Data Mining and Machine Learning: Fundamental Concepts and Algorithms

dataminingbook.info

Mohammed J. Zaki[1] Wagner Meira Jr.[2]

[1]Department of Computer Science
Rensselaer Polytechnic Institute, Troy, NY, USA

[2]Department of Computer Science
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Chapter 19: Decision Tree Classifier

Decision Tree Classifier

Let the training dataset $\boldsymbol{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ consist of n points in a d-dimensional space, with y_i being the class label for point $\boldsymbol{x}_i$.

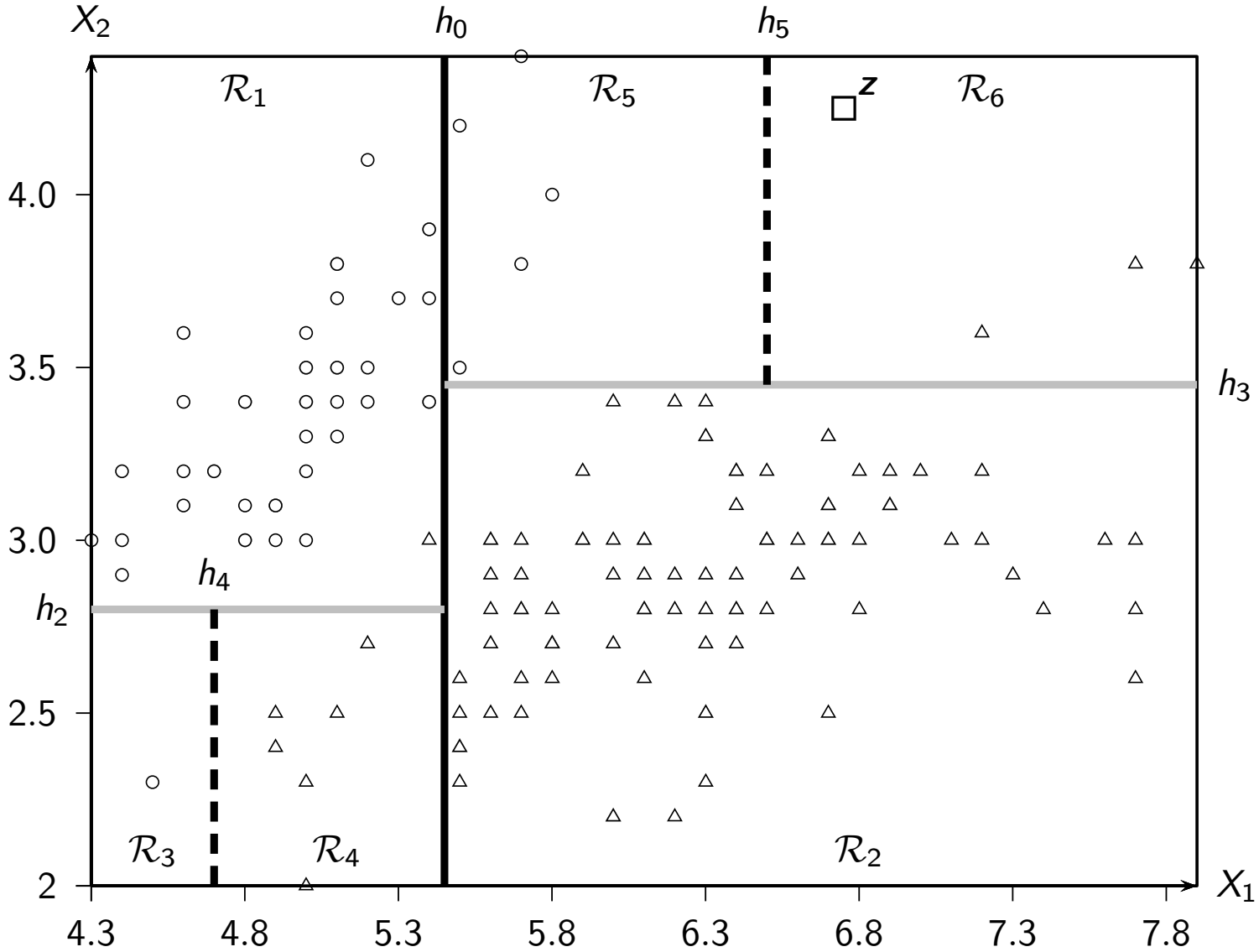
A decision tree classifier is a recursive, partition-based tree model that predicts the class $\hat{y}_i$ for each point $\boldsymbol{x}_i$.

