Chapter IX: Classification

Information Retrieval & Data Mining
Universität des Saarlandes, Saarbrücken
Winter Semester 2013/14

Chapter IX: Classification*

1. Basic idea
2. Decision trees
3. Naïve Bayes classifier
4. Support vector machines
5. Ensemble methods

* Zaki & Meira: Ch. 18, 19, 21 & 22; Tan, Steinbach & Kumar: Ch. 4, 5.3–5.6

IX.1 Basic idea

1. Definitions

1.1. Data

1.2. Classification function

1.3. Predictive vs. descriptive

1.4. Supervised vs. unsupervised

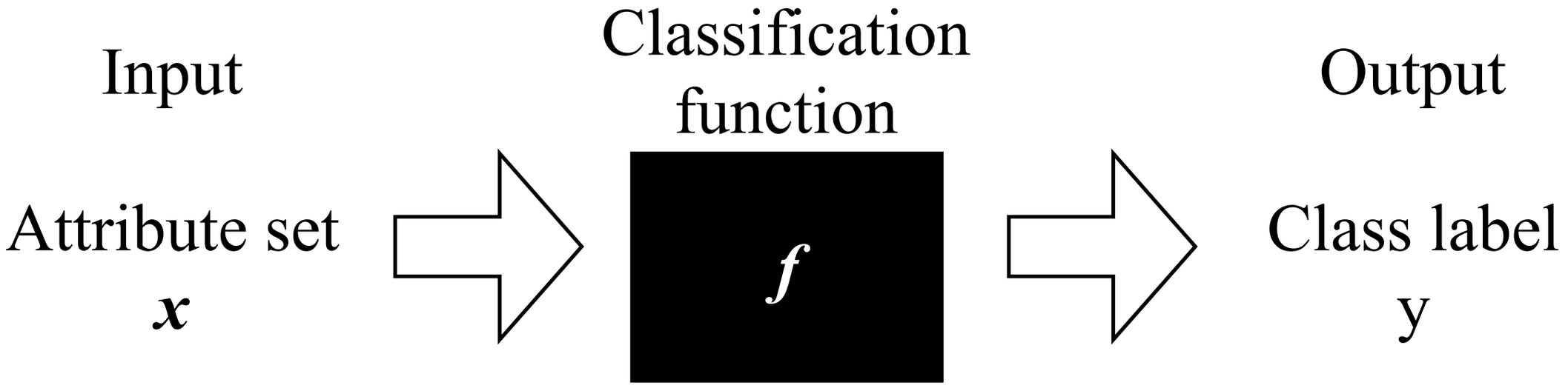
Definitions

- *Data* for classification comes in tuples ($\boldsymbol{x}$, y)
  - Vector $\boldsymbol{x}$ is the **attribute (feature) set**
    - Attributes can be binary, categorical or numerical
  - Value y is the **class label**
    - We concentrate on binary or nominal class labels
    - Compare classification with regression!
- A **classifier** is a function that maps attribute sets to class labels, $f(\boldsymbol{x}) = y$

attribute set class

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Classification function as a black box



Descriptive vs. predictive

- In **descriptive** data mining the goal is to give a description of the data
  - Those who have bought diapers have also bought beer
  - These are the clusters of documents from this corpus
- In **predictive** data mining the goal is to predict the future
  - Those who will buy diapers will also buy beer
  - If new documents arrive, they will be similar to one of the cluster centroids
- The difference between predictive data mining and machine learning is hard to define

Descriptive vs. predictive classification

- Who are the borrowers that will default?
  - Descriptive
- If a new borrower comes, will they default?
  - Predictive
- Predictive classification is the usual application
  - What we will concentrate on

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

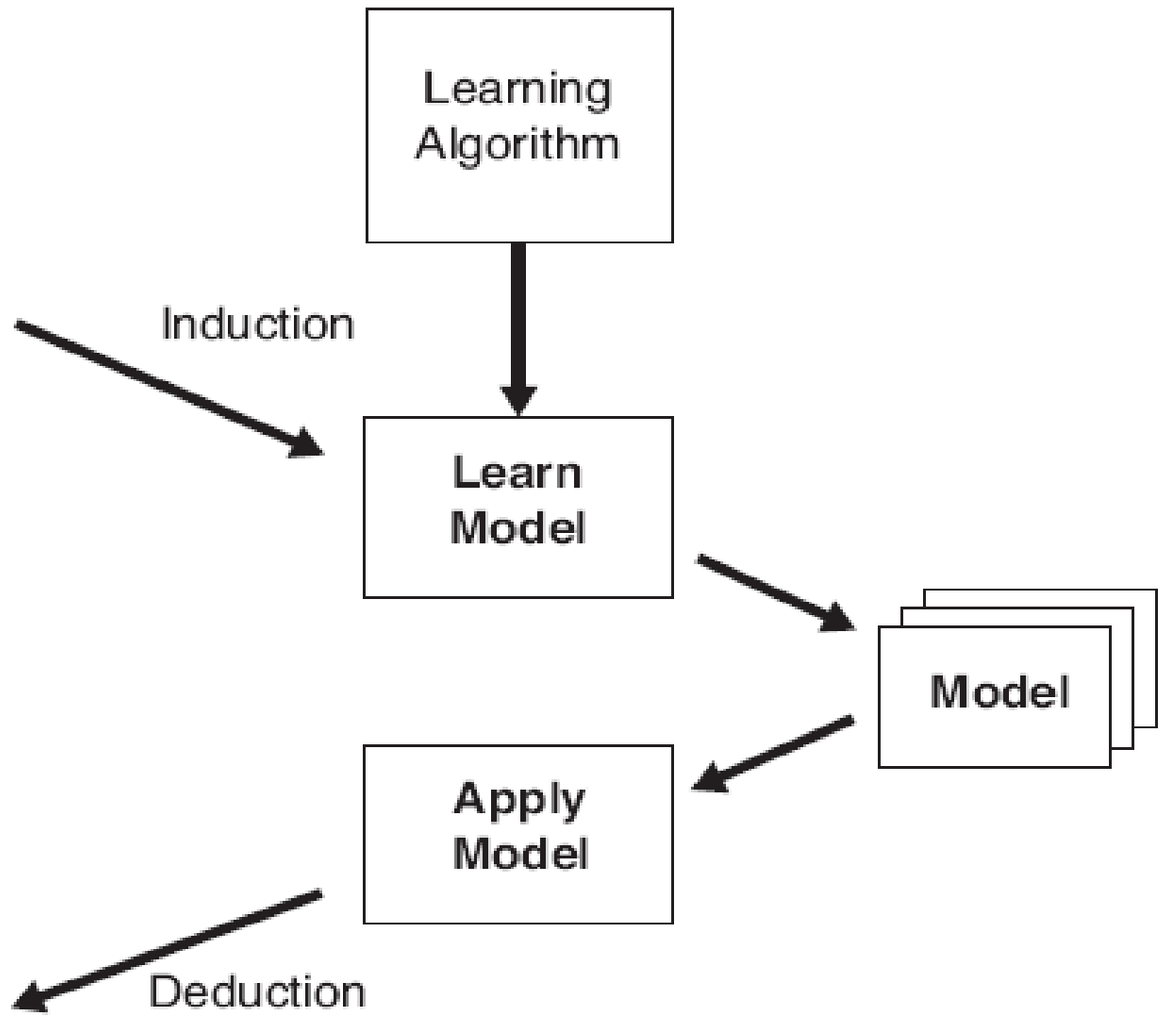
General classification framework

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Test Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?



