dataset $\mathbf{D}$. Different split-points are evaluated for each attribute in $\mathcal{D}$. The best split-point is chosen to partition the data into two subsets: $\mathcal{D}_Y$ and $\mathcal{D}_N$. $\mathcal{D}_Y$ corresponds to all points $\mathbf{x} \in \mathcal{D}$ that satisfy the split

decision, and $\mathcal{D}_N$ corresponds to all points that do not satisfy the split decision. The decision tree method is called recursively on $\mathcal{D}_Y$ and $\mathcal{D}_N$. The partitioning process stops when appropriate stopping conditions are met. Details of the various steps are given next.

19.1.1 Enumerating Split-points

As noted above the form of the split-point depends on the type of attribute.

Numeric Attributes If the j-th dimension or attribute X_j is numeric, then the split-point is of the form $X_j \leq b_j$, corresponding to the hyperplane

$$\begin{aligned} & h(\mathbf{x}) : \mathbf{e}_j^T \mathbf{x} - b_j = 0 \\ \implies & h(\mathbf{x}) : x_j - b_j = 0 \end{aligned}$$

In other words, for the j-th dimension the weight vector is fixed to be the j-th standard basis vector $\mathbf{e}_j$, i.e., the hyperplane is orthogonal to the j-dimension. The main issue concerns the values to consider for the offset b_j. Even if we restrict b_j to lie within the range (2.12) of the attribute X, there are still an infinite number of choices for b_j. One reasonable approach is to consider all the mid-points between distinct values for X_j in the sample $\mathbf{D}$. Since there can be at most n distinct values for X_j, there are at most $n - 1$ mid-point values to consider.

Categorical Attributes If the j-th attribute is categorical, then the split-point is of the form $X_j \in V$, where $V \subset dom(X_j)$ and $V \neq \emptyset$. In words, all distinct partitions of the set of values of X_j are considered. Notice that $X_j \in V$ is simply the reverse of the split-point $X_j \in \overline{V}$, where $\overline{V} = dom(X_j) - V$ is the complement of V. The total number of distinct partitions is given as

$$\sum_{i=1}^{\lfloor m/2 \rfloor} \binom{m}{i} = O(2^{m-1}) \tag{19.5}$$

where m is the number of values in the domain of X_j, i.e., $m = |dom(X_j)|$. Thus, the number of split-points to consider is exponential in m, which can pose problems if m is large. One simplification is to restrict V to be of size one, in which case the split-point is of the form $X_j = v$, where $v \in dom(X_j)$. In this case, the two regions consist of points $\mathbf{x} \in \mathbf{D}$ such that $x_j = v$ and $x_j \neq v$. There are m such split-points, one for each value $v \in dom(X_j)$.

Categorical Splits as Hyperplanes If we model a categorical attribute X_j with $|dom(X_j)| = m$ as a vector random variable $\mathbf{V}_j = (V_1, \ldots, V_m)$, with the only

allowable values as $\mathbf{e}_i$ $(i = 1, \ldots m)$, then a split-point of the form $X_j = v_r$, where v_r is the r-th value in $dom(X_j)$, corresponds to the hyperplane

$$h(\mathbf{x}) : \mathbf{v}_j^T \mathbf{e}_r - 0.5 = 0$$

Here $\mathbf{v}_j$ is the value of $\mathbf{V}_j$ corresponding to the value x_j for the j-th dimension in data point $\mathbf{x}$. The offset $b = -0.5$ corresponds to the mid-point for the Bernoulli variable V_r. If we allow arbitrary sized partitions, then a split of the form $X_j \in V$, with $V = \{v_{i_1}, \ldots, v_{i_r}\}$ of size r, corresponds to the hyperplane

$$h(\mathbf{x}) : \mathbf{v}_j^T \mathbf{e}_\mathbf{i} - 0.5 = 0$$

where $\mathbf{e_i}$ is a m-dimensional (binary) random vector, with ones for the elements given by the index set $\mathbf{i} = (i_1, \ldots, i_r)$, and zeros for all other elements.

19.1.2 Evaluating Split-points

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 labels.