Let $\mathcal{R}$ denote the data space that encompasses the set of input points $\boldsymbol{D}$. 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 $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$, respectively.

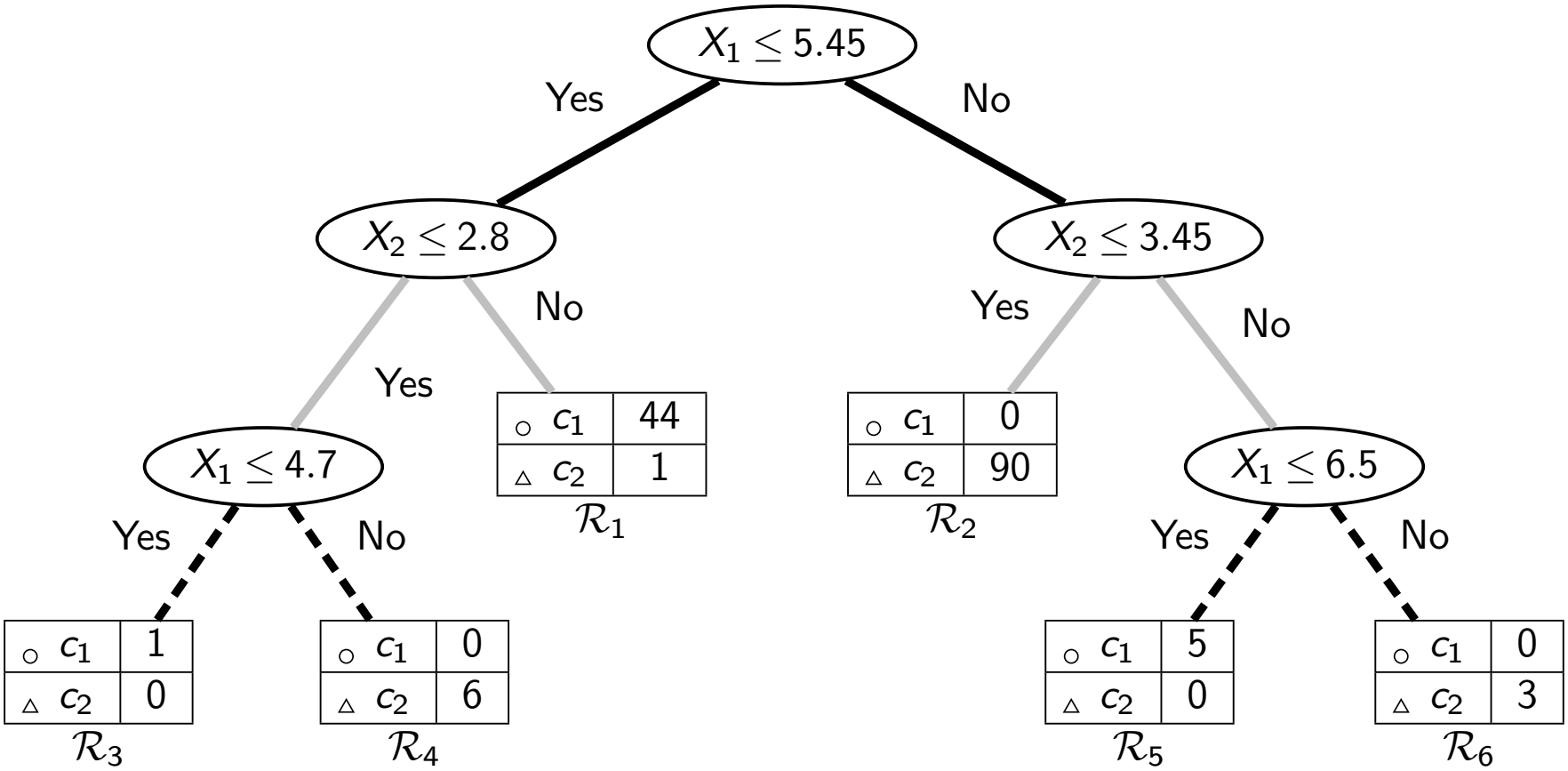
Each of these regions is recursively split via axis-parallel hyperplanes until most of the points belong to the same class.

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.

