Machine Learning & Pattern Recognition

SONG Xuemeng

sxmustc@gmail.com

http://xuemeng.bitcron.com/

- Suppose you are given a binary classification task: whether a customer is likely to buy a computer or not.

- Suppose you are given a binary classification task: whether a customer is likely to buy a computer or not.
- One approach is to pose a series of questions about the characteristics of the customer.
  - How old is the customer (young, middle or senior)?
  - If he is of middle age, then he is likely to buy.
  - Else if he is young, you are not sure and may need to ask a follow-up question: is the customer a undergraduate?
    - If so, then he is likely to buy.
- This illustrates how we can solve a classification problem by asking a series of carefully crafted questions about the attributes of the test record.
- Each time we receive an answer, a follow-up question is asked until we reach a conclusion (the class label).

Decision Tree

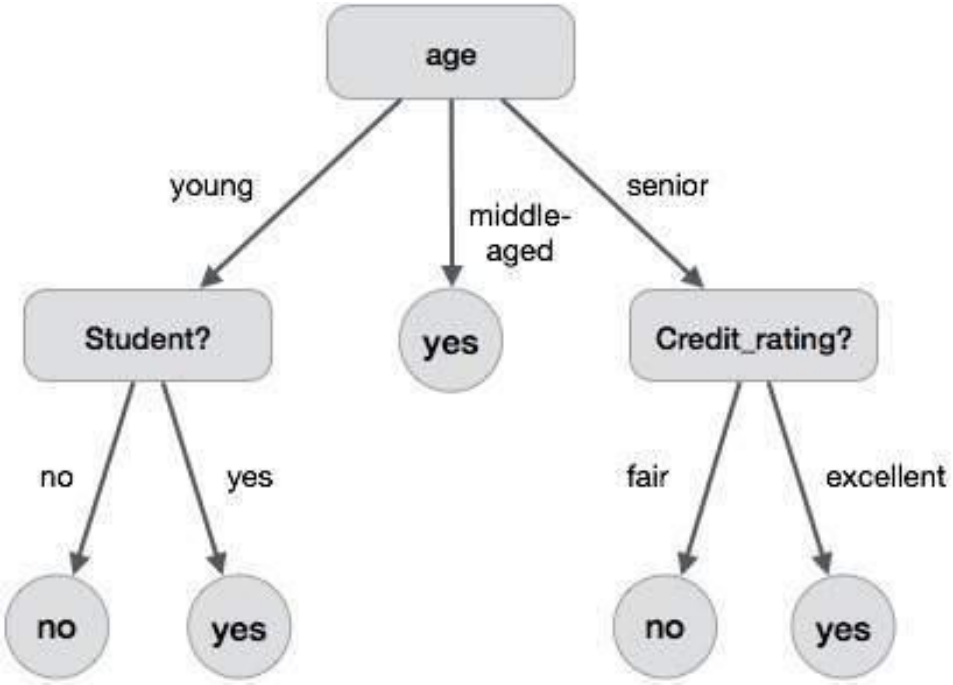
Decision Tree

- Introduction of Decision Trees
  - Attribute Selection
  - Deal with Numeric Attributes
  - Deal with Missing Values
  - Pruning to Avoid Overfitting
- Brief Introduction of Regression Trees

Decision Tree

A decision tree consists of **nodes, edges and leaves.**

- Nodes:
  - Test for the value of a certain attribute.
  - The top most node in the tree is the root node.

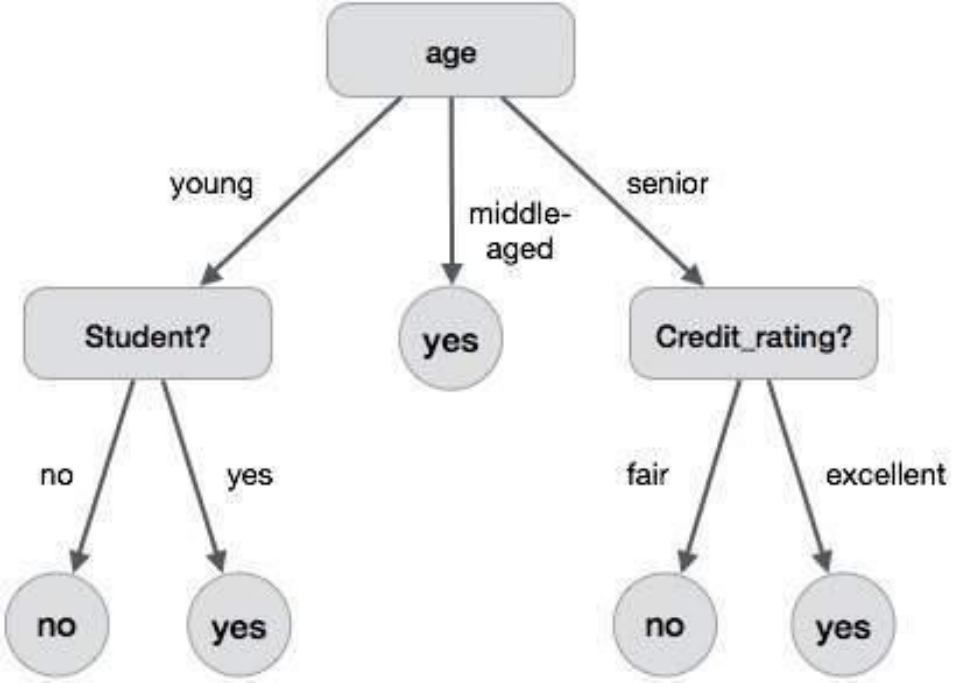


The above decision tree is to determine whether a customer is likely to buy a computer or not.

Decision Tree

A decision tree consists of **nodes, edges and leaves.**

- Edges (Branches)
  - Correspond to the outcome of a test.
  - Connect to the next node or leaf.

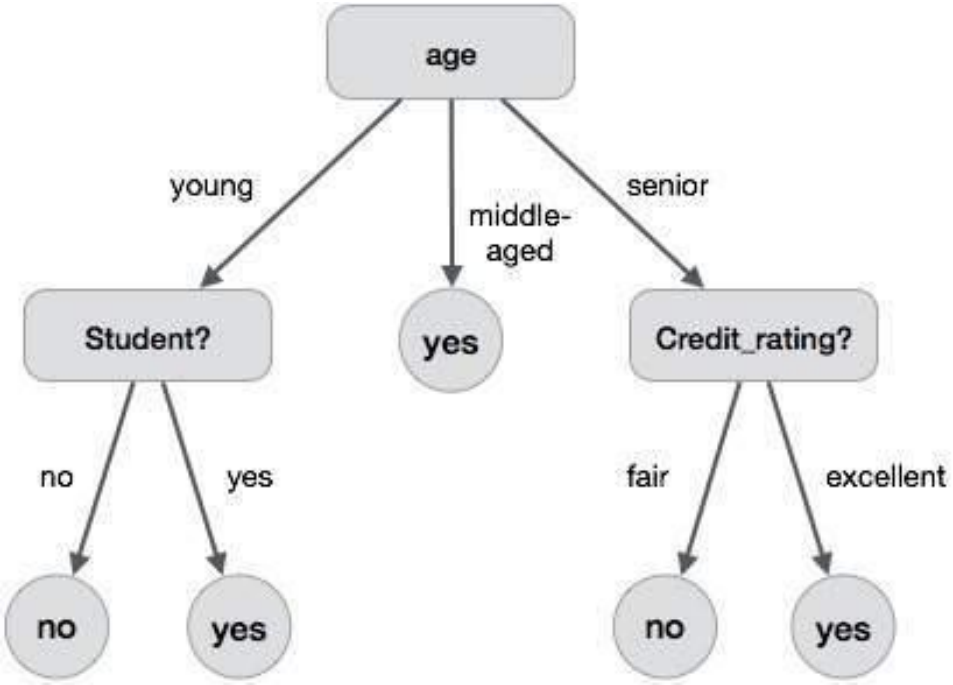


The above decision tree is to determine whether a customer is likely to buy a computer or not.

Decision Tree

A decision tree consists of **nodes, edges and leaves.**

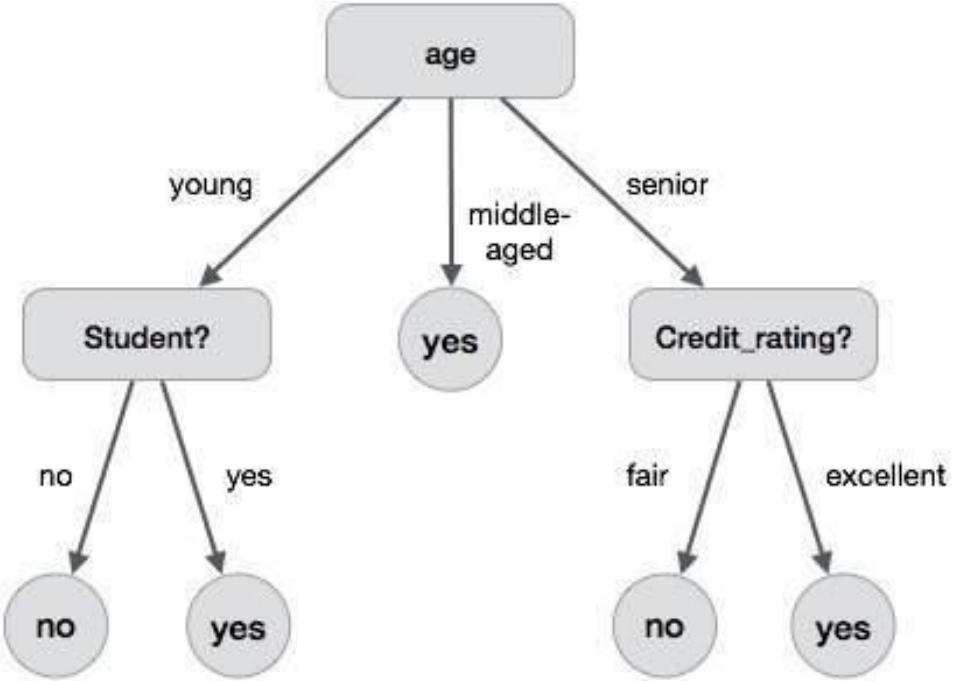
- Leaves
  - Terminal nodes that predict the outcome (class labels).



The above decision tree is to determine whether a customer is likely to buy a computer or not.