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

In information theory, the entropy of a partition or region $\mathcal{D}$ is defined as follows

$$H(\mathcal{D}) = -\sum_{i=1}^{k} P(c_i|\mathcal{D}) \ \log_2 P(c_i|\mathcal{D}) \tag{19.6}$$

where $P(c_i|\mathcal{D})$ is the probability of class c_i in $\mathcal{D}$, and k is the number of classes. If a region is pure, i.e., has points having 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 $P(c_i|\mathcal{D}) = \frac{1}{k}$, then the entropy has the highest value, $H(\mathcal{D}) = log_2 k$.

Assume that a split-point applied to $\mathcal{D}$ yields the two regions $\mathcal{D}_Y$ and $\mathcal{D}_N$. We can compute the entropy of each of the resulting regions and obtain the resulting entropy of a split-point as follows

$$H(\mathcal{D}_Y, \mathcal{D}_N) = \frac{n_Y}{n} H(\mathcal{D}_Y) + \frac{n_N}{n} H(\mathcal{D}_N) \tag{19.7}$$

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

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

$$Gain(\mathcal{D}, \mathcal{D}_Y, \mathcal{D}_N) = H(\mathcal{D}) - H(\mathcal{D}_Y, \mathcal{D}_N) \tag{19.8}$$

The higher the information gain, the more the reduction in entropy, and the better the split-point. Thus, given split-point and the resulting 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

$$G(\mathcal{D}) = 1 - \sum_{i=1}^{k} P(c_i|\mathcal{D})^2 \tag{19.9}$$

If the partition is pure, then the probability of the majority class is 1, and the probability of all other classes is 0. Thus the Gini-index is 0. On the other hand, when each class is equally represented, with probability $P(c_i|\mathcal{D}) = \frac{1}{k}$, 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

$$G(\mathcal{D}_Y, \mathcal{D}_N) = \frac{n_Y}{n} G(\mathcal{D}_Y) + \frac{n_N}{n} G(\mathcal{D}_N) \tag{19.10}$$

where n, n_Y and n_N denote the number of points in regions $\mathcal{D}$, $\mathcal{D}_Y$ and $\mathcal{D}_N$, respectively. The lower the Gini-index value, the better the split-point. As in the case of information gain, we can define a gini-index based gain as follows

$$Gain_{gini}(\mathcal{D}, \mathcal{D}_Y, \mathcal{D}_N) = G(\mathcal{D}) - G(\mathcal{D}_Y, \mathcal{D}_N)$$

Split-points can be scored accoring to the gini-based gain, so that higher the gain, 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

$$CART(\mathcal{D}_Y, \mathcal{D}_N) = 2\frac{n_Y}{n}\frac{n_N}{n} \sum_{i=1}^{k} |P(c_i|\mathcal{D}_Y) - P(c_i|\mathcal{D}_N)| \tag{19.11}$$

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

19.1.3 Estimating the Class Probability Mass Function

The purity measures for scoring the split-points rely on the class probability mass functions (PMF) for both of the resulting partitions $\mathcal{D}_Y$ and $\mathcal{D}_N$. For each attribute X we have to check many split-points of the form $X \leq v$ (for all distinct mid-points) if X is numeric, and of the form $X \in V$ (for all symbol subsets) if X is categorical. Evaluating each of these split-points independently results in a lot of computational overhead. Instead, one can incrementally compute the class PMF as described below.

Numeric Attributes If X is a numeric attribute, we have to evaluate all split-points of the form $X \leq v$, where v is a mid-point between two successive distinct values for X in the sample $\mathcal{D}$. Let $\{v_1, \cdots, v_m\}$ denote the set of all such mid-points, such that $v_1 < v_2 < \cdots < v_m$. For each split-point $X \leq v_j$, we have to estimate the class PMFs $P(c_i|\mathcal{D}_Y)$ and $P(c_i|\mathcal{D}_N)$.

Let $I()$ be an indicator variable that takes on the value 1 only when the condition inside the brackets is satisfied and is 0 otherwise. Given region $\mathcal{D}$ with n points, the prior probability for each class can be estimated directly from the data as follows

$$\hat{P}(c_i) = \hat{P}(Y = c_i) = \frac{1}{n}\sum_{j=1}^{n} I(y_j = c_i) = \frac{n_i}{n} \tag{19.12}$$

where y_j is the class for point $\mathbf{x}_j$. Here n is the total number of points, and n_i is the number of points with class c_i, in region $\mathcal{D}$.