Decision Tree: Recursive Splits



Decision Tree



Decision Trees: Axis-Parallel Hyperplanes

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

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

where $\boldsymbol{w} \in \mathbb{R}^d$ 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:

$$h(\boldsymbol{x}): x_j + b = 0$$

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

Decision Trees: Split Points

A hyperplane specifies a decision or *split point* because it splits the data space $\mathcal{R}$ into two half-spaces. All points $\boldsymbol{x}$ such that $h(\boldsymbol{x}) \leq 0$ are on the hyperplane or to one side of the hyperplane, whereas all points such that $h(\boldsymbol{x}) > 0$ are on the other side.

The split point is written as $h(\boldsymbol{x}) \leq 0$, i.e.

$$X_j \leq v$$

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.

Categorical Attributes: 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.

Decision Trees: Data Partition and Purity

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

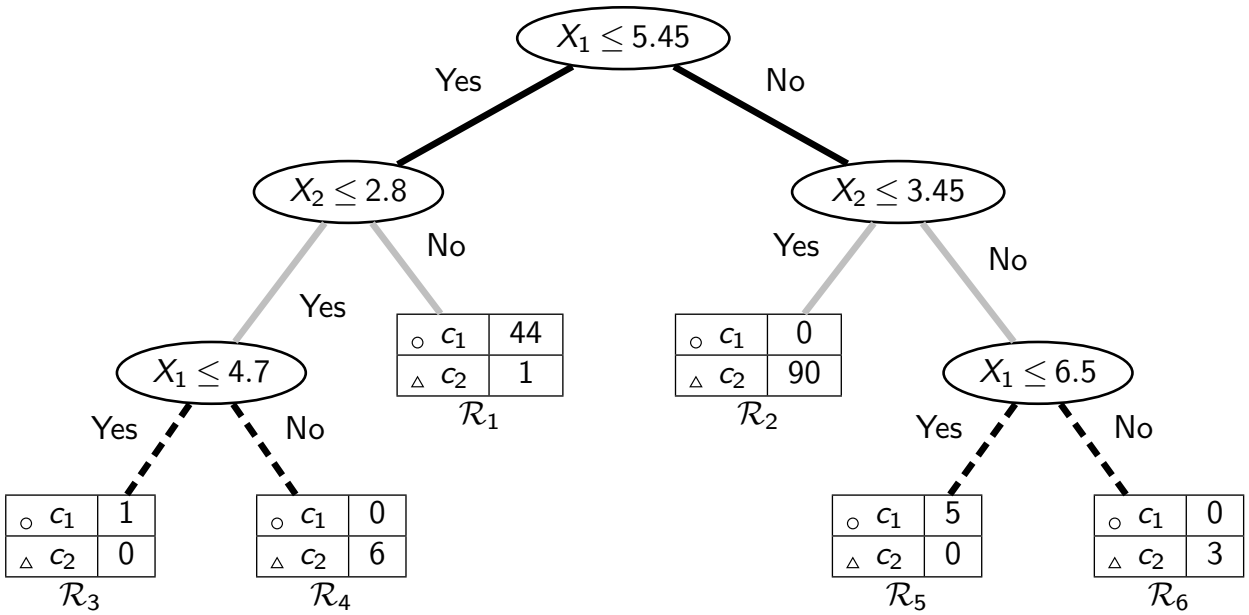
$$\boldsymbol{D}_Y = \{\boldsymbol{x} \mid \boldsymbol{x} \in \boldsymbol{D}, x_j \leq v\}$$
$$\boldsymbol{D}_N = \{\boldsymbol{x} \mid \boldsymbol{x} \in \boldsymbol{D}, x_j > v\}$$

The purity of a region $\mathcal{R}_j$ is the fraction of points with the majority label in $\boldsymbol{D}_j$, that is,

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

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

Decision Trees to Rules



A tree is a set of decision rules; each comprising the decisions on the path to a leaf:

$\mathcal{R}_3$: If $X_1 \leq 5.45$ and $X_2 \leq 2.8$ and $X_1 \leq 4.7$, then class is c_1, or

$\mathcal{R}_4$: If $X_1 \leq 5.45$ and $X_2 \leq 2.8$ and $X_1 > 4.7$, then class is c_2, or

$\mathcal{R}_1$: If $X_1 \leq 5.45$ and $X_2 > 2.8$, then class is c_1, or

$\mathcal{R}_2$: If $X_1 > 5.45$ and $X_2 \leq 3.45$, then class is c_2, or

$\mathcal{R}_5$: If $X_1 > 5.45$ and $X_2 > 3.45$ and $X_1 \leq 6.5$, then class is c_1, or