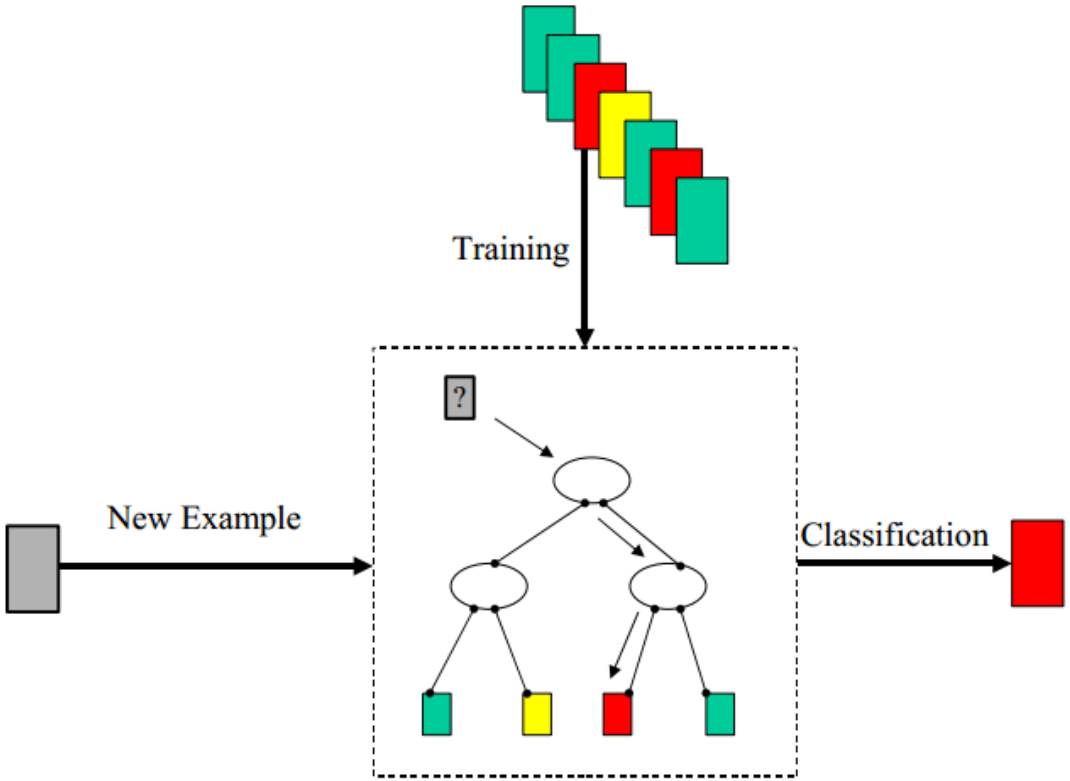
Decision Tree

- **To classify an instance**
  1. Start at the root
  2. Perform the test
  3. Follow the edge corresponding to outcome
  4. Go to 2 unless leaf
  5. Predict that outcome associated with the leaf



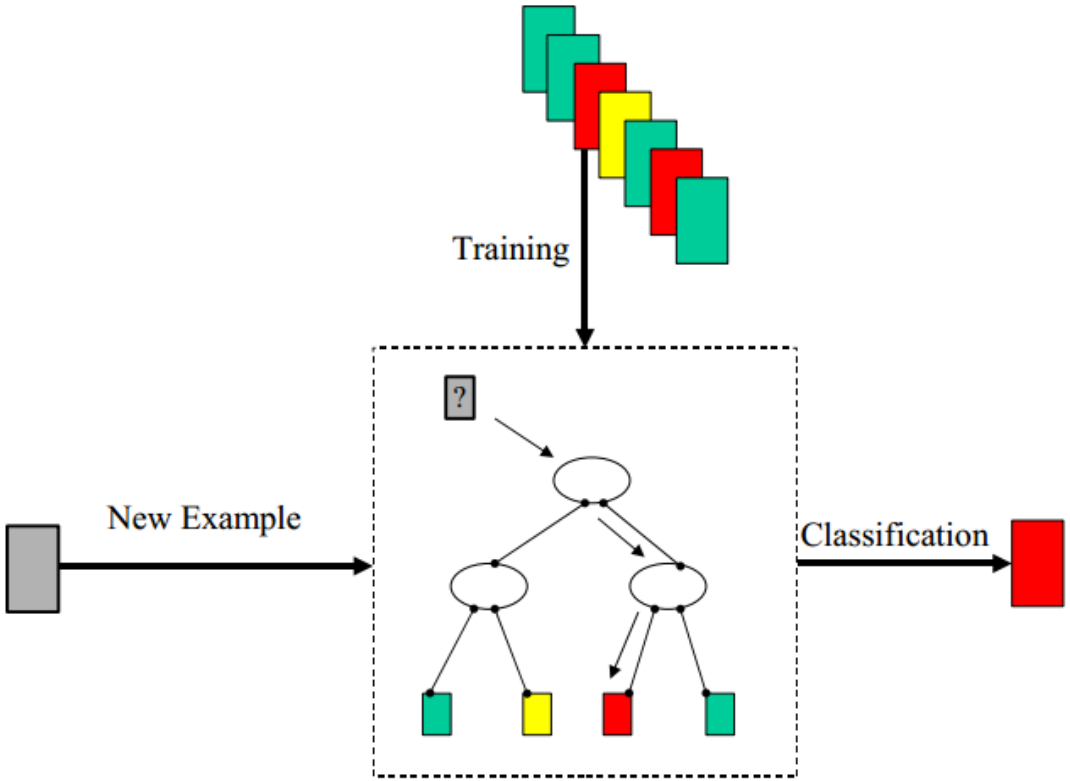
Decision Tree

- A new example is classified by submitting it to a series of tests that determine the class label of the example.
- These tests are organized in a hierarchical structure, called a *decision tree*.



Decision Tree

- The training examples are used for choosing appropriate tests in the decision tree.
- Typically, a tree is built from top to bottom, where tests that maximize the information gain for classification are selected first.

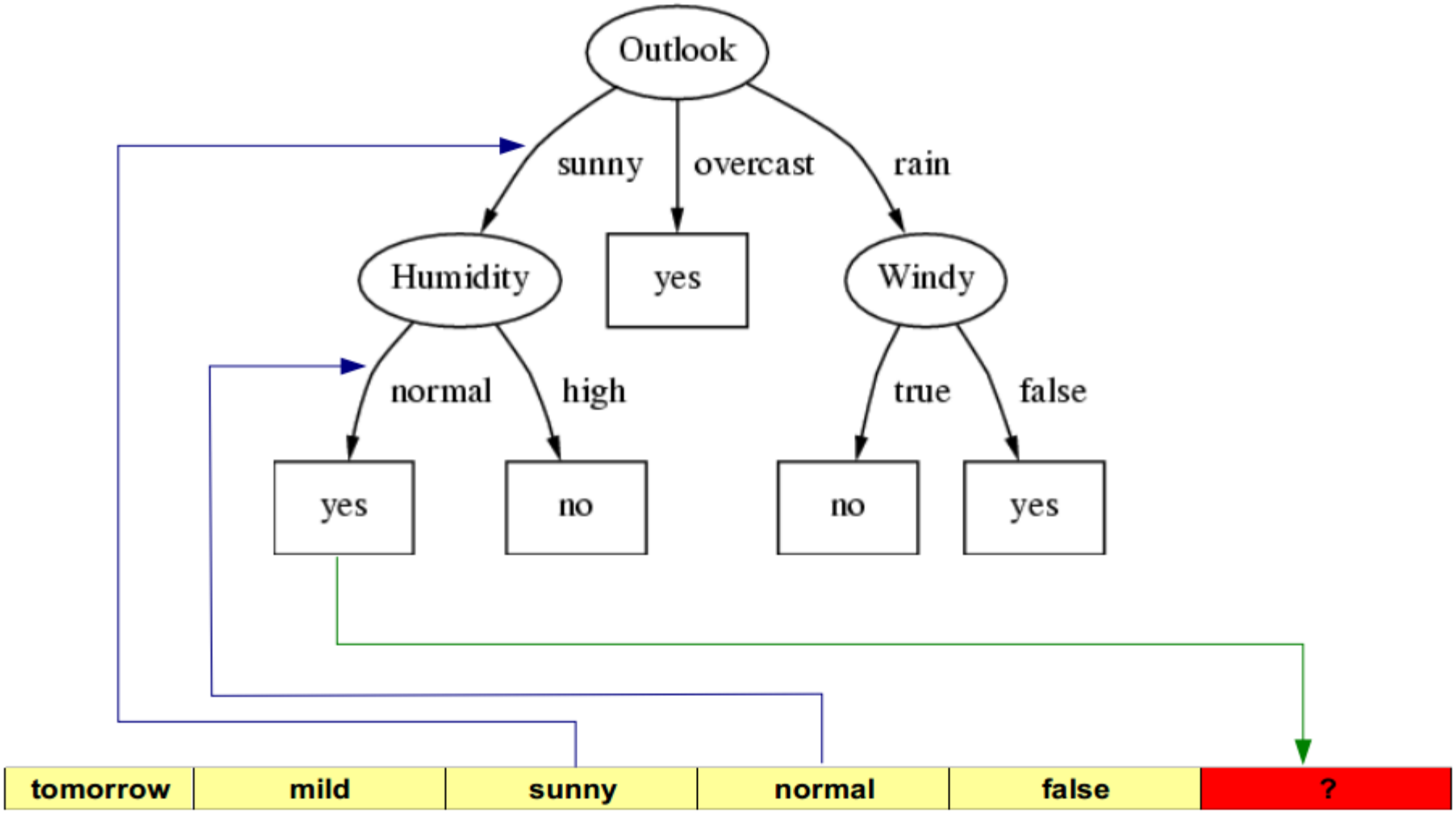


A Sample Task

Day	Temperature	Outlook	Humidity	Windy	Play Golf?
07-05	hot	sunny	high	false	no
07-06	hot	sunny	high	true	no
07-07	hot	overcast	high	false	yes
07-09	cool	rain	normal	false	yes
07-10	cool	overcast	normal	true	yes
07-12	mild	sunny	high	false	no
07-14	cool	sunny	normal	false	yes
07-15	mild	rain	normal	false	yes
07-20	mild	sunny	normal	true	yes
07-21	mild	overcast	high	true	yes
07-22	hot	overcast	normal	false	yes
07-23	mild	rain	high	true	no
07-26	cool	rain	normal	true	no
07-30	mild	rain	high	false	yes

today	cool	sunny	normal	false	?
tomorrow	mild	sunny	normal	false	?