Classification model evaluation

- Recall the *confusion matrix*:
- Much the same measures as with IR methods
  - Focus on *accuracy* and *error rate*

Predicted class

Actual class		Class = 1	Class = 0
	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

$$\text{Accuracy} = \frac{f_{11} + f_{00}}{f_{11} + f_{00} + f_{10} + f_{01}}$$

$$\text{Error rate} = \frac{f_{10} + f_{01}}{f_{11} + f_{00} + f_{10} + f_{01}}$$

  - But also precision, recall, F-scores, …

Supervised vs. unsupervised learning

- In **supervised learning**
  - Training data is accompanied by class labels
  - New data is classified based on the training set
    - Classification
- In **unsupervised learning**
  - The class labels are unknown
  - The aim is to establish the existence of classes in the data based on measurements, observations, etc.
    - Clustering

IX.2 Decision trees

1. Basic idea
2. Hunt's algorithm
3. Selecting the split

Zaki & Meira: Ch. 19; Tan, Steinbach & Kumar: Ch. 4

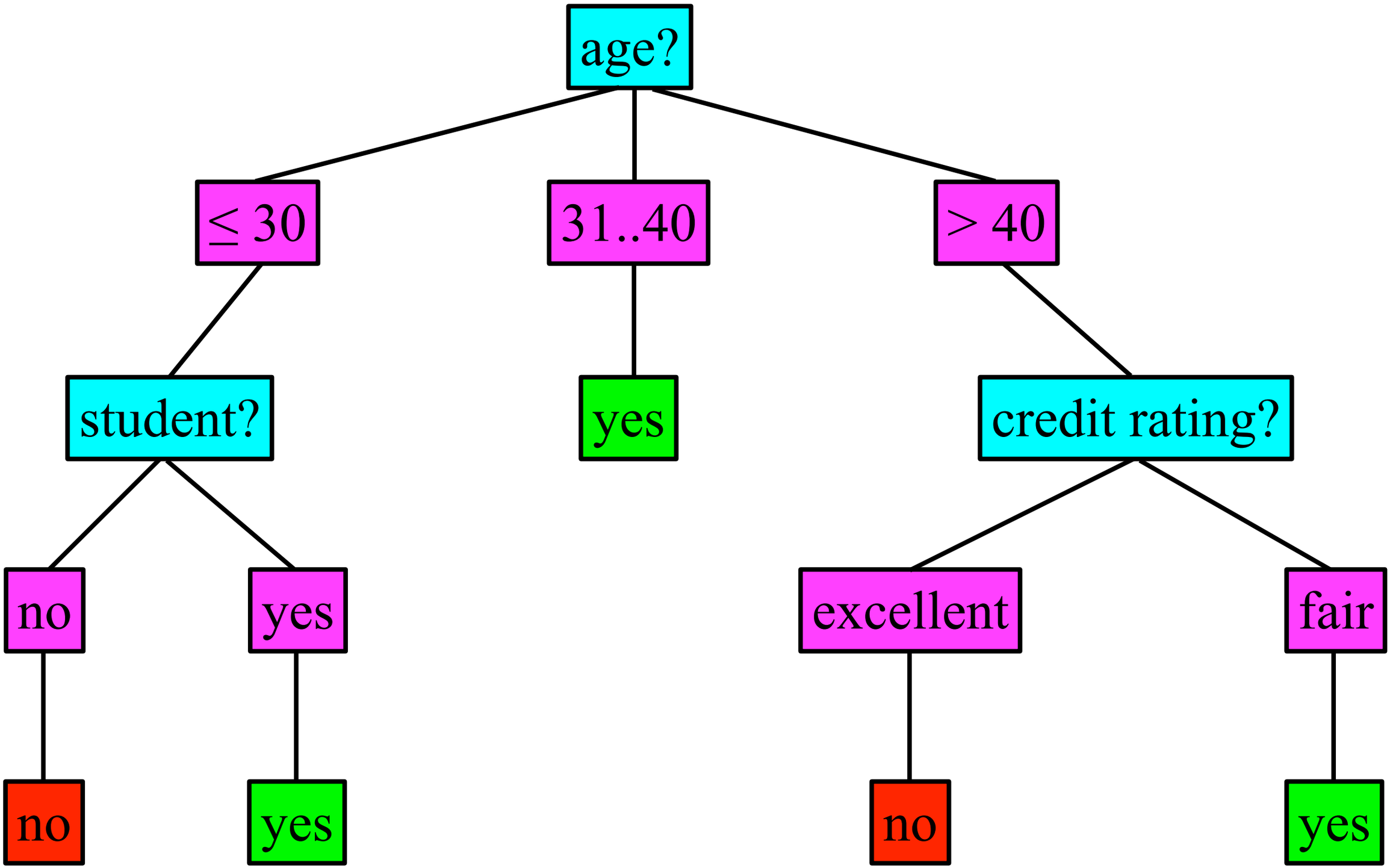
Basic idea

- We define the label by asking series of questions about the attributes
  - Each question depends on the answer to the previous one
  - Ultimately, all samples with satisfying attribute values have the same label and we're done
- The flow-chart of the questions can be drawn as a tree
- We can classify new instances by following the proper edges of the tree until we meet a leaf
  - Decision tree leafs are always class labels

Example: training data

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Example: decision tree



Hunt's algorithm

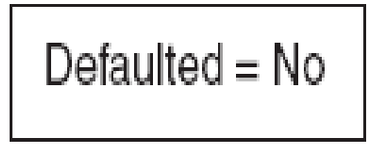
- The number of decision trees for a given set of attributes is exponential
- Finding the the most accurate tree is NP-hard
- Practical algorithms use *greedy heuristics*
  - The decision tree is grown by making a series of locally optimum decisions on which attributes to use
- Most algorithms are based on Hunt's algorithm

Hunt's algorithm

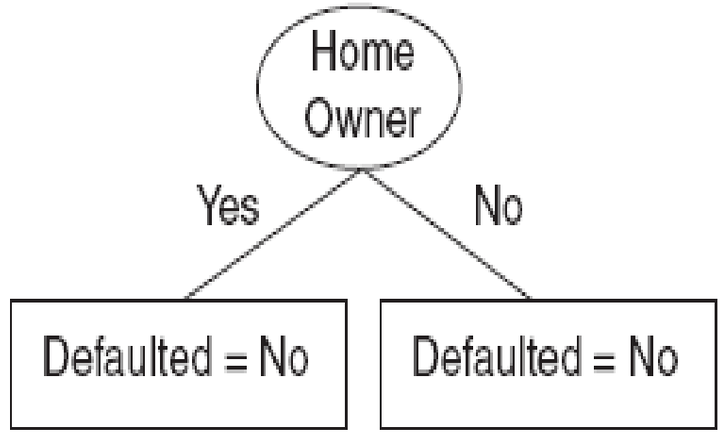
- Let X_t be the set of training records for node t
- Let $y = \{y_1, \ldots y_c\}$ be the class labels
- **Step 1**: If all records in X_t belong to the same class y_t, then t is a leaf node labeled as y_t
- **Step 2:** If X_t contains records that belong to more than one class
  - Select *attribute test condition* to partition the records into smaller subsets
  - Create a *child node* for each outcome of test condition
  - Apply algorithm recursively to each child