$\mathcal{R}_6$: If $X_1 > 5.45$ and $X_2 > 3.45$ and $X_1 > 6.5$, then class is c_2

Decision Tree Algorithm

The method takes as input a training dataset $\boldsymbol{D}$, 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 $\boldsymbol{D}$. 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, $\boldsymbol{D}_Y$ and $\boldsymbol{D}_N$, where $\boldsymbol{D}_Y$ corresponds to all points $\boldsymbol{x} \in \boldsymbol{D}$ that satisfy the split decision, and $\boldsymbol{D}_N$ corresponds to all points that do not satisfy the split decision. The decision tree method is then called recursively on $\boldsymbol{D}_Y$ and $\boldsymbol{D}_N$.

We stop the process if the leaf size drops below η or if the purity is at least π.

Decision Tree Algorithm

DecisionTree $(\boldsymbol{D}, \eta, \pi)$**:**

$n \leftarrow |\boldsymbol{D}|$ // partition size
$n_i \leftarrow |\{\boldsymbol{x}_j | \boldsymbol{x}_j \in \boldsymbol{D}, y_j = c_i\}|$ // size of class c_i
$purity(\boldsymbol{D}) \leftarrow \max_i \left\{ \frac{n_i}{n} \right\}$
if $n \leq \eta$ *or* $purity(\boldsymbol{D}) \geq \pi$ **then** // stopping condition
    $c^* \leftarrow \arg\max_{c_i} \left\{ \frac{n_i}{n} \right\}$ // majority class
    create leaf node, and label it with class c^*
    return
$(split\ point^*, score^*) \leftarrow (\emptyset, 0)$ // initialize best split point
foreach *(attribute* X_j*)* **do**
    if *(*X_j *is numeric)* **then**
        $(v, score) \leftarrow$ Evaluate-Numeric-Attribute$(\boldsymbol{D}, X_j)$
        if $score > score^*$ **then** $(split\ point^*, score^*) \leftarrow (X_j \leq v, score)$
    else if *(*X_j *is categorical)* **then**
        $(V, score) \leftarrow$ Evaluate-Categorical-Attribute$(\boldsymbol{D}, X_j)$
        if $score > score^*$ **then** $(split\ point^*, score^*) \leftarrow (X_j \in V, score)$
$\boldsymbol{D}_Y \leftarrow \{\boldsymbol{x} \in \boldsymbol{D} \mid \boldsymbol{x}$ satisfies $split\ point^*\}$
$\boldsymbol{D}_N \leftarrow \{\boldsymbol{x} \in \boldsymbol{D} \mid \boldsymbol{x}$ does not satisfy $split\ point^*\}$
create internal node $split\ point^*$, with two child nodes, $\boldsymbol{D}_Y$ and $\boldsymbol{D}_N$
DecisionTree$(\boldsymbol{D}_Y)$; DecisionTree$(\boldsymbol{D}_N)$

Split Point Evaluation Measures: Entropy

Intuitively, we want to select a split point that gives the best separation or discrimination between the different class labels.

Entropy measures the amount of disorder or uncertainty in a system. 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 $\boldsymbol{D}$ is defined as follows:

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

where $P(c_i|\boldsymbol{D})$ is the probability of class c_i in $\boldsymbol{D}$, 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 $P(c_i|\boldsymbol{D}) = \frac{1}{k}$, then the entropy has the highest value, $H(\boldsymbol{D}) = \log_2 k$.

Split Point Evaluation Measures: Entropy

Define the *split entropy* as the weighted entropy of each of the resulting partitions

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

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

Define the *information gain* for a split point as

$$Gain(\boldsymbol{D}, \boldsymbol{D}_Y, \boldsymbol{D}_N) = H(\boldsymbol{D}) - H(\boldsymbol{D}_Y, \boldsymbol{D}_N)$$

The higher the information gain, the more the reduction in entropy, and the better the split point.

We score each split point and choose the one that gives the highest information gain.

Split Point Evaluation Measures

Gini Index and CART Measure

Gini Index: The Gini index is defined as follows:

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

If the partition is pure, then Gini index is 0. The weighted Gini index is:

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

The lower the Gini index value, the better the split point.

CART: The CART measure is

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

CART maximizes the difference between the class PMF for the two partitions; the higher the CART measure, the better the split point.