Decision Tree Learning



Divide-and-Conquer Algorithms

- Family of decision tree learning algorithms
  - TDIDT: Top-Down Induction of Decision Trees
    - Divide the problem in sub-problems
    - Solve each problem

Basic Divide-And-Conquer Algorithm:

1. select a test for root node
   - ➢ Create branch for each possible outcome of the test
2. split instances into subsets
   - ➢ One for each branch extending from the node
3. repeat recursively for each branch, using only instances that reach the branch
4. stop recursion for a branch if all its instances have the same class

Real Implementations

- **Ross Quinlan**
  - **ID3, 1986**
    - Available in WEKA (no discretization, no missing values)
  - **C4.5, 1993**
    - Handles missing attributes and continuous attributes
    - Performs tree post-pruning (Pessimistic error pruning)
    - Available in WEKA as J48
  - **C5.0**
    - Commercial successor of C4.5

- **Breiman et al.**
  - **CART (Classification and Regression Trees), 1984**
    - Binary trees
    - Able to generate regression trees.

John Ross Quinlan

- BSc degree in Physics and Computing from the University of Sydney in 1965
- Computer science doctorate at the University of Washington in 1968.

http://www.rulequest.com/Personal/

Top 10 Algorithms (2007)

Top 10 algorithms in data mining

Xindong Wu · Vipin Kumar · J. Ross Quinlan · Joydeep Ghosh · Qiang Yang · Hiroshi Motoda · Geoffrey J. McLachlan · Angus Ng · Bing Liu · Philip S. Yu · Zhi-Hua Zhou · Michael Steinbach · David J. Hand · Dan Steinberg

Received: 9 July 2007 / Revised: 28 September 2007 / Accepted: 8 October 2007
Published online: 4 December 2007

Abstract This paper presents the top 10 data mining algorithms identified by the IEEE International Conference on Data Mining (ICDM) in December 2006: C4.5, *k*-Means, SVM, Apriori, EM, PageRank, AdaBoost, *k*NN, Naive Bayes, and CART. These top 10 algorithms are among the most influential data mining algorithms in the research community. With each algorithm, we provide a description of the algorithm, discuss the impact of the algorithm, and review current and further research on the algorithm. These 10 algorithms cover classification,

ID3 Algorithm

- **Function ID3**
  - **Input**: example set S
  - **Output**: decision tree DT

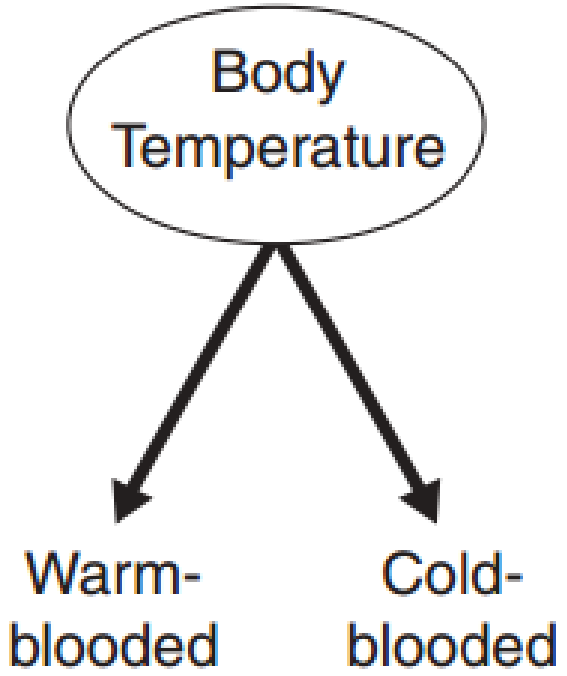
❑ If all examples in S belong to the same class c
  - Return a new leaf and label it with c

❑ Else
  - Select an attribute A according to *some heuristic function*
  - Generate a new node DT with A as test
  - For each value v_i of A
    - ➢ Let S_i =all examples in S with $A = v_i$
    - ➢ Use ID3 to construct a decision tree for example set S_i
    - ➢ Generate an edge that connects DT and DT_i

Attribute (test) Selection Criteria

Methods for Expressing Attribute Test

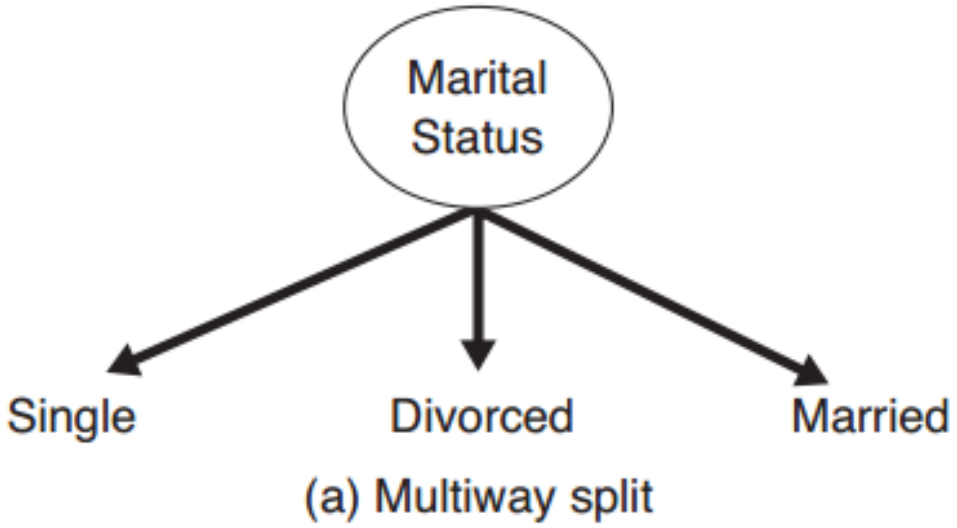
- **Binary Attributes**



Test condition for binary attributes.

Methods for Expressing Attribute Test

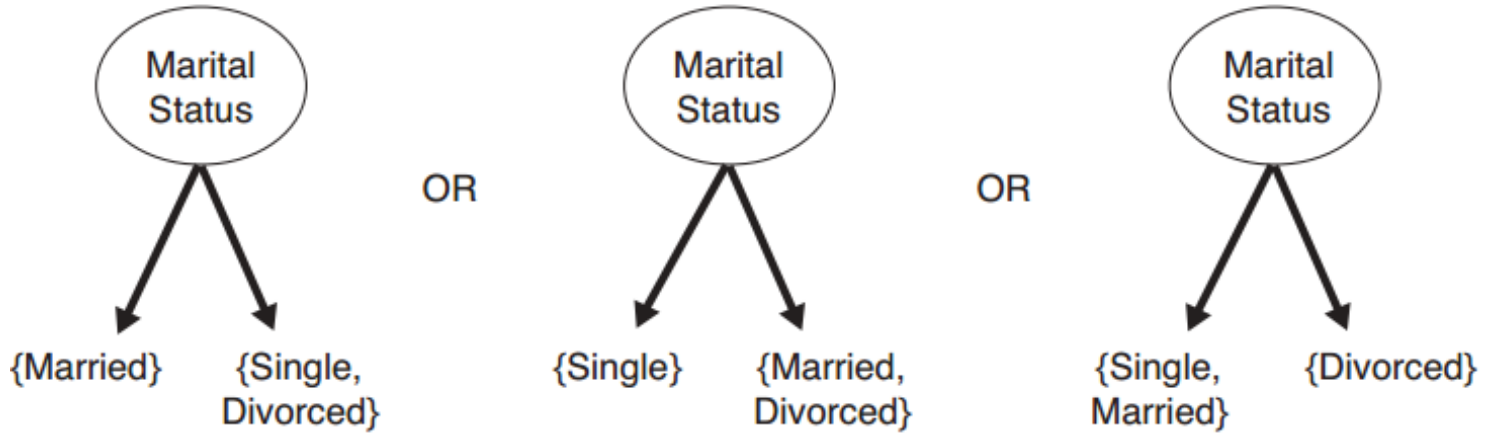
- **Nominal Attributes** (Can have many values)
  - **Multiway- split**
  - The number of outcomes depends on the number of distinct values for the attribute



Test condition for nominal attributes.

Methods for Expressing Attribute Test

- **Nominal Attributes** (Can have many values)
  - **Binary split**
  - Some algorithms, such as CART, produce only binary splits

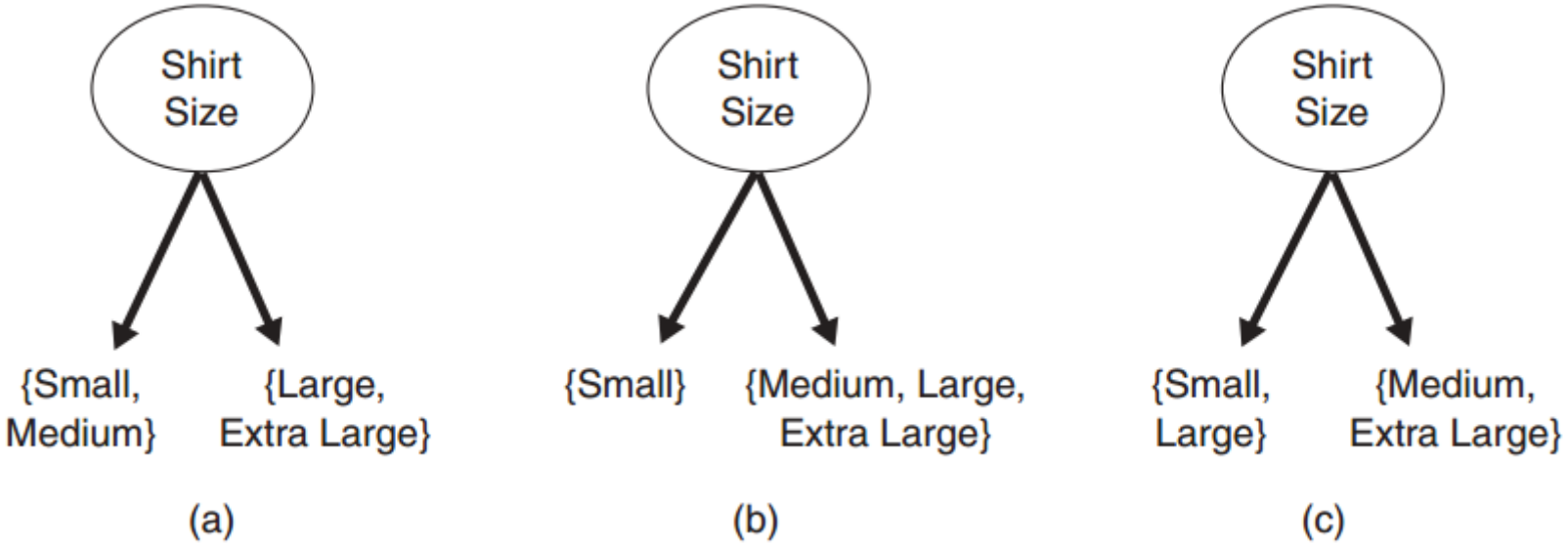


(b) Binary split {by grouping attribute values}

Different ways of grouping the attribute values.