Example decision tree construction

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

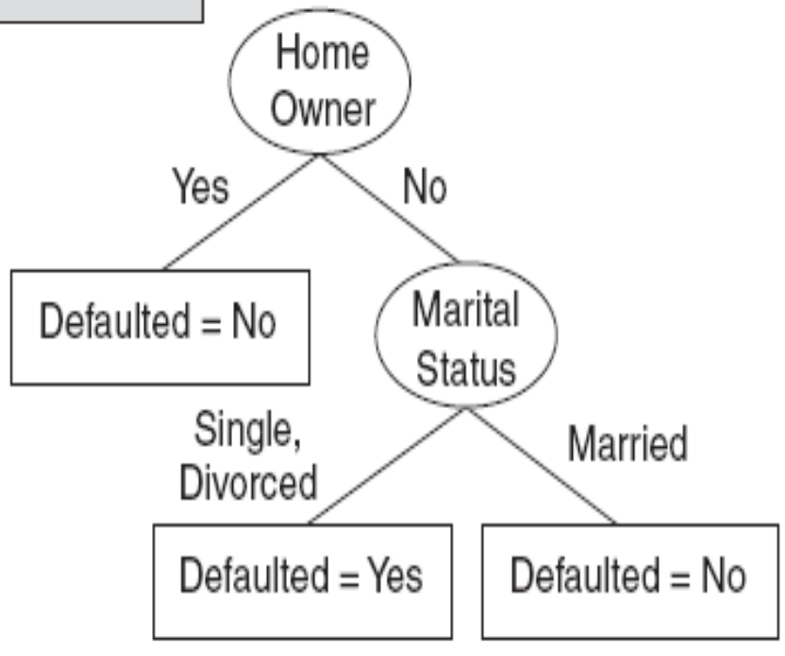


Has multiple labels



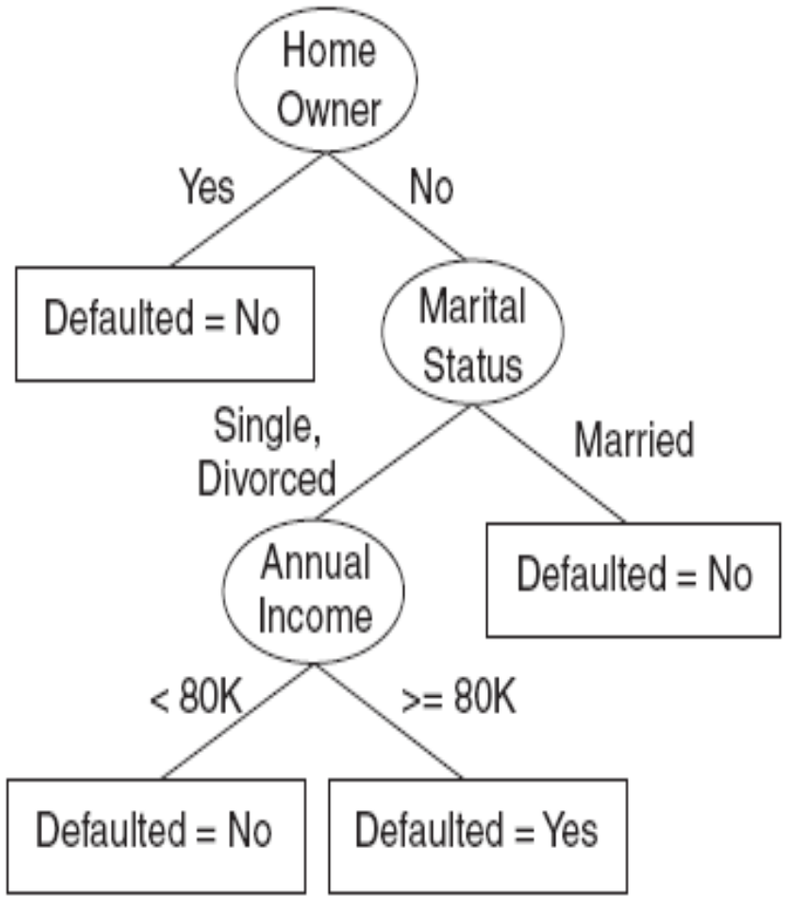
Only one label

Has multiple labels



Has multiple labels

Only one label



Only one label

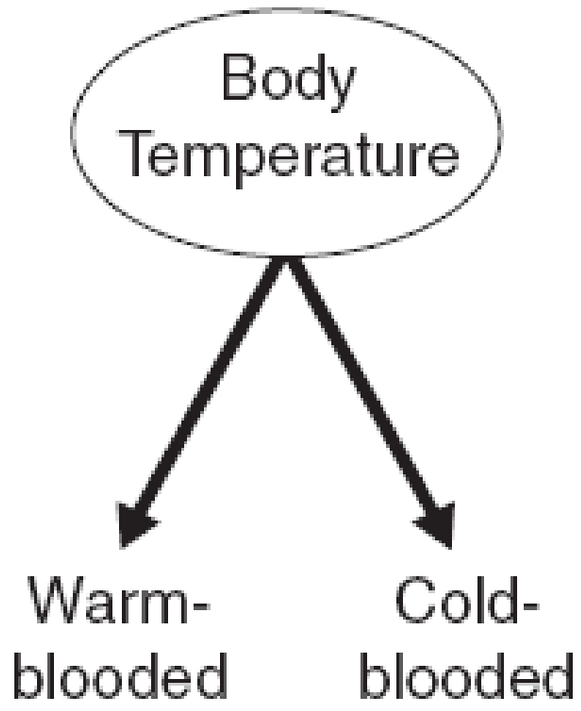
Only one label

Selecting the split

- Designing a decision-tree algorithm requires answering two questions
  1. How should the training records be split?
  2. How should the splitting procedure stop?