Evaluating Split Points: Numeric Attributes

All of the split point evaluation measures depend on the class probability mass function (PMF) for $\boldsymbol{D}$, namely, $P(c_i|\boldsymbol{D})$, and the class PMFs for the resulting partitions $\boldsymbol{D}_Y$ and $\boldsymbol{D}_N$, namely $P(c_i|\boldsymbol{D}_Y)$ and $P(c_i|\boldsymbol{D}_N)$.

We have to evaluate split points of the form $X \leq v$. We consider only the midpoints between two successive distinct values for X in the sample $\boldsymbol{D}$. 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 $X \leq v$, we have to estimate the class PMFs:

$$\hat{P}(c_i|\boldsymbol{D}_Y) = \hat{P}(c_i|X \leq v)$$
$$\hat{P}(c_i|\boldsymbol{D}_N) = \hat{P}(c_i|X > v)$$

Using Bayes theorem, we have

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

Thus we have to estimate the prior probability and likelihood for each class in each partition.

Evaluating Split Points: Numeric Attributes

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

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

where y_j is the class for point $\boldsymbol{x}_j$, $n = |\boldsymbol{D}|$ is the total number of points, and n_i is the number of points in $\boldsymbol{D}$ 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 $\boldsymbol{x}_j$ for the attribute X, given as

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

Evaluating Split Points: Numeric Attributes

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

$$\hat{P}(X \leq v|c_i) = \frac{N_{vi}}{n_i}$$

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

Finally, we have

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

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

The total cost of evaluating a numeric attribute is $O(n \log n + nk)$, where k is the number of classes, and n is the number of points.