Given a numeric attribute X in region $\mathcal{D}$, define the class conditional empirical cumulative distribution function (ECDF; see (2.1)) over all the mid-points v for attribute X, as follows

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

where x_j is the value of data point $\mathbf{x}_j$ for the attribute X, and $N_{vi} = \sum_{j=1}^{n} I(x_j \leq v \text{ and } y_j = c_i)$ is the number of points with class c_i with values $x_j \leq v$.

Let $\mathcal{D}_Y$ and $\mathcal{D}_N$ denote the two resulting partitions from a split-point $X \leq v$. The class PMF for $\mathcal{D}_Y$ can be computed from the class conditional ECDF for $\mathcal{D}$, using Bayes theorem

$$\hat{P}(c_i|\mathcal{D}_Y) = \hat{P}(Y = c_i|X \leq v) = \frac{\hat{F}(v|c_i)\hat{P}(c_i)}{\sum_{j=1}^{k} \hat{F}(v|c_j)\hat{P}(c_j)}$$

Plugging in (19.13) and (19.12), we have

$$= \frac{N_{vi}}{\sum_{j=1}^{k} N_{vj}} \tag{19.14}$$

Let $\hat{F}^c(v|c_i) = 1 - \hat{F}(v_j|c_i) = \hat{P}(x > v|c_i)$ be the complement of the ECDF. From (19.13), we get

$$\hat{F}^c(v|c_i) = 1 - \frac{N_{vi}}{n_i} = \frac{n_i - N_{vi}}{n_i} \tag{19.15}$$

Using (19.12) and (19.15), the class PMF $\hat{P}(c_i|\mathcal{D}_N)$ can be computed as follows

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

Algorithm 19.2: Evaluate Numeric Attribute (Using Gain)

```
   EVALUATE-NUMERIC-ATTRIBUTE (D, X):
 1 sort D on attribute X, so that x_j ≤ x_{j+1}, ∀j = 1, ..., n − 1
 2 M = ∅ // set of mid-points
 3 for i = 1, ..., k do n_i = 0
 4 for j = 1, ..., n do
 5     if y_j = c_i then n_i = n_i + 1 // running count for class c_i
 6     if x_{j+1} ≠ x_j then
 7         v = (x_{j+1} + x_j)/2; M = M ∪ {v} // mid-points
 8         for i = 1, ..., k do N_vi = n_i // #points s.t. x_j ≤ v and y_j = c_i
   // evaluate split-points X ≤ v
 9 initialize (v*, score*) = ∅ // best split-point
10 forall v ∈ M do
11     for i = 1, ..., k do P̂(c_i|D_Y) = N_vi / Σ_{j=1}^k N_vj; P̂(c_i|D_N) = (n_i − N_vi) / Σ_{j=1}^k n_j − N_vj
12     score(X ≤ v) = Gain(D, D_Y, D_N)
13     if score(X ≤ v) > score* then v* = v; score* = score(X ≤ v)
14 return (v*, score*)
```

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 mid-point values v and the number of points having value $x_j \leq v$ and class c_i, given as N_{vi}. The for loop on line 10 enumerates all possible split-points of the form $X \leq v$ for each mid-point v, and scores them using the gain criterion (19.8); any of the other scoring criteria can also be used. The best split-point and score are recorded and returned to the main procedure.

Categorical Attributes If X is a categorical attribute we have to compute all splits of the form $X \in V$, where $V \subset dom(X)$. The class conditional empirical PMF for X is given as

$$\hat{P}(v|c_i) = \frac{\hat{P}(X = v \text{ and } Y = c_i)}{\hat{P}(c_i)} = \frac{1}{n_i} \sum_{j=1}^{n} I(x_j = v \text{ and } y_j = c_i) = \frac{n_{vi}}{n_i} \quad (19.17)$$

where $v \in dom(X)$ is one of the symbols or values of attribute X. Here n_{vi} is the number of points $\mathbf{x}_j$, with value $x_j = v$ and class $y_j = c_i$.