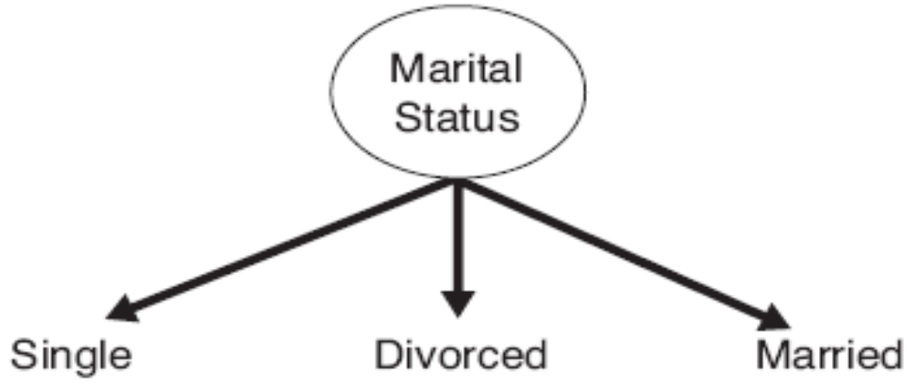
Splitting methods

Binary attributes

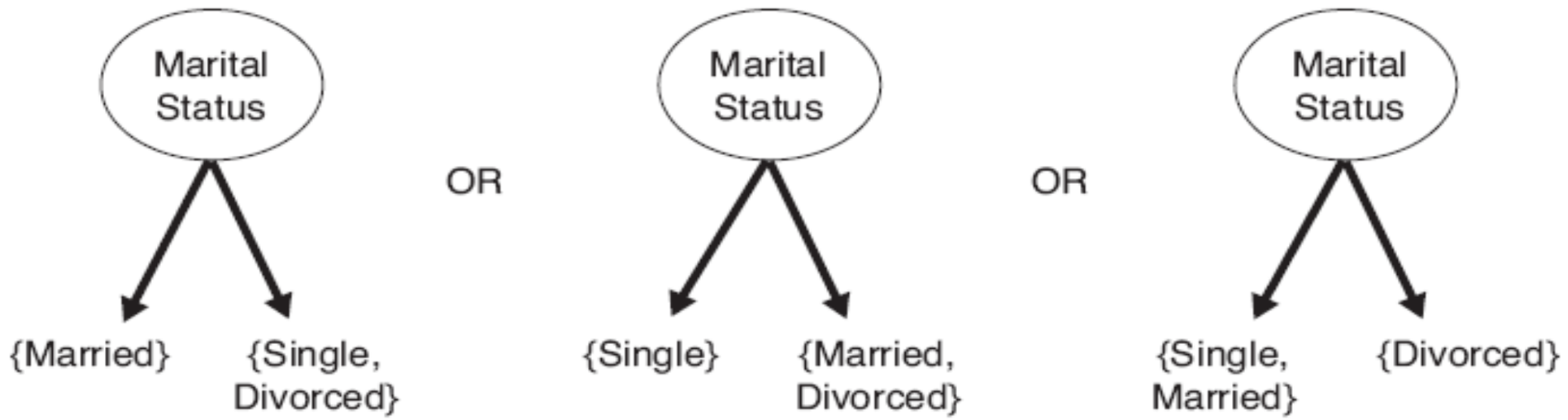


Splitting methods

Nominal attributes



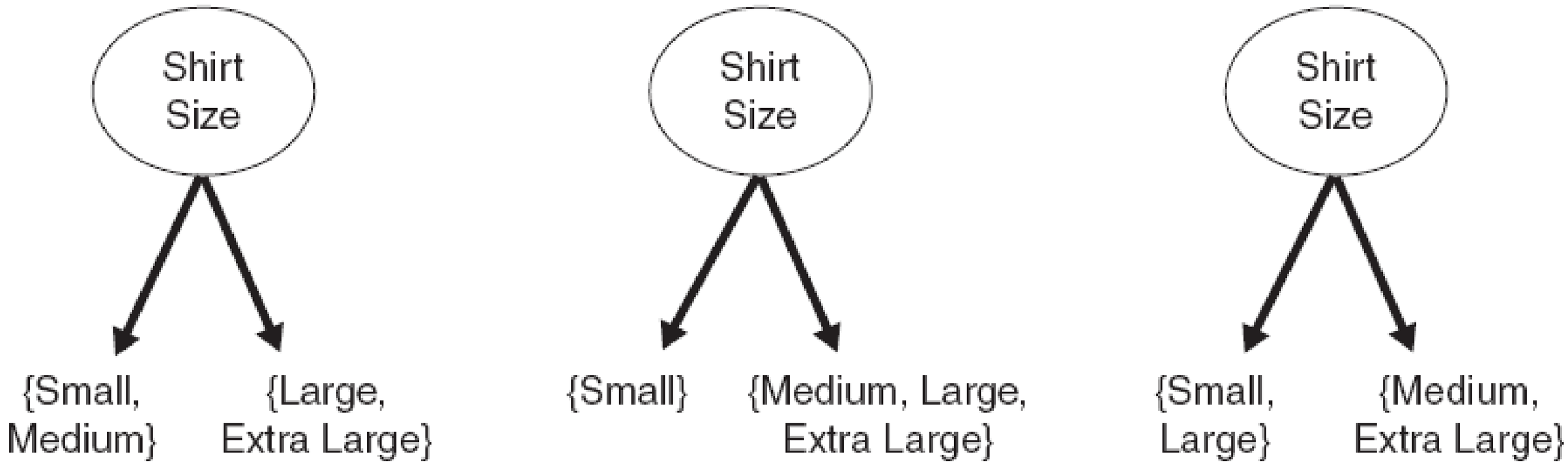
Multiway split



Binary split

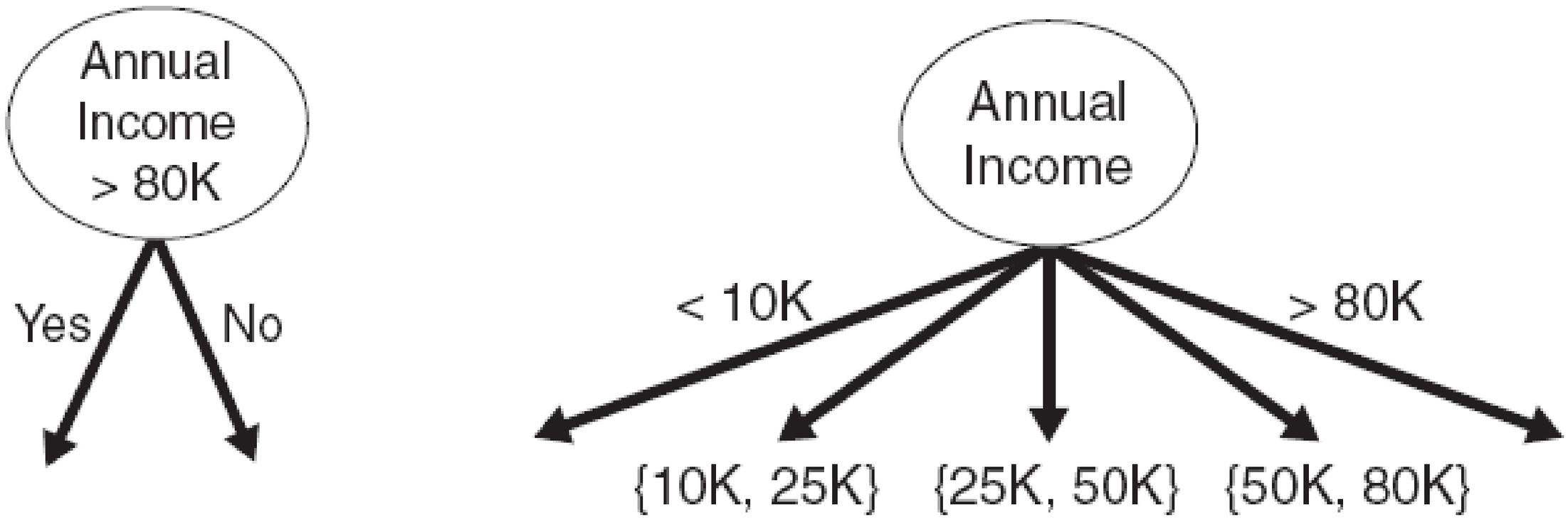
Splitting methods

Ordinal attributes



Splitting methods

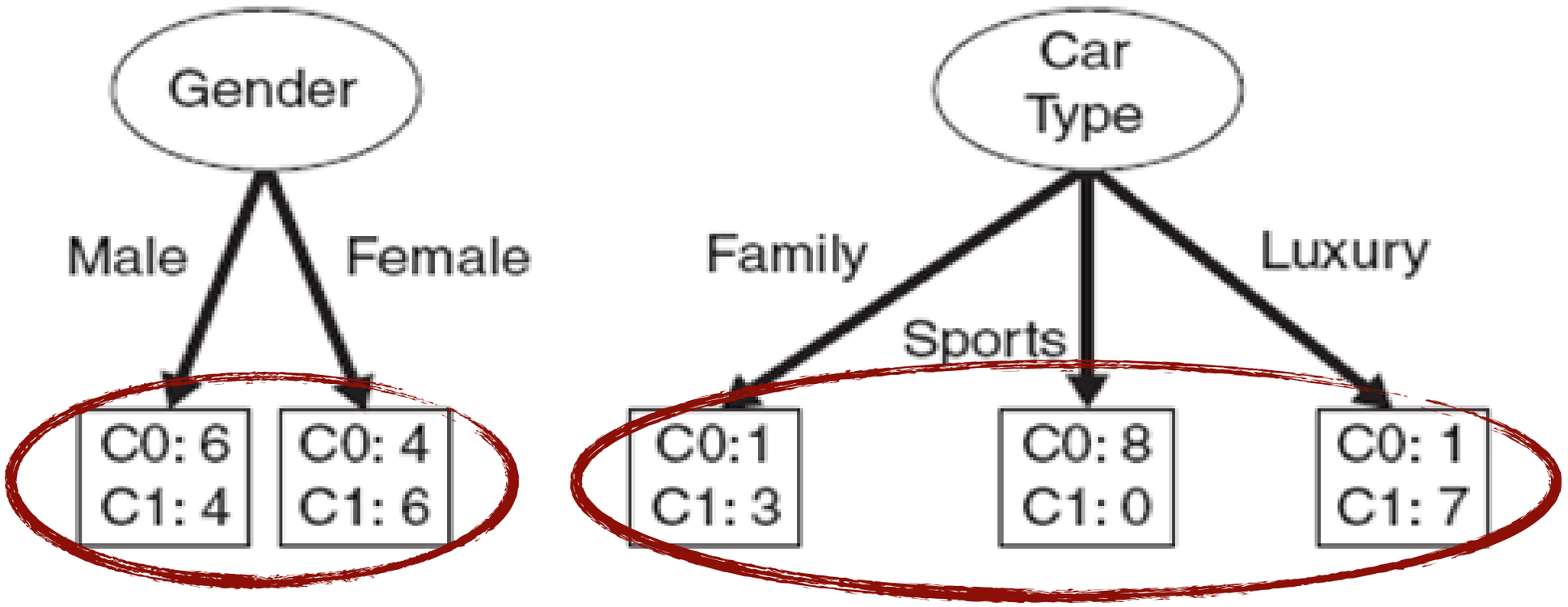
Continuous attributes



Selecting the best split

- Let $p(i \mid t)$ be the fraction of records belonging to class i at node t
- *Best split* is selected based on the degree of **impurity** of the child nodes
  - $p(0 \mid t) = 0$ and $p(1 \mid t) = 1$ has *high purity*
  - $p(0 \mid t) = 1/2$ and $p(1 \mid t) = 1/2$ has the *smallest purity* (*highest impurity*)