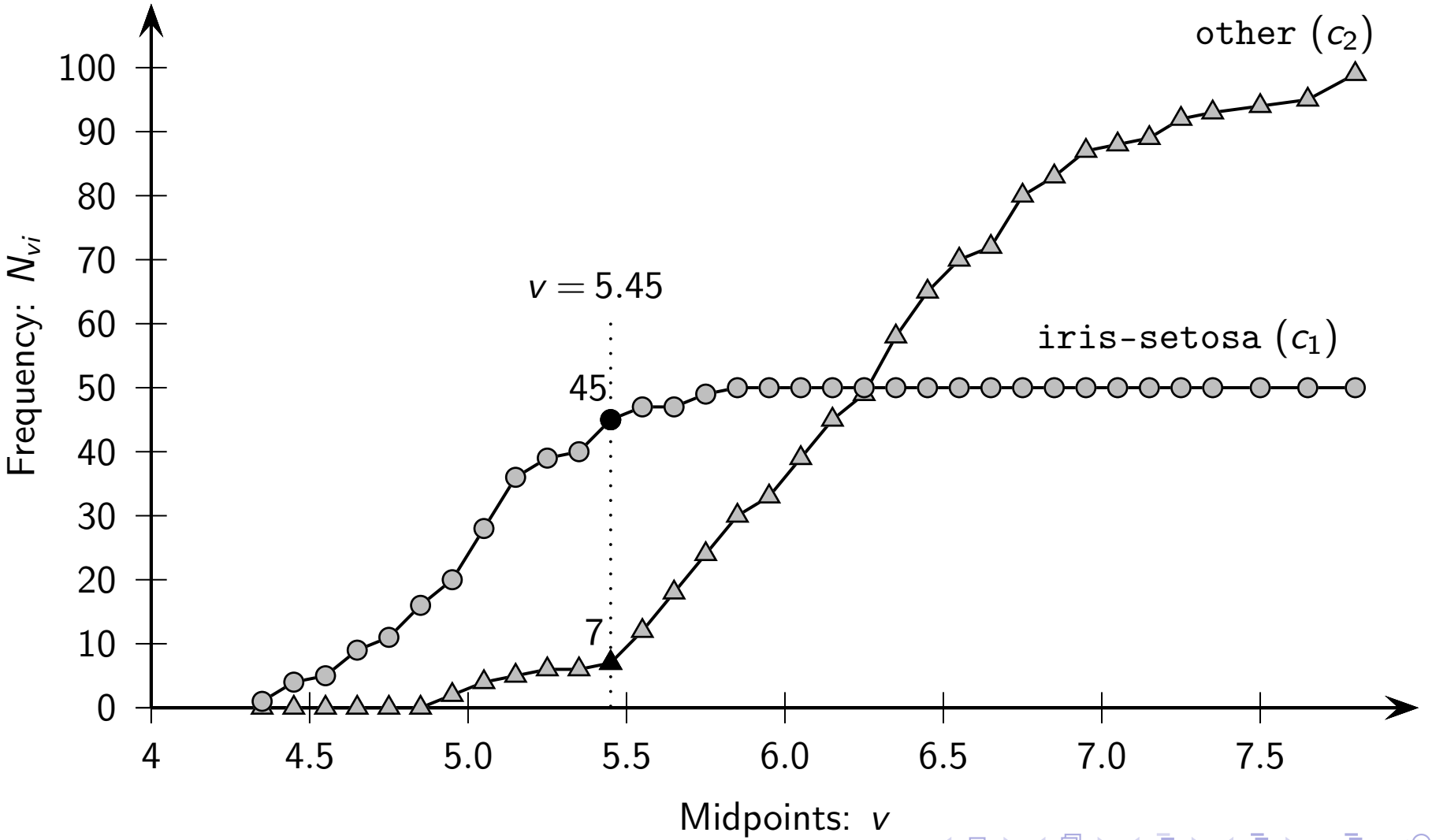
Algorithm Evaluate-Numeric-Attribute

Evaluate-Numeric-Attribute $(\boldsymbol{D}, X)$:

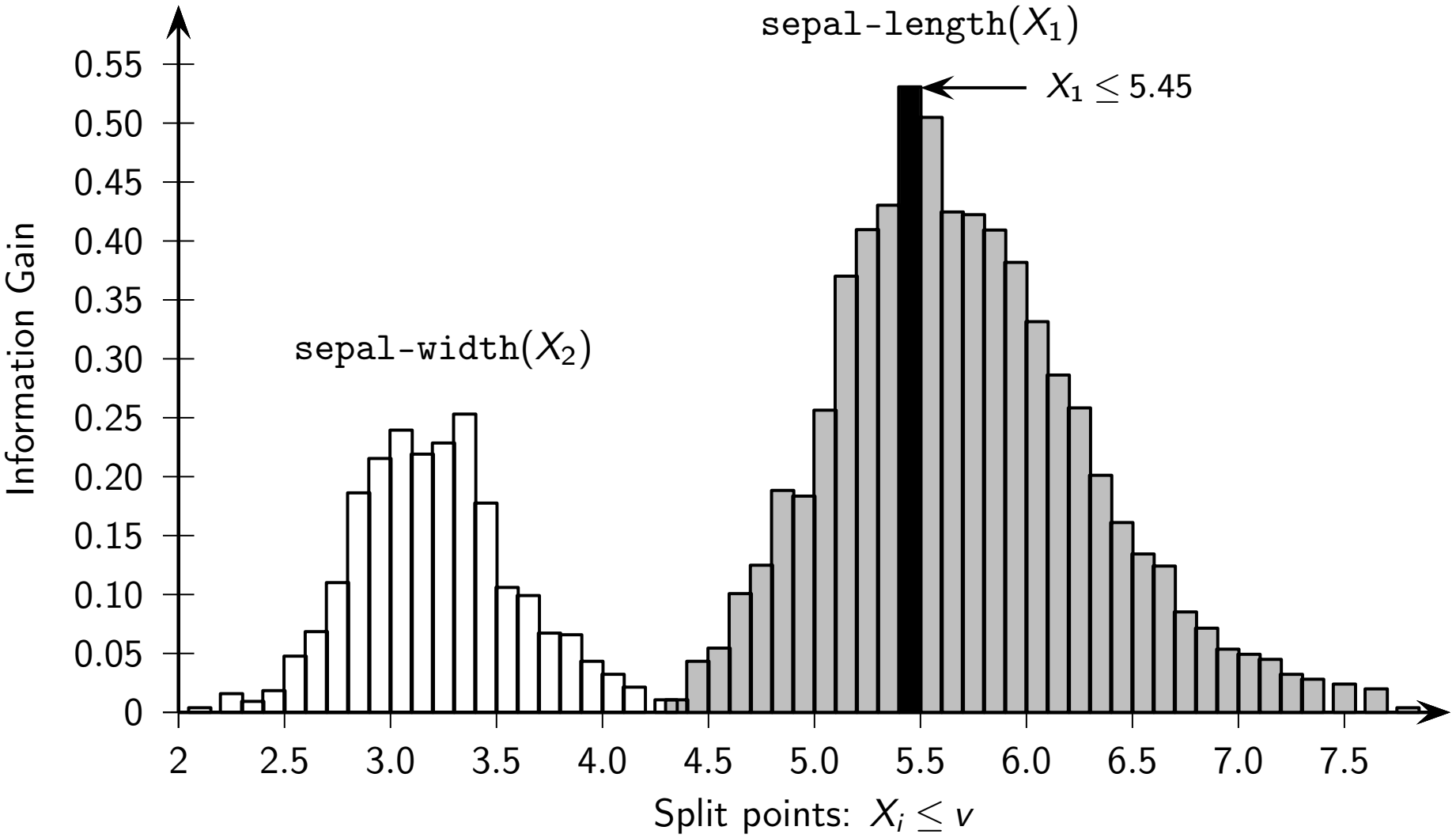
1 sort $\boldsymbol{D}$ on attribute X, so that $x_j \leq x_{j+1}, \forall j = 1, \ldots, n-1$
2 $\mathcal{M} \leftarrow \emptyset$ // set of midpoints
3 **for** $i = 1, \ldots, k$ **do** $n_i \leftarrow 0$
4 **for** $j = 1, \ldots, n-1$ **do**
5 **if** $y_j = c_i$ **then** $n_i \leftarrow n_i + 1$
//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