The class PMF for the partition $\mathcal{D}_Y$ for the split-point $X \in V$, can be obtained via Bayes theorem

$$\hat{P}(c_i|\mathcal{D}_Y) = \hat{P}(Y = c_i|X \in V) = \frac{\left(\sum_{v \in V} \hat{P}(v|c_i)\right) \hat{P}(c_i)}{\sum_{j=1}^{k} \left(\sum_{v \in V} \hat{P}(v|c_j)\right) \hat{P}(c_j)}$$

Plugging in (19.17), we have

$$= \frac{\sum_{v \in V} \hat{P}(X = v \text{ and } Y = c_i)}{\sum_{j=1}^{k} \sum_{v \in V} \hat{P}(X = v \text{ and } Y = c_j)} = \frac{\sum_{v \in V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \in V} n_{vj}} \quad (19.18)$$

Likewise, for the partition $\mathcal{D}_n$, we have

$$\hat{P}(c_i|\mathcal{D}_N) = \hat{P}(Y = c_i|X \notin V) = \frac{\sum_{v \notin V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \notin V} n_{vj}} \quad (19.19)$$

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 the number of points having value $v \in dom(X)$ and class c_i, given as n_{vi}. The for loop on line 7 enumerates all possible split-points of the form $X \in V$ for $V \subset dom(X)$, and scores them using the gain criterion (19.8). As before, any of the other scoring criteria can also be used. The best split-point and score are recorded and returned to the main decision tree algorithm.

19.1.4 Stopping Criteria

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 $\mathcal{D}$. If the number of points in $\mathcal{D}$ is less than some user-specified size threshold, then we stop the partitioning process and make $\mathcal{D}$ 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, since if the partition is already "pure" then it does not make sense to split the region further. In fact, for better generalization, the recursive partitioning is terminated if the purity of $\mathcal{D}$ (19.1) is above some purity threshold.

Algorithm 19.3: Evaluate Categorical Attribute (Using Gain)

EVALUATE-CATEGORICAL-ATTRIBUTE $(\mathcal{D}, X)$:

for $i = 1, \ldots, k$ **do**
  $n_i = 0$
  forall $v \in dom(X)$ **do** $n_{vi} = 0$
for $j = 1, \ldots, n$ **do**
  if $x_j = v$ *and* $y_j = c_i$ **then** $n_{vi} = n_{vi} + 1$
// `evaluate split-point` $X \in V$
initialize $(V^*, score^*) = \emptyset$ // `best split-point`
forall $V \subset dom(X)$, *such that* $1 \leq |V| \leq |dom(X)|/2$ **do**
  for $i = 1, \ldots, k$ **do**
  $\hat{P}(c_i|\mathcal{D}_Y) = \frac{\sum_{v \in V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \in V} n_{vj}}$; $\hat{P}(c_i|\mathcal{D}_N) = \frac{\sum_{v \notin V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \notin V} n_{vj}}$
  $score(X \in V) = Gain(\mathcal{D}, \mathcal{D}_Y, \mathcal{D}_N)$
  if $score(X \in V) > score^*$ **then** $V^* = V; score^* = score(X \in V)$
return $(V^*, score^*)$

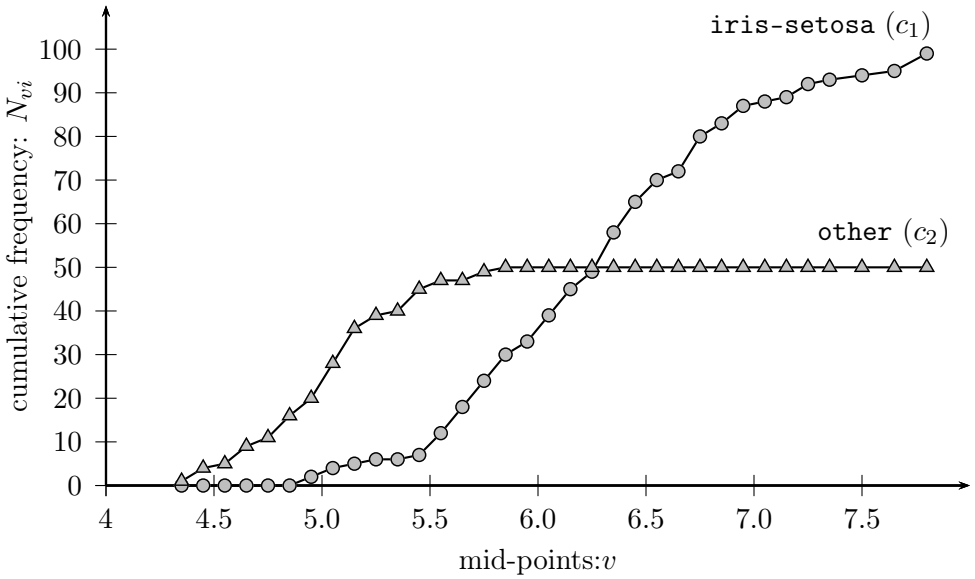


Figure 19.2: Iris: cumulative frequencies N_{vi} for `sepal length`

Example 19.2: 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 $n = 150$ points is considered for the root of the decision tree. The task is to find the best split-point considering both the X_1 (`sepal length`) and X_2 (`sepal width`) attributes.