Methods for Expressing Attribute Test

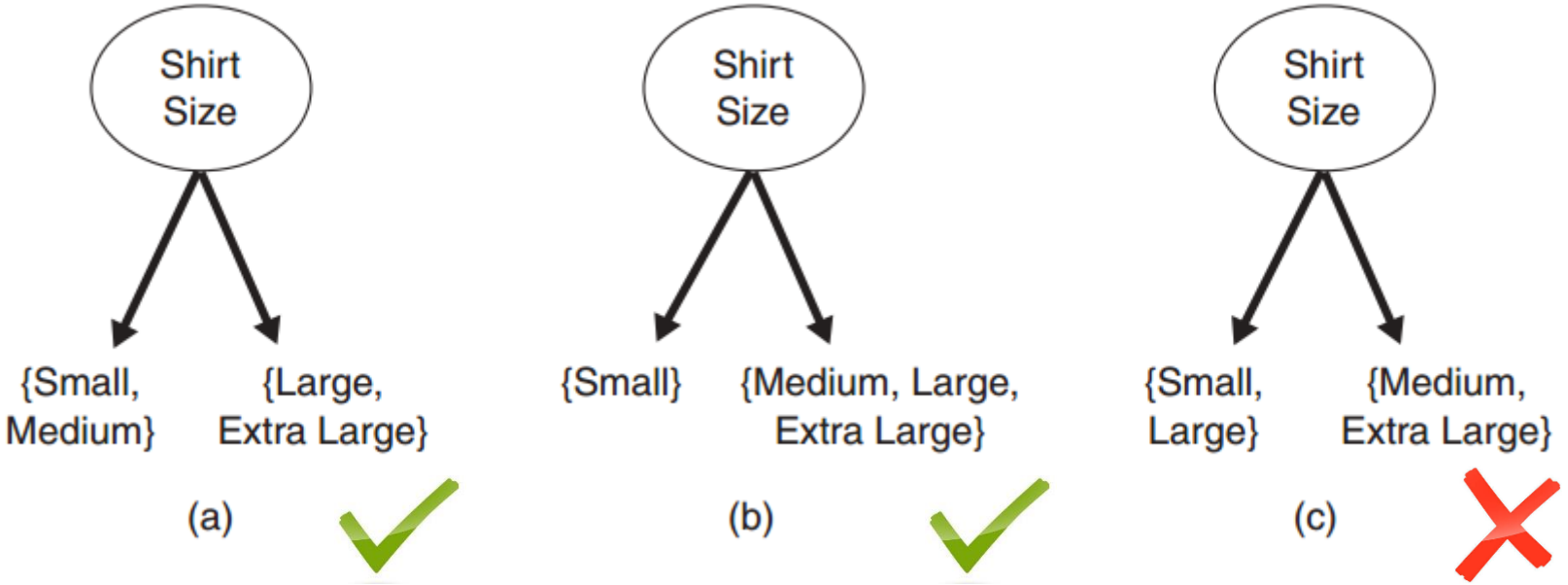
- **Ordinal Attributes**
  - **Multi-way split** and **Binary split**
  - Ordinal attribute values can be grouped as long as the grouping does not violate the order property of the values.



Different ways of grouping the ordinal attribute values.

Methods for Expressing Attribute Test

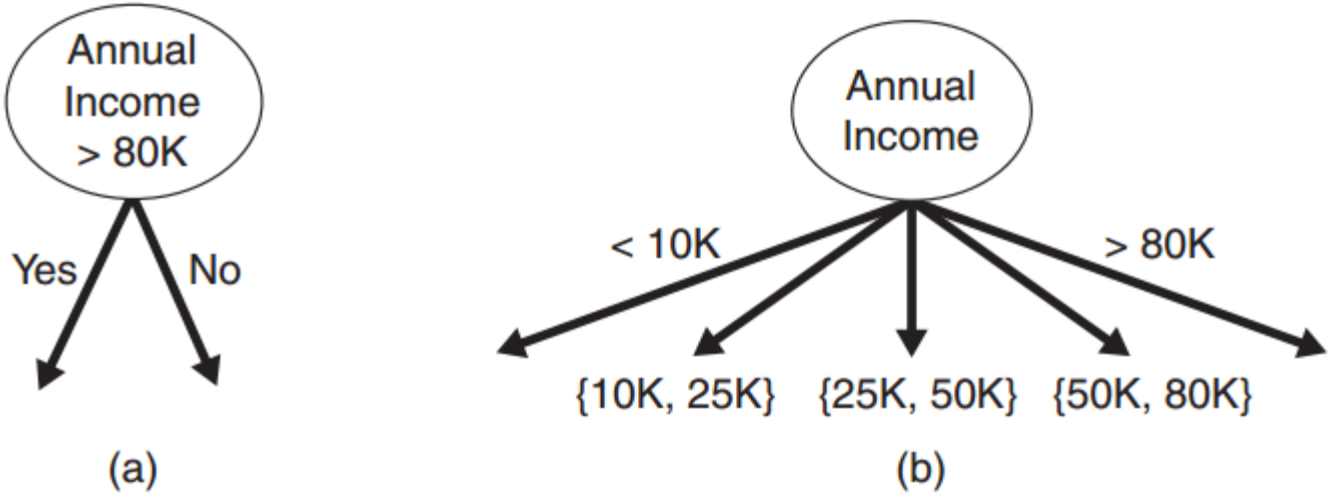
- **Ordinal Attributes**
  - **Multi-way split** and **Binary split**
  - Ordinal attribute values can be grouped as long as the grouping does not violate the order property of the values.



Different ways of grouping the ordinal attribute values.

Methods for Expressing Attribute Test

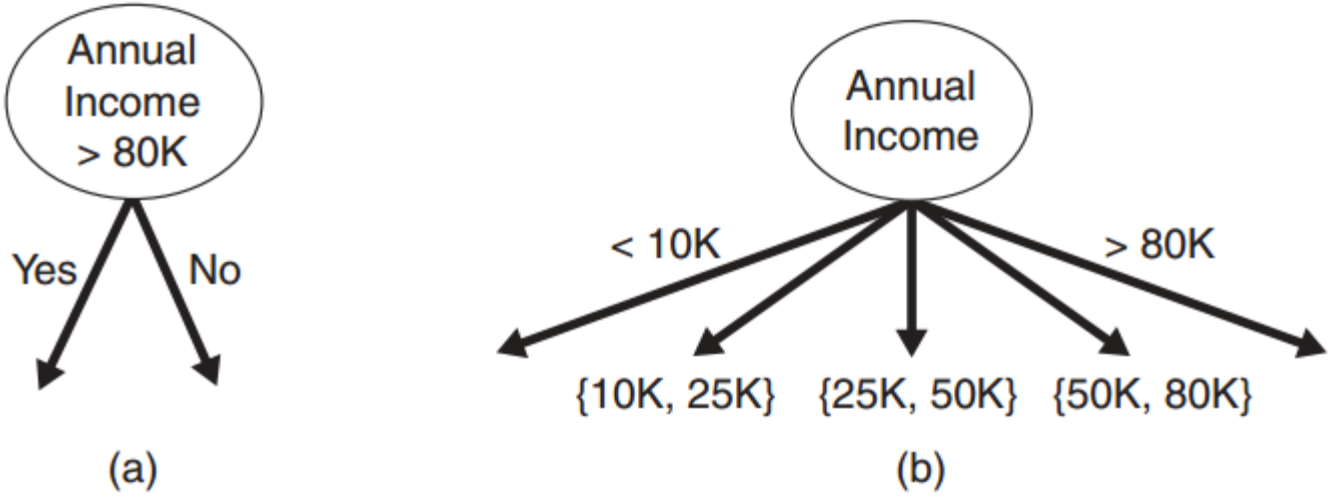
- **Continuous Attributes**
  - **Binary split**
  - The algorithm must consider all possible split position v, and select the one that produces the best partition.



Test condition for continuous attributes.

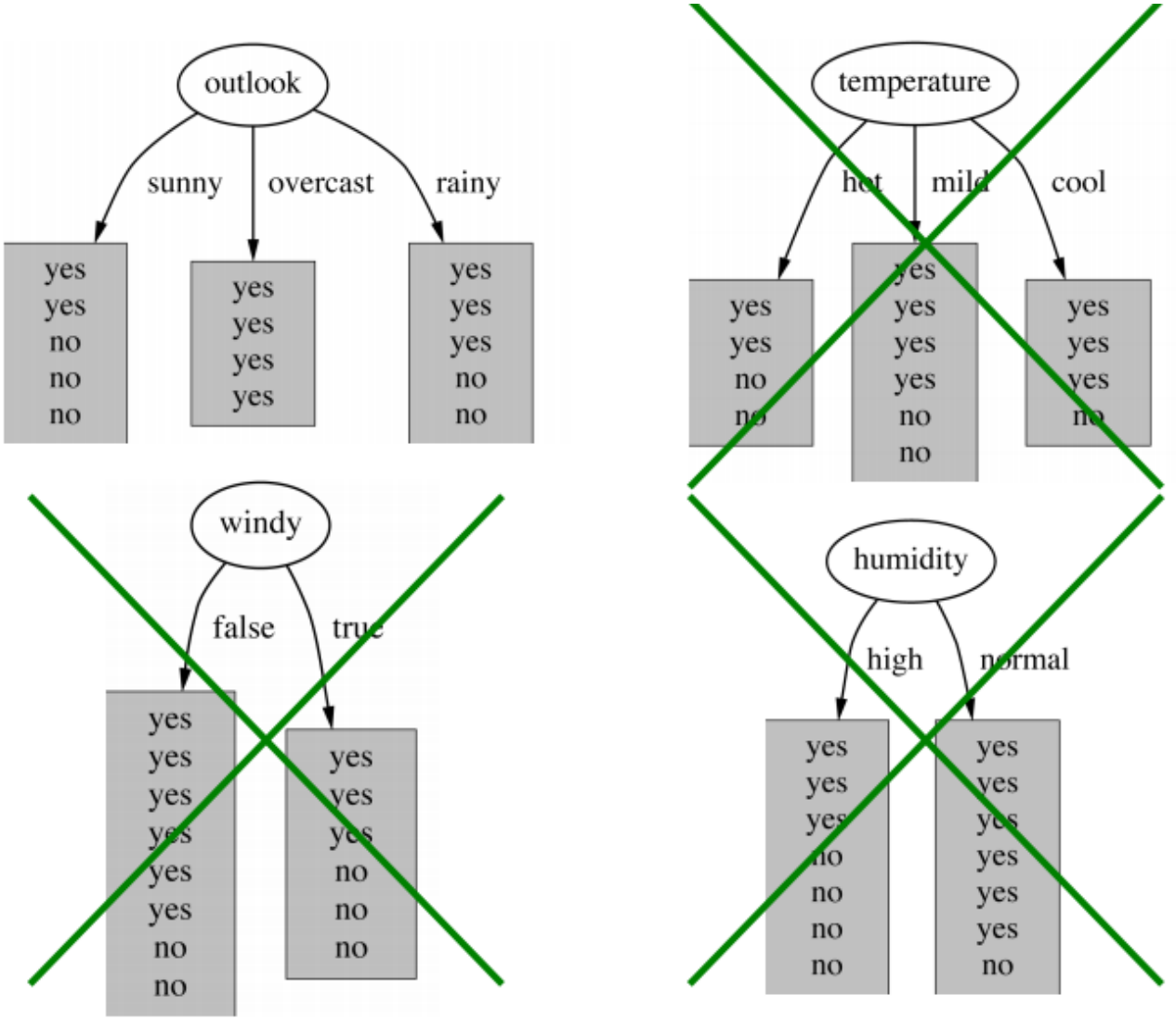
Methods for Expressing Attribute Test

- **Continuous Attributes**
  - **Multi-way split**
  - Must consider all possible ranges of continuous values.
  - Discretization strategies can be used (ordered value will be produced).