- Intuition: high purity $\Rightarrow$ small value of impurity measures $\Rightarrow$ better split

Example of purity



high impurity

high purity

Impurity measures

$$\text{Entropy}(t) = -\sum_{i=0}^{c-1} p(i \mid t) \log_2 p(i \mid t)$$

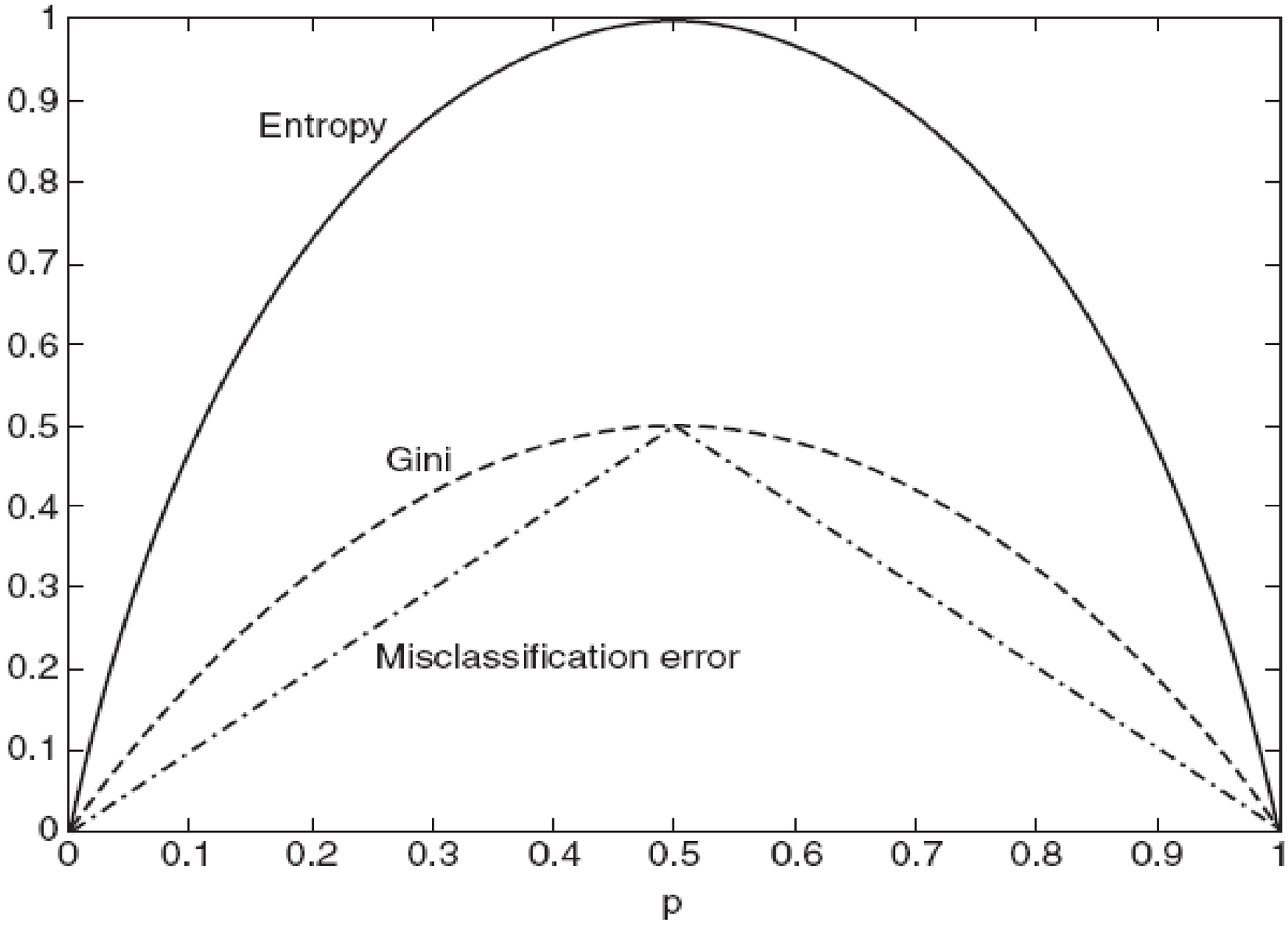
$0 \times \log_2(0) = 0$

≤ 0

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} \big(p(i \mid t)\big)^2$$

$$\text{Classification error}(t) = 1 - \max_i \{p(i \mid t)\}$$

Comparing impurity measures



Comparing conditions

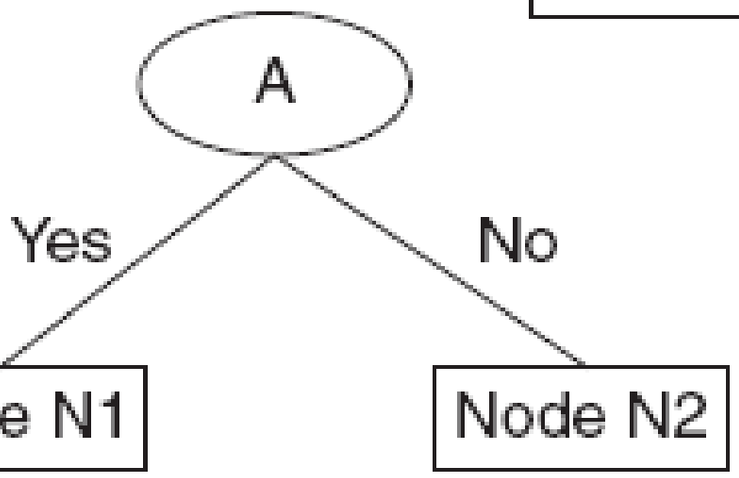
- The quality of the split: the change in the impurity
  - Called the **gain** of the test condition