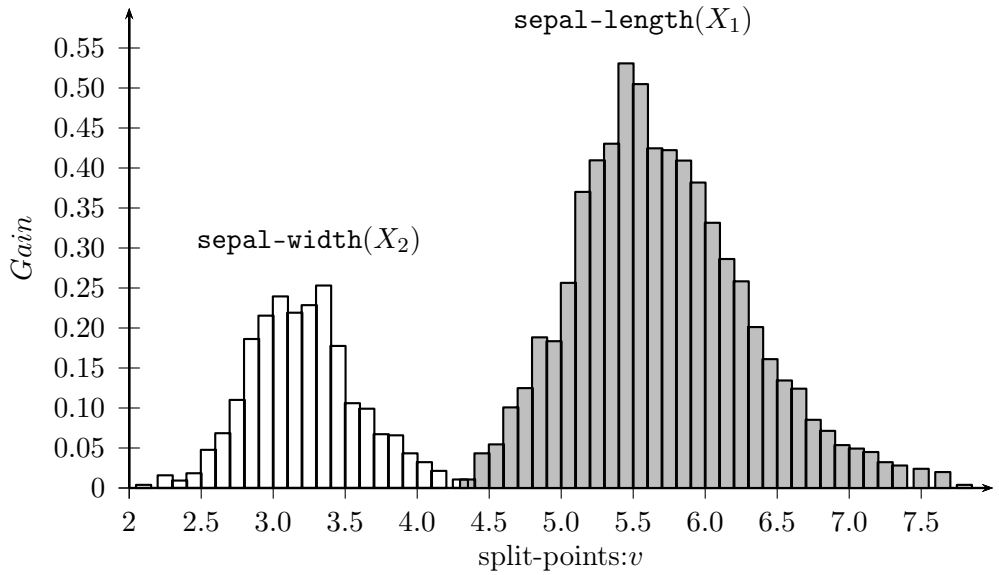


Figure 19.3: Iris: gain for different split-points on `sepal length` and `sepal width`

Since there are $n_1 = 50$ points labeled `iris-setosa`, the second class has $n_2 = 100$ points. We have $\hat{P}(c_1) = \frac{50}{150} = \frac{1}{3}$ and $\hat{P}(c_2) = \frac{100}{150} = \frac{2}{3}$. The entropy (19.6) of the dataset is therefore

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

To evaluate the splits we compute the cumulative frequencies N_{vi} to help estimate the empirical CDF (19.13). Figure 19.2 plots the cumulative frequencies for the two classes for attribute X_1. Plugging in these values into (19.14) and (19.16), we next compute the gain (19.8) for different split-points on X_1 and X_2. For example, consider the split-point $X_1 \leq 5.45$. For $\mathbf{D}_Y$, we have

$$\hat{P}(c_1|\mathbf{D}_Y) = \frac{45}{7+45} = 0.865$$
$$\hat{P}(c_2|\mathbf{D}_Y) = \frac{7}{7+45} = 0.135$$

with its entropy given as

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

For $\mathbf{D}_N$, we have

$$\hat{P}(c_1|\mathbf{D}_N) = \frac{50-45}{(50-45)+(100-7)} = 0.051$$
$$\hat{P}(c_2|\mathbf{D}_N) = \frac{(100-7)}{(50-45)+(100-7)} = 0.949$$

with its entropy given as

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

The entropy of the split-point (19.7) is given as

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

The gain is then given as

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

Figure 19.3 plots the gain values for the different split-points for both X_1 and X_2. We can observe that $X \leq 5.45$ is the best split-point, and it is this 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. In this example, we use a leaf size threshold of 5 and a purity threshold of 0.95.

bins	domain	class frequencies (n_{vi})	
		`iris-setosa`	`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

Table 19.1: Discretized `sepal length` Attribute: class frequencies

Example 19.3 (Categorical Attributes): Let us assume that `sepal length` has been discretized as shown in Table 19.1. The class frequencies n_{vi} are also shown, which can be used to compute the class PMF (19.17).