Test condition for continuous attributes.

Which Attribute to Select As the Root?



What Is a Good Attribute?

- A good attribute prefers attributes that split the data so that each successor node is as ***pure*** as possible
  - i.e., most examples of each node belongs to the same class

- In other words:
  - We want a measure that prefers attributes that have a high degree of “order“:
    - Maximum order: All examples are of the same class
    - Minimum order: All classes are equally likely

What Is a Good Attribute?

- A good attribute prefers attributes that split the data so that each successor node is as ***pure*** as possible
  - i.e., most examples of each node belongs to the same class

- In other words:
  - We want a measure that prefers attributes that have a high degree of "order":
    - Maximum order: All examples are of the same class
    - Minimum order: All classes are equally likely

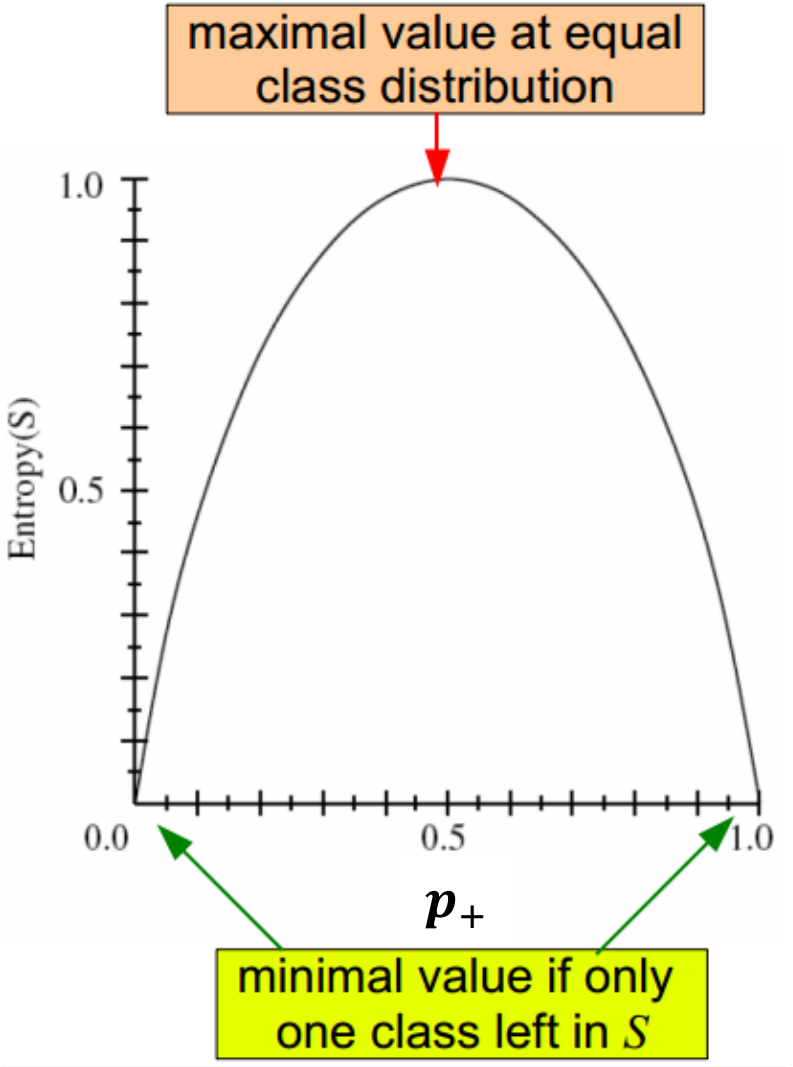
  → Entropy is a promising measure
    - Entropy is the amount of information that is contained
    - All examples of the same class → no information

Entropy (for two classes)

- S is a set of example
- p_+ is the proportion of examples in class +
- $p_- = 1 - p_+$ is the proportion of examples in class -

Entropy

$$E(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$



Entropy (for multiple classes)

- Entropy can be easily generalized for n>2 classes
- p_i is the proportion of examples in S that belong to the i-th class

$$E(S) = -\sum_{i}^{n} p_i \log_2 p_i$$

Task Revisit

Day	Temperature	Outlook	Humidity	Windy	Play Golf?
07-05	hot	sunny	high	false	no
07-06	hot	sunny	high	true	no
07-07	hot	overcast	high	false	yes
07-09	cool	rain	normal	false	yes
07-10	cool	overcast	normal	true	yes
07-12	mild	sunny	high	false	no
07-14	cool	sunny	normal	false	yes
07-15	mild	rain	normal	false	yes
07-20	mild	sunny	normal	true	yes
07-21	mild	overcast	high	true	yes
07-22	hot	overcast	normal	false	yes
07-23	mild	rain	high	true	no
07-26	cool	rain	normal	true	no
07-30	mild	rain	high	false	yes

today	cool	sunny	normal	false	?
tomorrow	mild	sunny	normal	false	?

Entropy (for two classes)

- Outlook = sunny: 2 examples yes, 3 examples no

$$E(outlook = sunny) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$$

Entropy (for two classes)

- Outlook = sunny: 2 examples yes, 3 examples no

$$E(outlook = sunny) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$$

- Outlook= overcast: 4 examples yes, 0 examples no

$$E(outlook = overcast) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = ?$$

Entropy (for two classes)

- Outlook = sunny: 2 examples yes, 3 examples no

$$E(outlook = sunny) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$$

- Outlook= overcast: 4 examples yes, 0 examples no

$$E(outlook = overcast) = -\frac{4}{4}\log_2\frac{4}{4} - \boxed{\frac{0}{4}\log_2\frac{0}{4}} = 0$$

Note: this is normally undefined. Here:=0

Entropy (for two classes)

- Outlook = sunny: 2 examples yes, 3 examples no

$$E(outlook = sunny) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$$

- Outlook= overcast: 4 examples yes, 0 examples no

$$E(outlook = overcast) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0$$

Note: this is normally undefined. Here:=0

- Outlook = rainy: 3 examples yes, 2 examples no

$$E(outlook = rainy) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$$

Entropy (for more classes)

- Entropy can be easily generalized for n>2 classes
- p_i is the proportion of examples in S that belong to the i-th class

$$E(S) = -\sum_{i}^{n} p_i \log_2 p_i$$

Entropy

- **Problem**:
  - Entropy only computes the quality of a single (sub-)set of examples
    - Corresponds to a single value
  - How can we compute the quality of the entire split (entire attribute)?
    - Corresponds to an entire attribute

Average Entropy / Information

- **Solution**:
  - Compute the weighted average over all sets resulting from the split
    - Weighted by their size

$$I(S,A) = \sum_i \frac{|S_i|}{|S|} E(S_i)$$

- **Example**:
  - Average entropy for attribute *Outlook*:

$$I(S, outlook) = \frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971 = 0.693$$

Information Gain

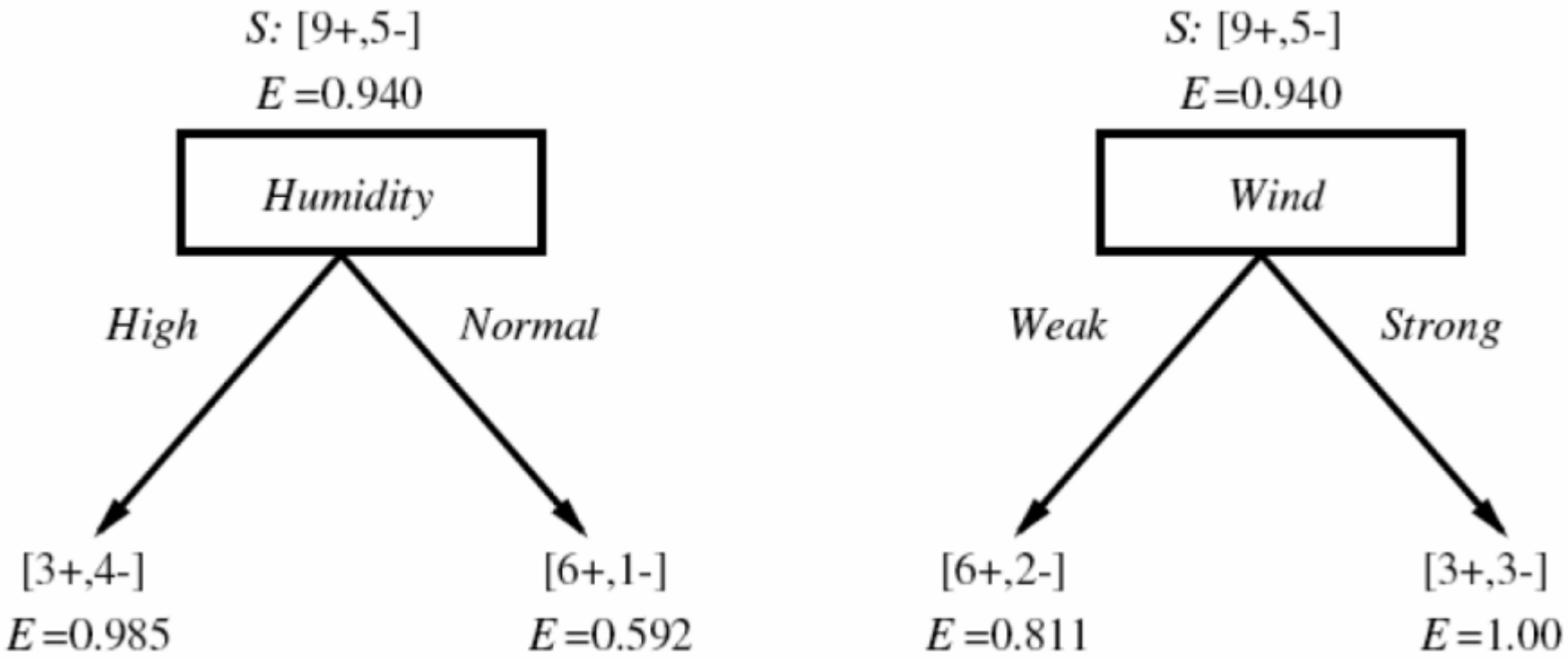
- When an attribute A splits the set S into subsets S_i's
  - We compute the average entropy
  - And compare the entropy of the original set S

- **Information Gain** for attribute A

$$Gain(S, A) = E(S) - I(S, A) = E(S) - \sum_i \frac{|S_i|}{|S|} E(S_i)$$

- The attribute that maximizes the difference is selected
  - I.e., the attribute that reduces the unorderedness most!
- Maximizing information gain is equivalent to minimizing average entropy, because $E(S)$ is constant for all attributes A.

