# 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 $y_i \in \{c_1, c_2, \ldots, c_k\}$. 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 $\mathcal{R}_1, \ldots, \mathcal{R}_6$, constitute the decision tree model. Note also the induced partitioning of the input points into these six regions.

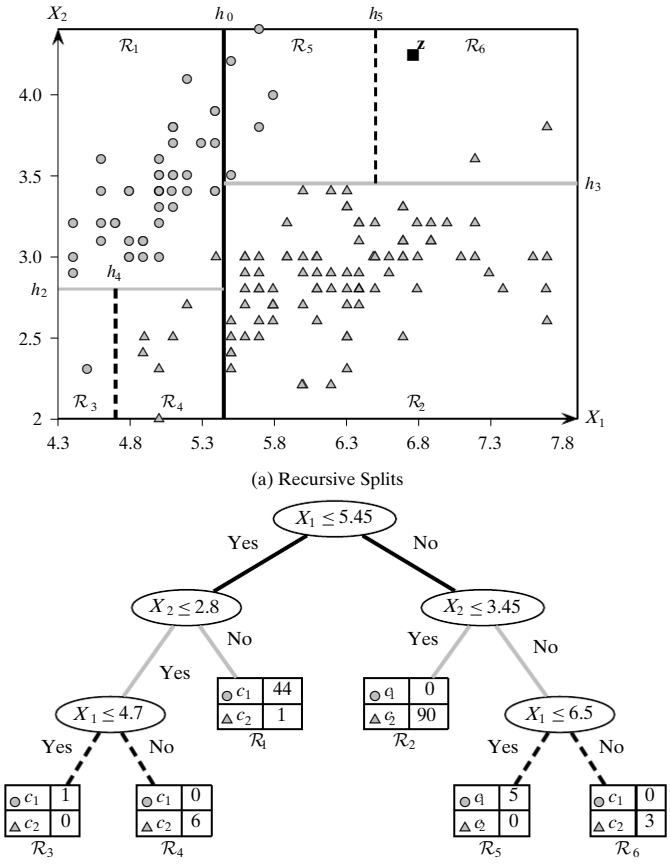


**Figure 19.1.** Decision trees: recursive partitioning via axis-parallel hyperplanes.

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

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

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 $j$th 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

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

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,

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

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 $\eta$ and $\pi$, where $\eta$ is the leaf size and $\pi$ 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 (D, η, π):
n ← |D| // partition size
n_i ← |{x_j | x_j ∈ D, y_j = c_i}| // size of class c_i
purity(D) ← max_i {n_i / n}
if n ≤ η or purity(D) ≥ π then // stopping condition
    c* ← arg max_{c_i} {n_i / n} // majority class
    create leaf node, and label it with class c*
    return
(split point*, score*) ← (∅, 0) // initialize best split point
foreach (attribute X_j) do
    if (X_j is numeric) then
        (v, score) ← EVALUATE-NUMERIC-ATTRIBUTE(D, X_j)
        if score > score* then (split point*, score*) ← (X_j ≤ v, score)
    else if (X_j is categorical) then
        (V, score) ← EVALUATE-CATEGORICAL-ATTRIBUTE(D, X_j)
        if score > score* then (split point*, score*) ← (X_j ∈ V, score)
// partition D into D_Y and D_N using split point*, and call recursively
D_Y ← {x^T | x ∈ D satisfies split point*}
D_N ← {x^T | x ∈ D does not satisfy split point*}
create internal node split point*, with two child nodes, D_Y and D_N
DECISIONTREE(D_Y, η, π); DECISIONTREE(D_N, η, π)
```

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 $\eta$, 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 $\pi$. 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:

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

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

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

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:

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

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:

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

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:

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