As noted in Example 19.2 the entropy of the whole dataset $\mathbf{D}$ is $H(\mathbf{D}) = 0.918$. Consider the split-point $X_1 \in \{a_1, a_3\}$. From table 19.2 we can compute the class PMF for partition $\mathbf{D}_Y$ (19.18) as follows

$$\hat{P}(c_1|\mathbf{D}_Y) = \frac{39 + 0}{(39 + 0) + (6 + 43)} = 0.443$$
$$\hat{P}(c_2|\mathbf{D}_Y) = \frac{6 + 43}{(39 + 0) + (6 + 43)} = 0.557$$

V	split entropy	Gain
$\{a_1\}$	0.509	0.41
$\{a_2\}$	0.897	0.217
$\{a_3\}$	0.711	0.207
$\{a_4\}$	0.869	0.049
$\{a_1, a_2\}$	0.632	0.286
$\{a_1, a_3\}$	0.86	0.058
$\{a_1, a_4\}$	0.667	0.251
$\{a_2, a_3\}$	0.667	0.251
$\{a_2, a_4\}$	0.86	0.058
$\{a_3, a_4\}$	0.632	0.286

Table 19.2: Categorical split-points for `sepal length`

with the entropy given as

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

Likewise the class PMF for $\mathbf{D}_N$ (19.19) is given as

$$\hat{P}(c_1|\mathbf{D}_N) = \frac{11+0}{(11+0)+(39+12)} = 0.177$$
$$\hat{P}(c_2|\mathbf{D}_N) = \frac{39+12}{(11+0)+(39+12)} = 0.823$$

with the entropy given as

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

The entropy of the split is given as

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

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.

19.1.5 Computational Complexity

To analyze the computational complexity of the decision tree method in Algorithm 19.1, we first analyze the cost of evaluating all the split-points for a numeric or categorical attribute.

For numeric attributes consider Algorithm 19.2. The initial sorting (line 1) takes time $O(n \log_2 n)$. The cost of computing the mid-points and the class-specific cumulative counts N_{vi} takes time $O(nk)$ (for loop on line 4). The cost of computing the score is also bounded by $O(nk)$, since the total number of mid-points v can be at most n (for loop on line 10). The total cost of evaluating a numeric attribute is therefore $O(n \log n + nk)$. Ignoring k, since it is usually a small constant, the total cost of numeric split-point evaluation is bounded as $O(n \log n)$. It is in fact possible to eliminate the sorting cost (line 1), by initially creating a projected dataset $\mathbf{D}(X)$ for each numeric attribute X. The projected dataset $\mathcal{D}(X)$ is always maintained in sorted order, even as it is recursively split into smaller parts $\mathcal{D}_Y(X)$ and $\mathcal{D}_N(X)$. The cost for numeric attributes can therefore be reduced to just $O(n)$.

For categorical attributes consider Algorithm 19.3. Computing the class-specific counts for each value n_{vi} takes $O(n)$ time (for loop on line 4). Let $m = |dom(X)|$, then as per (19.5) the number of partitions V to consider is $O(2^{m-1})$. Each split-point can be evaluated in time $O(mk)$, thus the for loop in line 7 takes time $O(mk2^{m-1})$. The total cost for categorical attributes is therefore $O(n + mk2^{m-1})$. We make the assumption that $2^{m-1} = O(n)$ (or $m = O(\log n)$), and thus, ignoring k, the cost of categorical splits is bounded as $O(n \log n)$. In case the assumption about m is not satisfied, we can consider split-points with $|V| = 1$, in which case there are only m split-points to consider.

Starting from the initial invocation of the decision tree algorithm with $\mathcal{D} = \mathbf{D}$ (with $|\mathbf{D}| = n$), the method evaluates all d attributes, with cost $(dn \log n)$. The partitioning of $\mathcal{D}$ into $\mathcal{D}_Y$ and $\mathcal{D}_N$ takes $O(n)$ time. Generalizing, each invocation of the method takes time $O(dn' \log n')$, where n' is the current size of the partition $\mathcal{D}$. The total cost thus depends on the depth of the decision tree. Let us assume that the smaller partition, say $\mathcal{D}_Y$, has at least a fraction r of the n' points. In this case the tree has depth $log_c n$, with $c = \frac{1}{1-r}$. Thus the total cost of decision tree construction is bounded as $O(dn \log_c^2 n)$ in the worst case.

19.2 Decision Tree Pruning

Overfitting