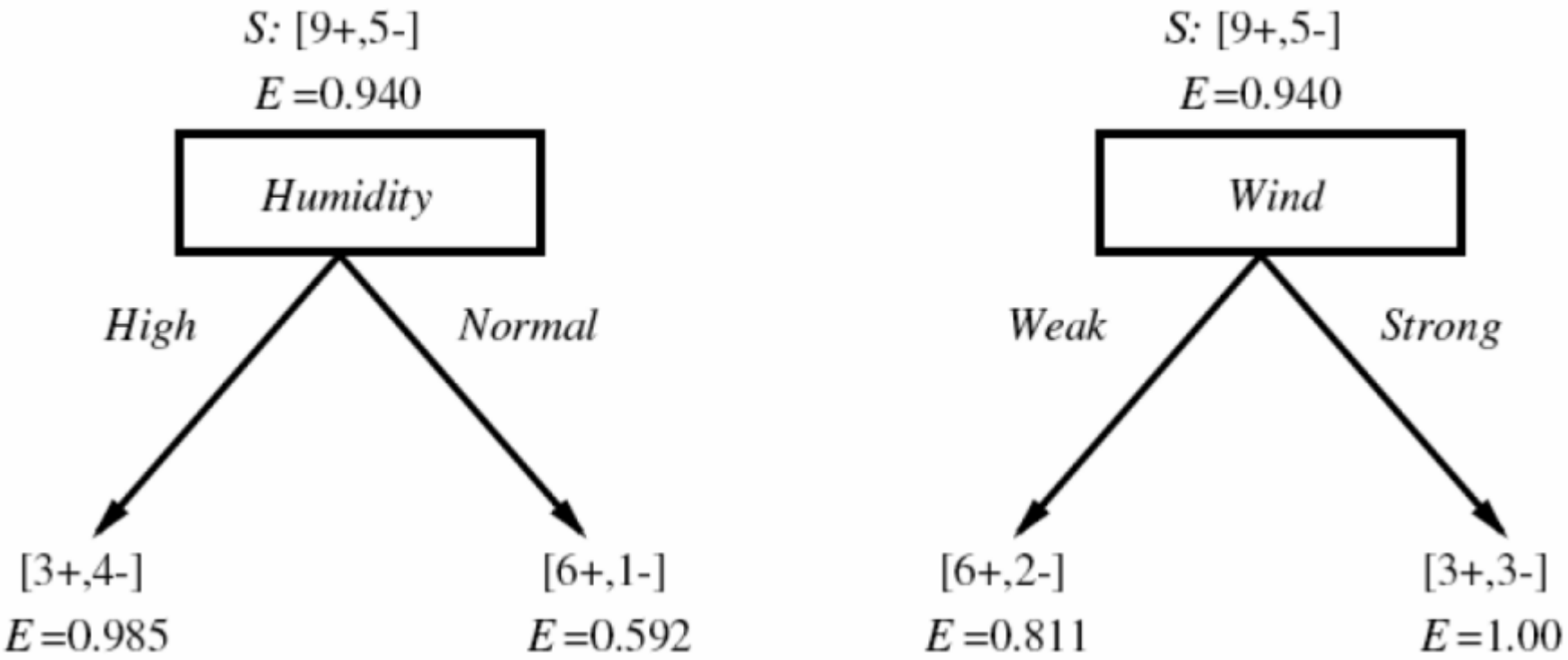
Example



$Gain(S, Humidity) = ?$

$Gain(S, Wind) = ?$

Example

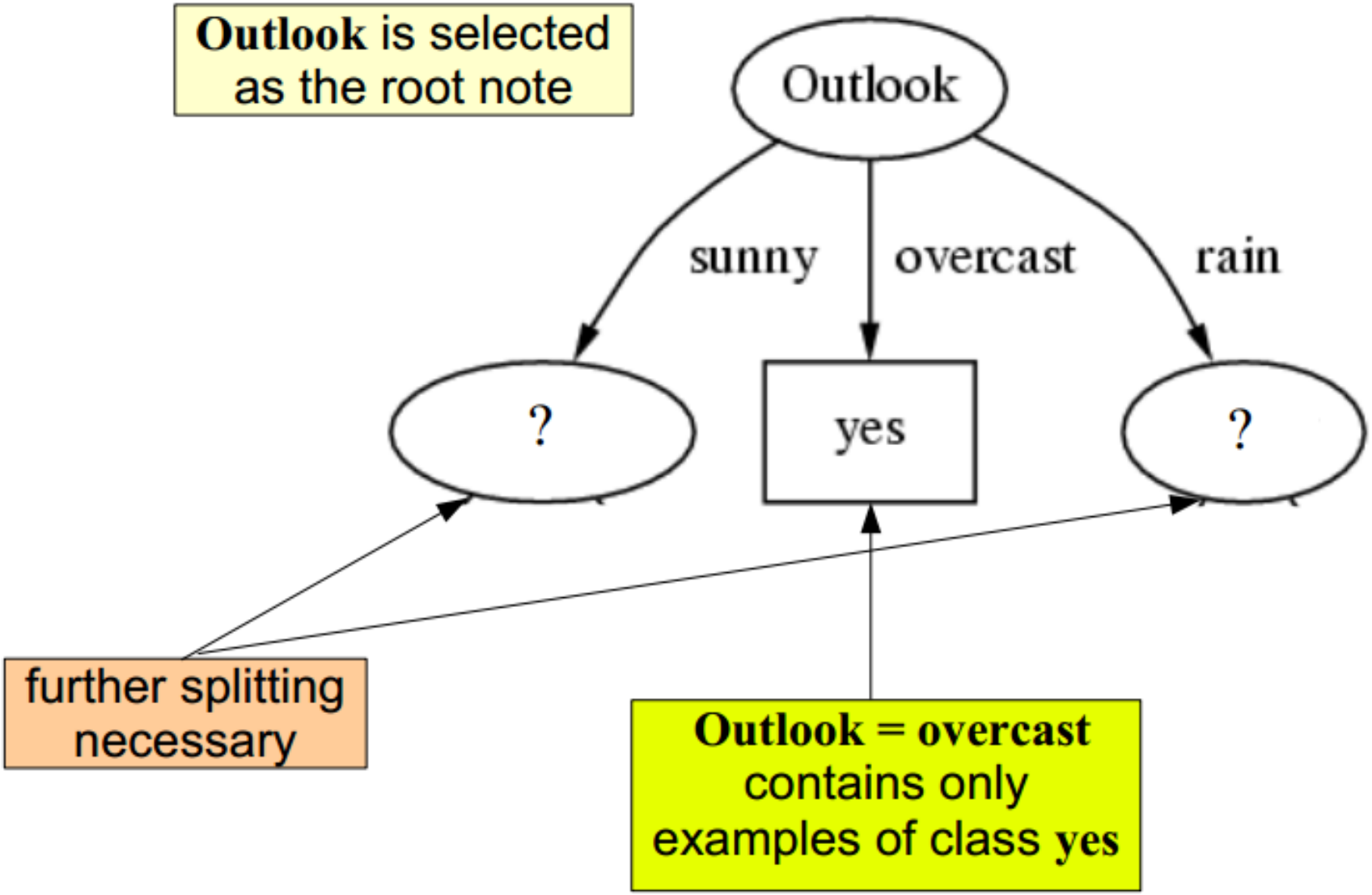


$$Gain(S, Humidity) = 0.940 - \frac{7}{14} \times 0.985 - \frac{7}{14} \times 0.592 = 0.151$$

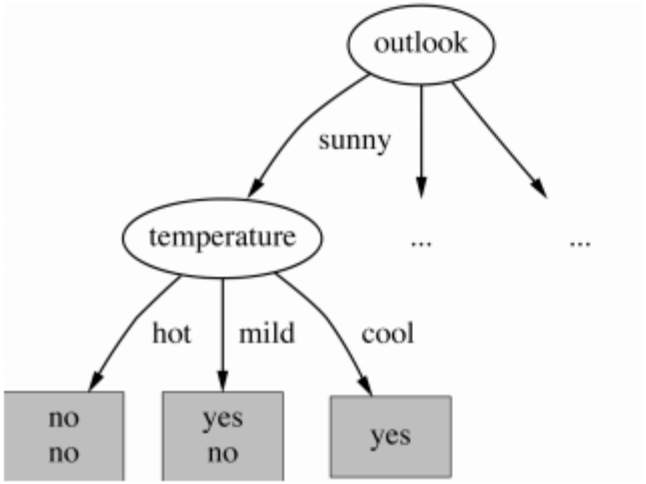
$$Gain(S, Wind) = 0.940 - \frac{8}{14} \times 0.811 - \frac{6}{14} \times 1.0 = 0.048$$

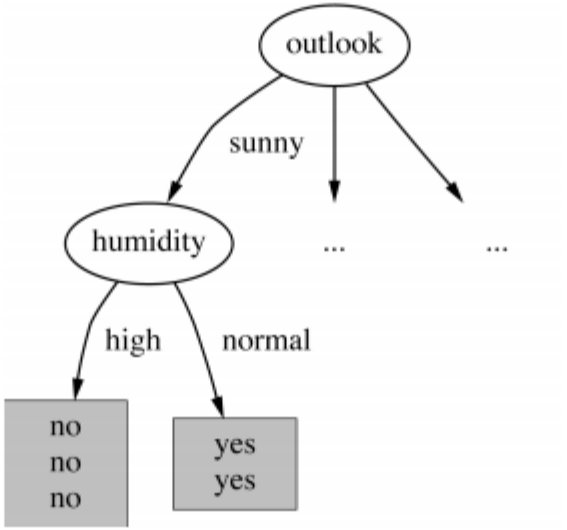
$$Gain(S, Outlook) = 0.246 \qquad Gain(S, Temperature) = 0.029$$