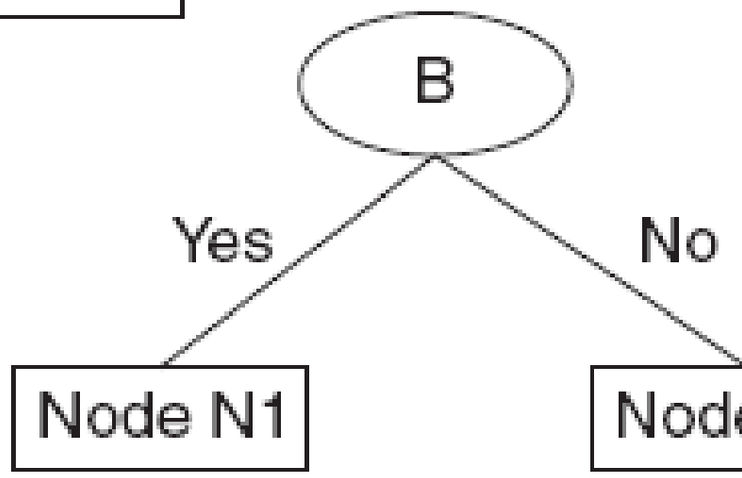
$$\Delta = \mathrm{I}(\mathrm{p}) - \sum_{\mathrm{j}=1}^{\mathrm{k}} \frac{\mathrm{N}(v_{\mathrm{j}})}{\mathrm{N}} \mathrm{I}(v_{\mathrm{j}})$$

    - $I(\ )$ is the impurity measure
    - k is the number of attribute values
    - p is the parent node, v_j is the child node
    - N is the total number of records at the parent node
    - $N(v_j)$ is the number of records associated with the child node
- Maximizing the gain ⇔ minimizing the weighted average impurity measure of child nodes
- If $I()$ = Entropy(), then $\Delta = \Delta_{\text{info}}$ is called **information gain**