Iris Data: Class-specific Frequencies N_{vi}

Classes c_1 and c_2 for attribute `sepal length`



Iris Data: Information Gain for Different Splits



Categorical Attributes

For categorical X the split points are of the form $X \in V$, where $V \subset dom(X)$ and $V \neq \emptyset$. All distinct partitions of the set of values of X are considered.

If $m = |dom(X)|$, then there are $O(2^{m-1})$ distinct partitions, which can be too many. 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 $v \in dom(X_j)$.

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

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

The class conditional empirical PMF for X is then given as

$$\hat{P}(X = v|c_i) = \frac{\hat{P}(X = v \text{ and } c_i)}{\hat{P}(c_i)} = \frac{n_{vi}}{n_i}$$

We then have

$$\hat{P}(c_i|\boldsymbol{D}_Y) = \frac{\sum_{v \in V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \in V} n_{vj}} \qquad \hat{P}(c_i|\boldsymbol{D}_N) = \frac{\sum_{v \notin V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \notin V} n_{vj}}$$

Algorithm Evaluate-Categorical-Attribute

Evaluate-Categorical-Attribute (D, X, l):

1 **for** $i = 1, \ldots, k$ **do**
2 $\quad n_i \leftarrow 0$
3 $\quad$ **forall** $v \in dom(X)$ **do** $n_{vi} \leftarrow 0$
4 **for** $j = 1, \ldots, n$ **do**
5 $\quad$ **if** $x_j = v$ *and* $y_j = c_i$ **then** $n_{vi} \leftarrow n_{vi} + 1$ // frequency statistics
6

// evaluate split points of the form $X \in V$
7 $V^* \leftarrow \emptyset$; $score^* \leftarrow 0$ // initialize best split point
8 **forall** $V \subset dom(X)$, *such that* $1 \leq |V| \leq l$ **do**
9 $\quad$ **for** $i = 1, \ldots, k$ **do**
10 $\quad\quad \hat{P}(c_i|\boldsymbol{D}_Y) \leftarrow \frac{\sum_{v \in V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \in V} n_{vj}}$
11 $\quad\quad \hat{P}(c_i|\boldsymbol{D}_N) \leftarrow \frac{\sum_{v \notin V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \notin V} n_{vj}}$
12 $\quad score(X \in V) \leftarrow Gain(\boldsymbol{D}, \boldsymbol{D}_Y, \boldsymbol{D}_N)$
13 $\quad$ **if** $score(X \in V) > score^*$ **then**
14 $\quad\quad V^* \leftarrow V; score^* \leftarrow score(X \in V)$
15 **return** $(V^*, score^*)$

Discretized sepal length: 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

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

Best split: $X \in \{a_1\}$.

Data Mining and Machine Learning: Fundamental Concepts and Algorithms

dataminingbook.info

Mohammed J. Zaki[1] Wagner Meira Jr.[2]

[1]Department of Computer Science
Rensselaer Polytechnic Institute, Troy, NY, USA

[2]Department of Computer Science
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Chapter 19: Decision Tree Classifier