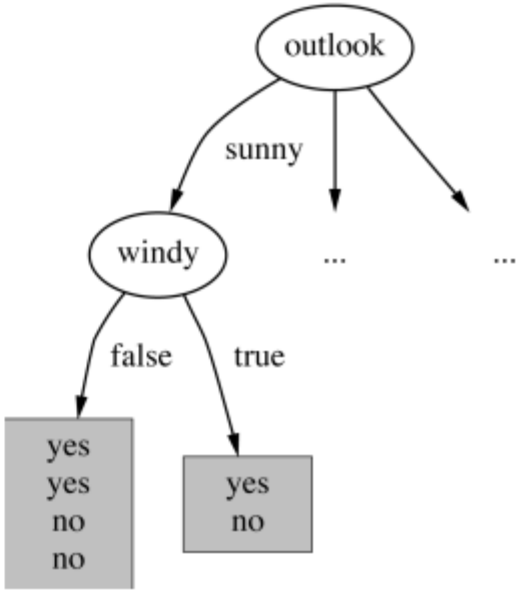
Example


Outlook is selected as the root note
Outlook
sunny
overcast
rain
?
yes
?
further splitting necessary
Outlook = overcast contains only examples of class yes

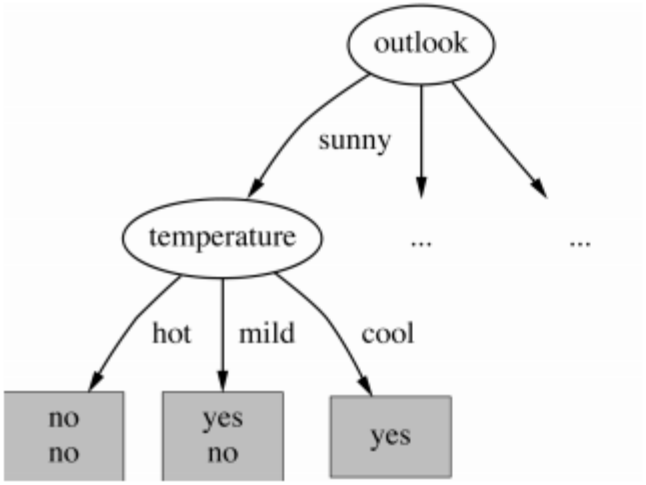
Example

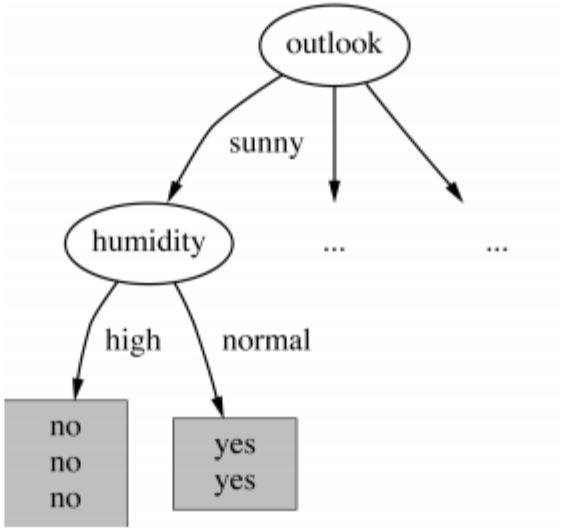

outlook
sunny
temperature
...
...
hot
mild
cool
no
no
yes
no
yes

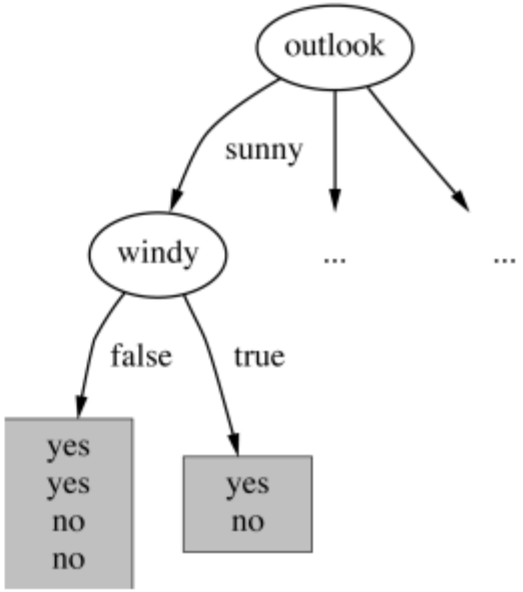

outlook
sunny
humidity
...
...
high
normal
no
no
no
yes
yes


outlook
sunny
windy
...
...
false
true
yes
yes
no
no
yes
no

Example



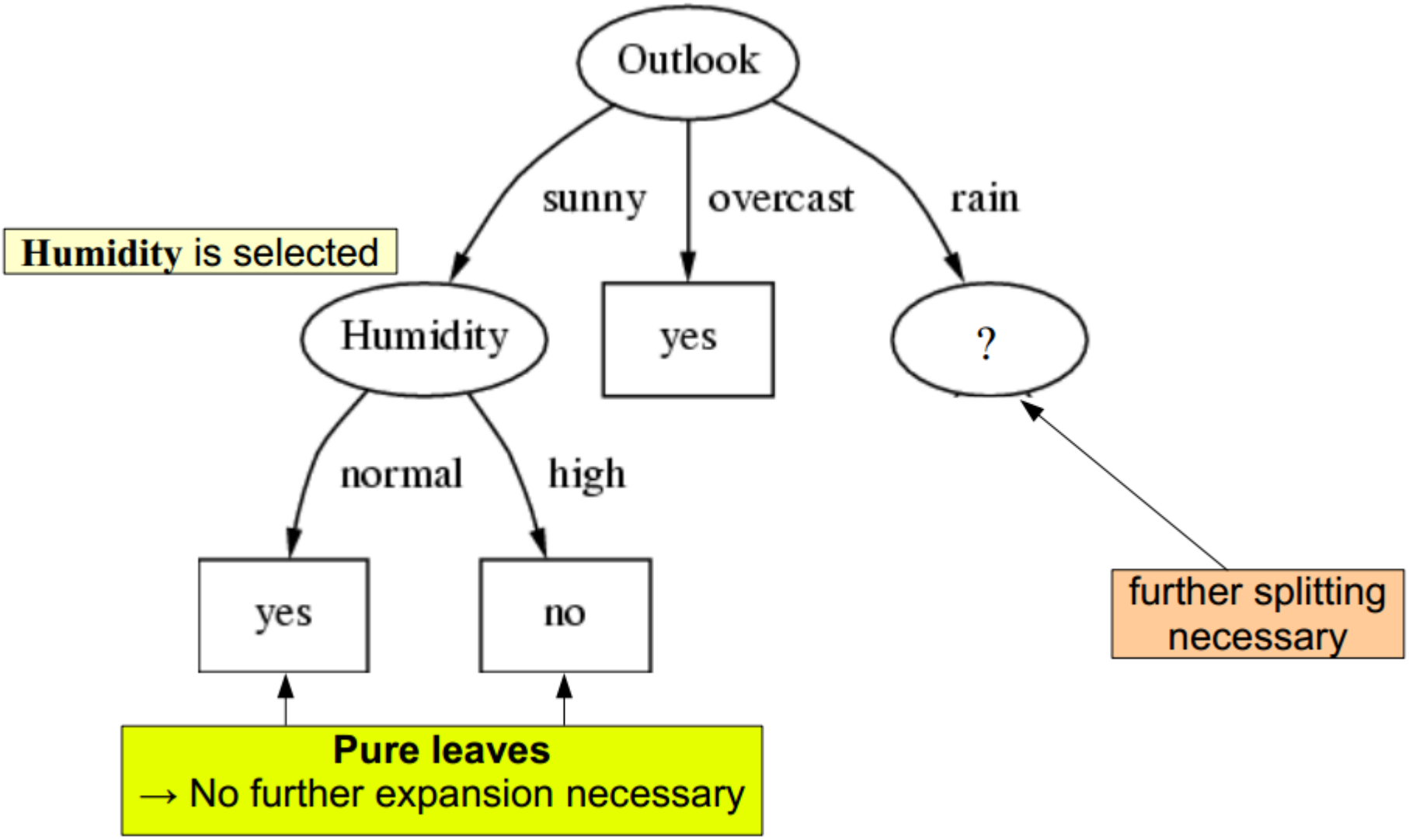




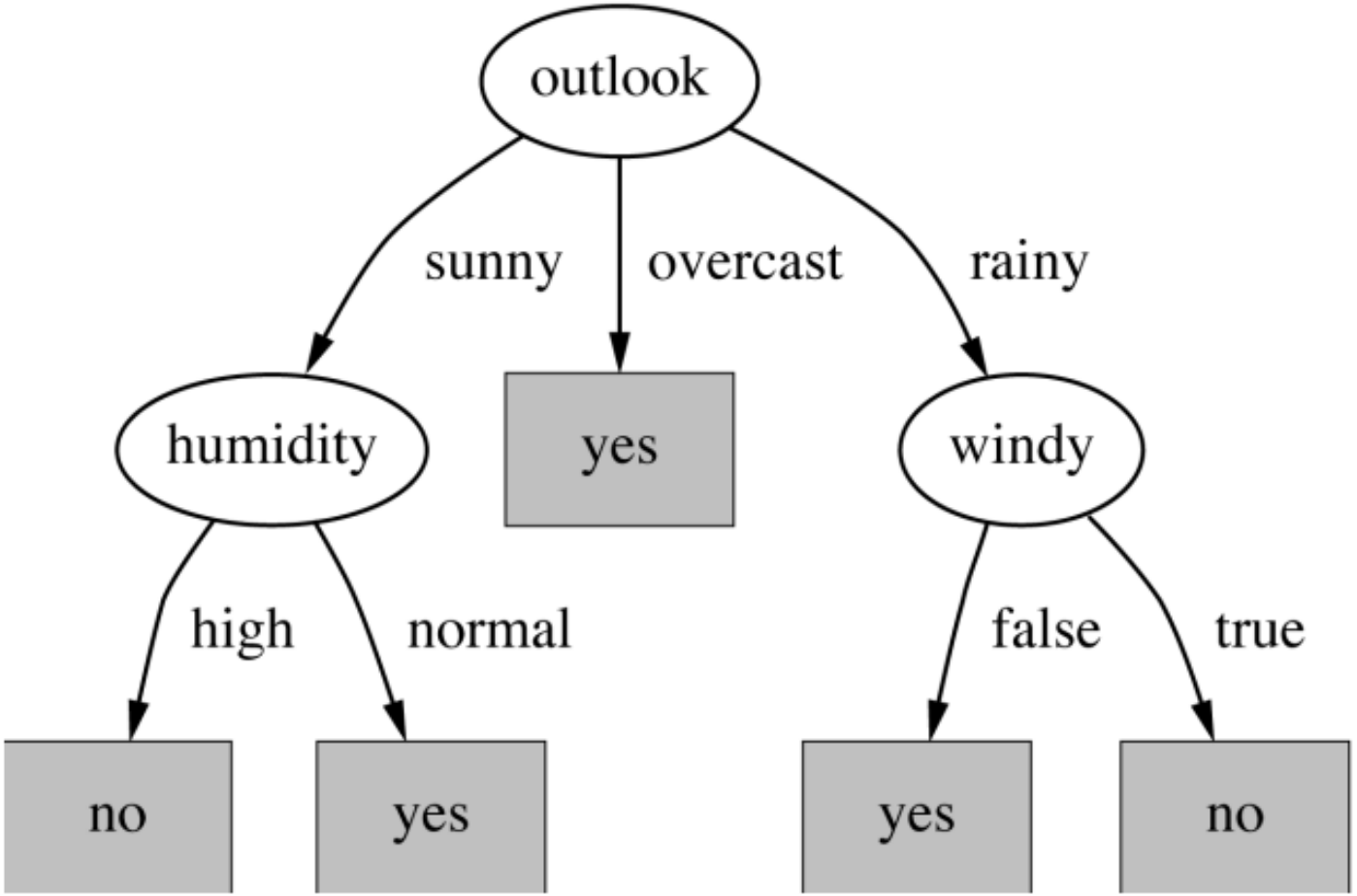
$$\text{Gain}(\textit{Temperature}) = 0.571$$
$$\text{Gain}(\textit{Humidity}) = 0.971$$
$$\text{Gain}(\textit{Windy}) = 0.020$$