Computing the gain: example

	Parent
C0	6
C1	6
Gini = 0.500	





G: 0.4898

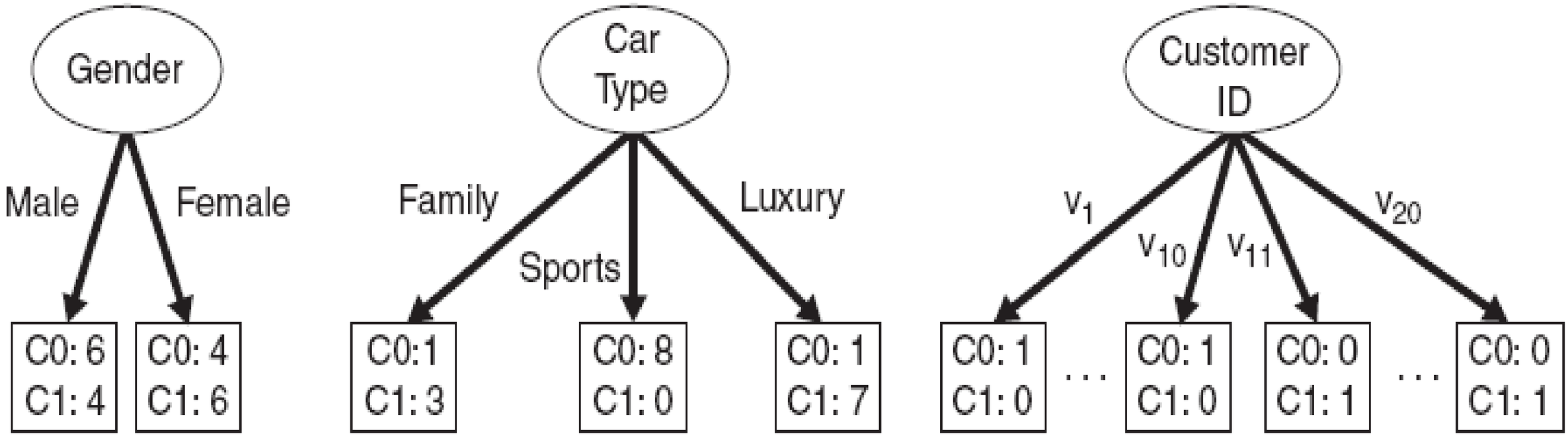
G: 0.480

	N1	N2
C0	4	2
C1	3	3
Gini = 0.486		

	N1	N2
C0	1	5
C1	4	2
Gini = 0.375		

$(7 \times 0.4898 + 5 \times 0.480) / 12 = 0.486$

Problems of maximizing Δ



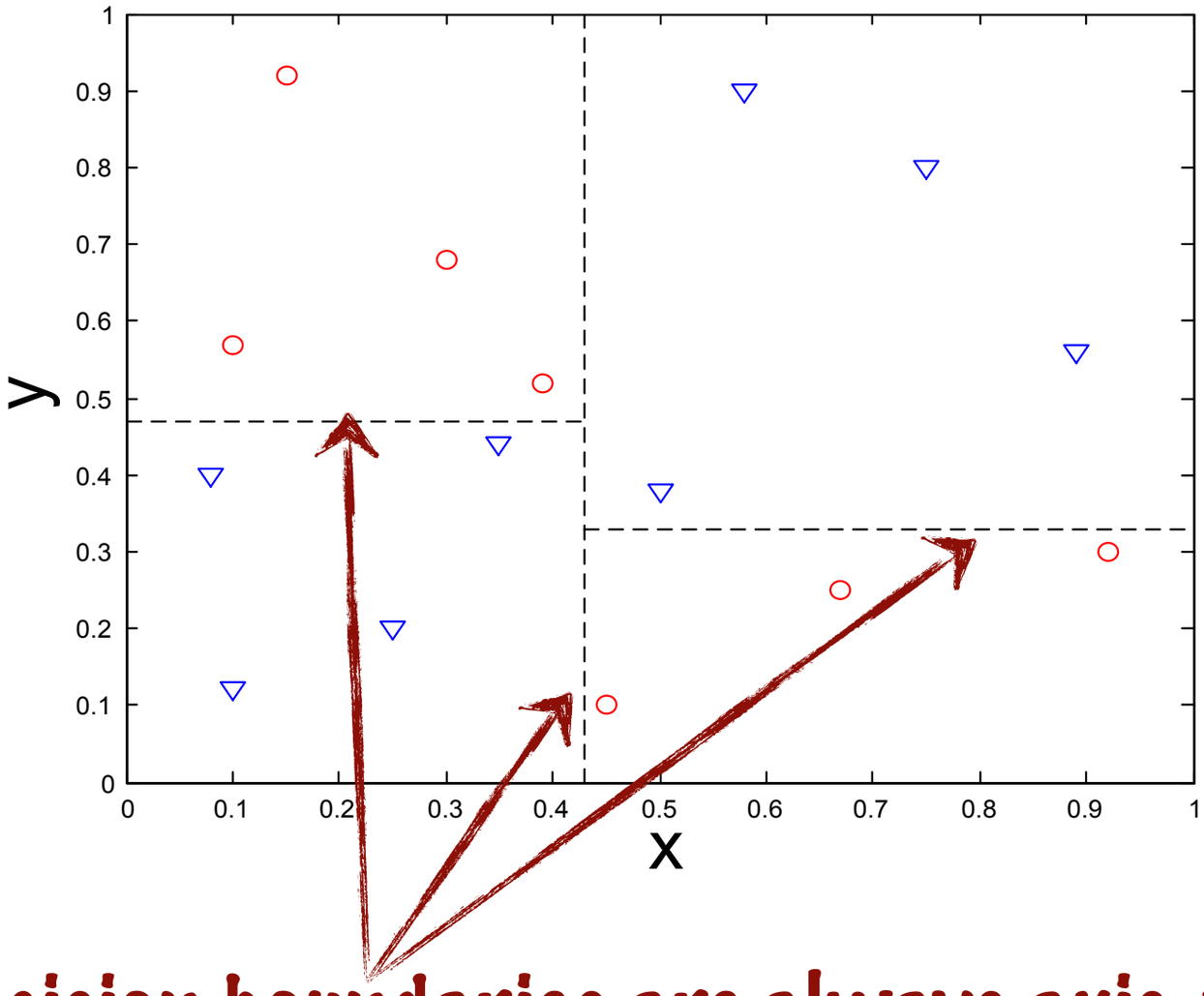
Problems of maximizing Δ

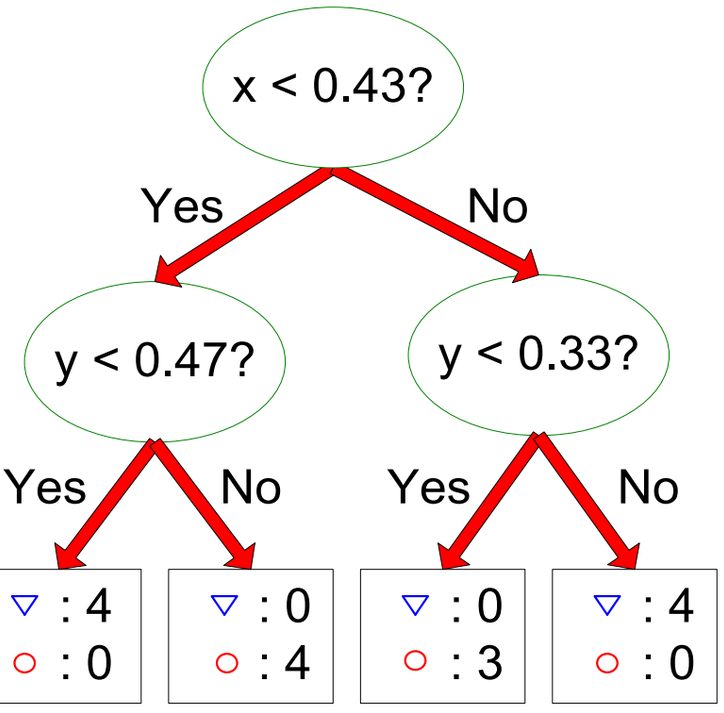
- Impurity measures favor attributes with large number of values
- A test condition with large number of outcomes might not be desirable
  - Number of records in each partition is too small to make predictions
- Solution 1: **gain ratio** = Δ_{info} / SplitInfo
  - $\text{SplitInfo} = -\sum_{i=1}^{k} P(v_i) \log_2(P(v_i))$
    - $P(v_i)$ = the fraction of records at child; k = total number of splits
  - Used e.g. in C4.5
- Solution 2: restrict the splits to binary

Stopping the splitting

- Stop expanding when all records belong to the same class
- Stop expanding when all records have similar attribute values
- Early termination
  - E.g. gain ratio drops below certain threshold
  - Keeps trees simple
  - Helps with overfitting

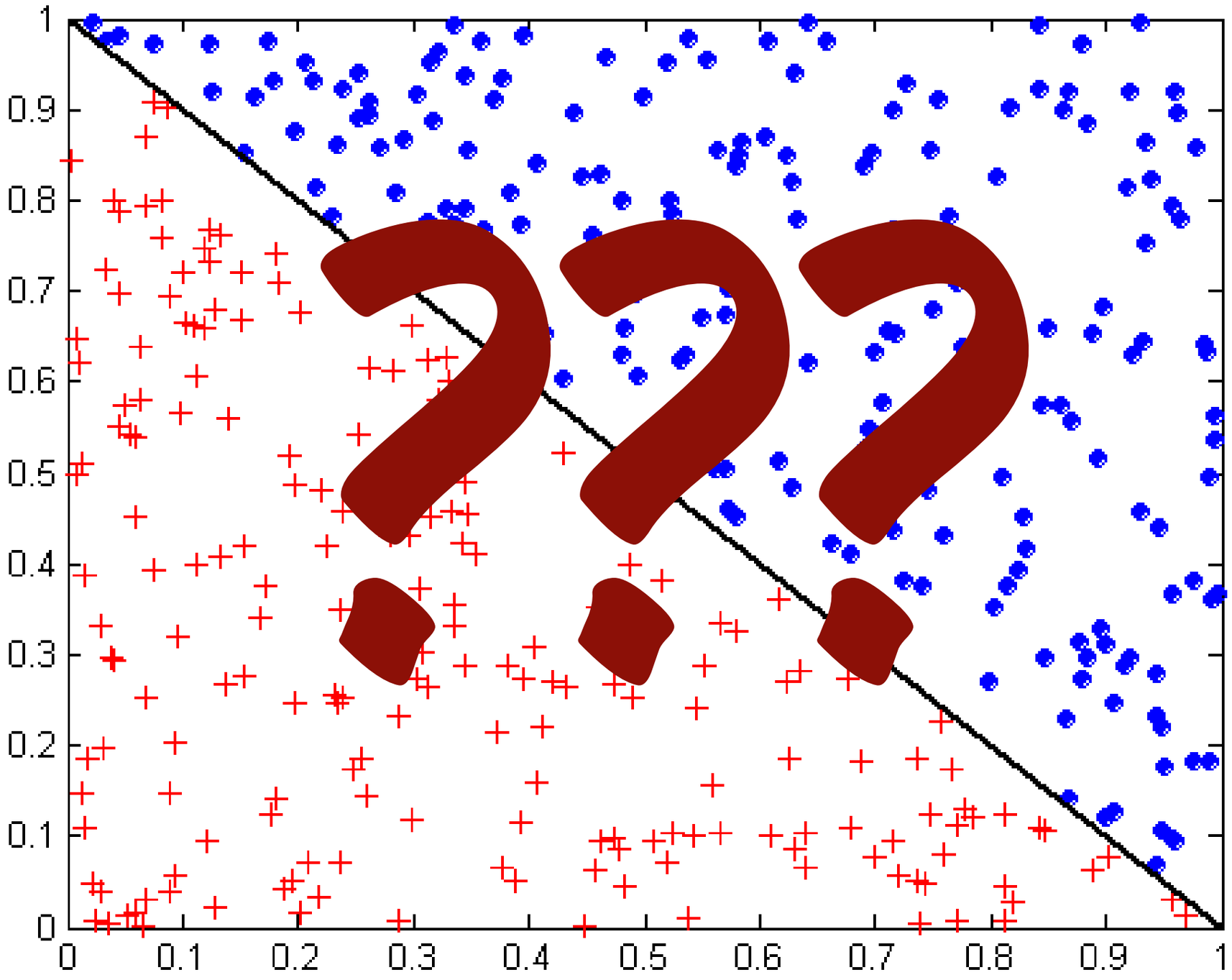
Geometry of single-attribute splits





Decision boundaries are always axis-parallel for single-attribute splits

Geometry of single-attribute splits



Summary of decision trees

- Fast to build
- Extremely fast to use
  - Small ones are easy to interpret
    - Good for domain expert's verification
    - Used e.g. in medicine
- Redundant attributes are not (much of) a problem
- Single-attribute splits cause axis-parallel decision boundaries
- Requires post-pruning to avoid overfitting

IX.3 Naïve Bayes classifier

1. Basic idea
2. Computing the probabilities
3. Summary

Zaki & Meira, Ch. 18; Tan, Steinbach & Kumar, Ch. 5.3

Basic idea

- Recall the Bayes' theorem

$$\Pr[Y \mid X] = \frac{\Pr[X \mid Y]\,\Pr[Y]}{\Pr[X]}$$

- In classification
  - RV X = attribute set
  - RV Y = class variable
  - Y depends on X in a *non-deterministic* way
- The dependency between X and Y is captured in $\Pr[Y \mid X]$ and $\Pr[Y]$
  - Posterior and prior probability

Building the classifier

- **Training phase**
  - Learn the posterior probabilities Pr[$Y \mid X$] for every combination of X and Y based on training data
- **Test phase**
  - For test record X', compute the class Y' that *maximizes the posterior probability* Pr[$Y' \mid X'$]
    - $Y' = \arg\max_i\{\Pr[c_i \mid X']\} = \arg\max_i\{\Pr[X' \mid c_i]\Pr[c_i]/\Pr[X']\}$
      $= \arg\max_i\{\Pr[X' \mid c_i]\Pr[c_i]\}$
- So we need Pr[$X' \mid c_i$] and Pr[c_i]
  - Pr[c_i] is the fraction of test records that belong to class c_i
  - Pr[$X' \mid c_i$]?

Computing the probabilities

- Assume that the attributes are conditionally independent given the class label
  - Naïvety of the classifier
  - $\Pr[X \mid Y = c_i] = \prod_{i=1}^{d} \Pr[X_i \mid Y = c_i]$
    - X_i is the attribute i
- Without independency there would be too many variables to estimate
- With independency, it is enough to estimate $\Pr[X_i \mid Y]$
  - $\Pr[Y \mid X] = \Pr[Y] \prod_{i=1}^{d} \Pr[X_i \mid Y] / \Pr[X]$
  - $\Pr[X]$ is fixed, so can be omitted
- But how to estimate the *likelihood* $\Pr[X_i \mid Y]$?

Categorical attributes

- If X_i is categorical $\Pr[X_i = x_i \mid Y = c]$ is the fraction of training instances in class c that take value x_i on the i-th attribute

Pr[HomeOwner = yes | No] = 3/7
Pr[MartialStatus = S | Yes] = 2/3

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Continuous attributes: discretization

- We can discretize continuous attributes to intervals
  - These intervals act like ordinal attributes
- Problem is where to discretize
  - Too many intervals: too few training records per interval
    ⇒ unreliable estimates
  - Too few intervals: intervals merge attributes from different classes and don't help distinguishing the classes

Continuous attributes continue

- Alternatively we can assume distribution for the continuous variables
  - Normally we assume normal distribution
- We need to estimate the distribution parameters
  - For normal distribution we can use sample mean and sample variance
  - For estimation we consider the values of attribute X_i that are associated with class c_j in the test data
- We hope that the parameters for distributions are different for different classes of the same attribute
  - Why?

Naïve Bayes example

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Annual Income:
Class = No
  Sample mean = 110
  Sample variance = 2975
Class = Yes
  Sample mean = 90
  Sample variance = 25

Test data: X = (HO = No, MS = M, AI = $120K)

Pr[Yes] = 0.3, Pr[No] = 0.7

$$\Pr[X \mid No] = \Pr[HO = No \mid No] \times \Pr[MS = M \mid No] \times \Pr[AI = \$120K \mid No]$$
$$= 4/7 \times 4/7 \times 0.0072 = 0.0024$$

$$\Pr[X \mid Yes] = \Pr[HO = No \mid Yes] \times \Pr[MS = M \mid Yes] \times \Pr[AI = \$120K \mid Yes]$$
$$= 1 \times 0 \times \varepsilon = 0$$

$$\Pr[No \mid X] = \alpha \times 0.7 \times 0.0024 = 0.0016\alpha, \quad \alpha = 1/\Pr[X]$$

$\Rightarrow$ Pr[No | X] has higher posterior and X should be classified as non-defaulter

Continuous distributions at fixed point

- If X_i is continuous, $\Pr[X_i = x_i \mid Y = c_i] = 0$!
  - But we still need to estimate that number
- Self-cancelling trick:

$$\Pr[x_i - \epsilon \leqslant X_i \leqslant x_i + \epsilon \mid Y = c_j] = \int_{x_i-\epsilon}^{x_i+\epsilon} (2\pi\sigma_{ij})^{-\frac{1}{2}} \exp\left(-\frac{(x-\mu_{ij})^2}{2\sigma_{ij}^2}\right)$$

$$\approx 2\epsilon\, f(x_i; \mu_{ij}, \sigma_{ij})$$

  - But 2ε cancels out in the normalization constant…

Zero likelihood

- We might have no samples with $X_i = x_i$ and $Y = c_j$
  - Naturally only problem with categorical variables
  - $\Pr[X_i = x_i \mid Y = c_j] = 0 \Rightarrow$ zero posterior probability
  - It can be that *all* classes have zero posterior probability for some validation data
- Answer is smoothing (*m-estimate*):
  - $\Pr[X_i = x_i \mid Y = c_j] = \frac{n_i + mp}{n + m}$
    - n = # of training instances from class c_j
    - n_i = # training instances from c_j that take value x_i
    - m = "equivalent sample size"
    - p = user-set parameter

More on m-estimate

$$\Pr[X_i = x_i \mid Y = c_j] = \frac{n_i + mp}{n + m}$$

- The parameters are p and m
  - If $n = 0$, then likelihood is p
    - p is "prior" of observing x_i in class c_j
  - Parameter m governs the trade-off between p and observed probability n_i/n
- Setting these parameters is again problematic…
- Alternatively, we can just add one *pseudo-count* to each class
  - $\Pr[X_i = x_i \mid Y = c_j] = (n_j + 1) / (n + |\text{dom}(X_i)|)$
    - $|\text{dom}(X_i)|$ = # values attribute X_i can take

Summary of naïve Bayes

- Robust to isolated noise
  - Averaged out
- Can handle missing values
  - Example is ignored when building the model and attribute is ignored when classifying new data
- Robust to irrelevant attributes
  - $\Pr(X_i \mid Y)$ is (almost) uniform for irrelevant X_i
- Can have issues with correlated attributes