Humidity is selected

Example


Outlook
sunny
overcast
rain
Humidity is selected
Humidity
yes
?
normal
high
yes
no
further splitting necessary
Pure leaves
→ No further expansion necessary

Final Decision Tree



Task Revisit

Day	Temperature	Outlook	Humidity	Windy	Play Golf?
07-05	hot	sunny	high	false	no
07-06	hot	sunny	high	true	no
07-07	hot	overcast	high	false	yes
07-09	cool	rain	normal	false	yes
07-10	cool	overcast	normal	true	yes
07-12	mild	sunny	high	false	no
07-14	cool	sunny	normal	false	yes
07-15	mild	rain	normal	false	yes
07-20	mild	sunny	normal	true	yes
07-21	mild	overcast	high	true	yes
07-22	hot	overcast	normal	false	yes
07-23	mild	rain	high	true	no
07-26	cool	rain	normal	true	no
07-30	mild	rain	high	false	yes

today	cool	sunny	normal	false	?
tomorrow	mild	sunny	normal	false	?

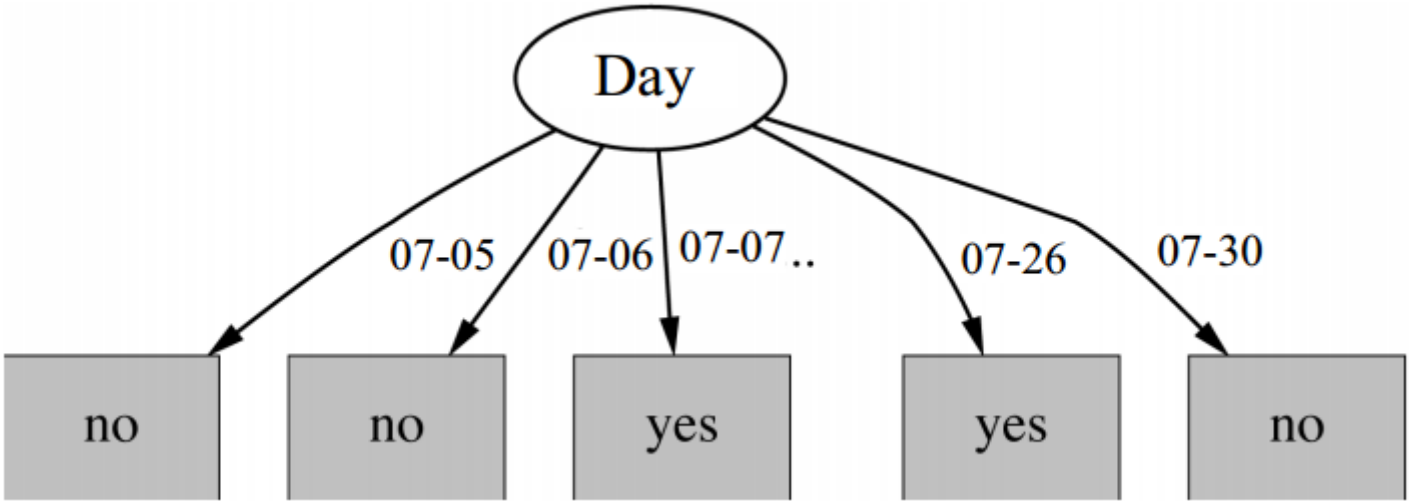
High-branching Attributes

- **Problematic**: attributes with a large number of values
  - Extreme case: each example has its own value
  - E.g., example ID; Day attribute in weather data
- Subsets are more likely to be pure if there is a large number of different attribute values
  - Information gain is **biased** towards choosing attributes with a large number of values

Decision Tree for Day Attribute

- Entropy of split:
  - Information gain is maximal for Day (0.940 bits)

$$I(S, Day) = \frac{1}{14}\big(E([0,1]) + E([0,1]) + \cdots, E([0,1])\big) = 0$$



High-branching Attributes

- **Problematic**: attributes with a large number of values
  - Extreme case: each example has its own value
  - E.g., example ID; Day attribute in weather data
- Subsets are more likely to be pure if there is a large number of different attribute values
  - Information gain is **biased** towards choosing attributes with a large number of values

- This may cause several problems:
  - **Overfitting**
    - Select an attribute that is non-optimal for prediction
  - **Fragmentation**
    - Data are fragmented into too many small sets

Intrinsic Information of An Attribute

- Intrinsic information of a split
  - Entropy of the distribution of instances into branches
  - i.e., how much information do we need to tell which branch an instance belongs to

$$IntI(S, A) = -\sum_i \frac{|S_i|}{|S|} \log\left(\frac{|S_i|}{|S|}\right)$$

- Example:
  - Intrinsic information of Day attribute:

$$IntI(Day) = 14 \times \left(-\frac{1}{14} \log \frac{1}{14}\right) = 3.807$$

- Observation:
  - Attributes with higher intrinsic information are less useful.

Gain Ratio

- Modification of the *information gain* that reduces its bias towards multi-valued attributes
- Takes number and size of branches into account
  - Corrects the information gain by taking the intrinsic information of a split into account

- Definition of Gain Ratio:

$$GR(S,A) = \frac{Gain(S,A)}{IntI(S,A)}$$

- Example:
  - Gain ratio of Day attribute

$$GR(Day) = \frac{0.940}{3.807} = 0.246$$

Gain Ratio of Other Attributes

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Split info: info([5,4,5])	1.577	Split info: info([4,6,4])	1.557
Gain ratio: 0.247/1.577	0.157	Gain ratio: 0.029/1.557	0.019
Humidity		**Windy**	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048
Split info: info([7,7])	1.000	Split info: info([8,6])	0.985
Gain ratio: 0.152/1	0.152	Gain ratio: 0.048/0.985	0.049

Gain Ratio of Other Attributes

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Split info: info([5,4,5])	1.577	Split info: info([4,6,4])	1.557
Gain ratio: 0.247/1.577	0.157	Gain ratio: 0.029/1.557	0.019
Humidity		**Windy**	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048
Split info: info([7,7])	1.000	Split info: info([8,6])	0.985
Gain ratio: 0.152/1	0.152	Gain ratio: 0.048/0.985	0.049

- Day attribute would still win…
  - One has to be careful which attributes to add…
- Nevertheless： Gain ratio is more reliable than information gain

Gini Index

- **Gini Index:** A popular alternative measures to Information Gain
- Used in e.g., in **CART** (**C**lassification **A**nd **R**egression **T**rees)
- Impurity measure (instead of entropy)

$$Gini(S) = 1 - \sum_i {p_i}^2$$

- Average Gini index (instead of average entropy / information)

$$Gini(S, A) = -\sum_i \frac{|S_i|}{|S|} Gini(S_i)$$

- Gini Gain
  - could be defined analogously to information gain
  - but typically average Gini index is minimized instead of maximizing Gini gain.

Comparison among Splitting Criteria

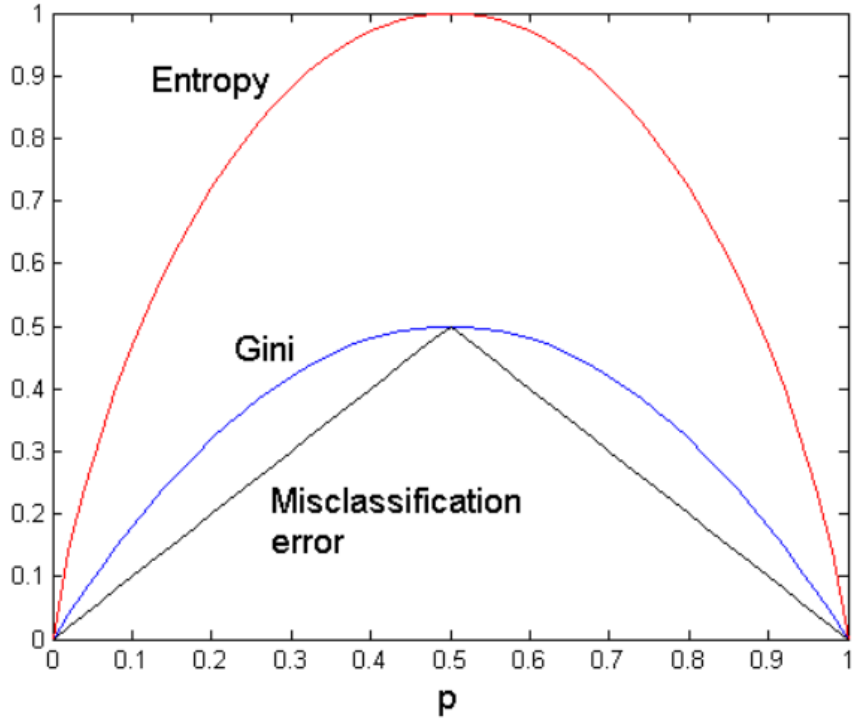
- Splitting Criteria (impurity measures)

$$E(S) = -\sum_i p_i \log_2 p_i$$

$$Gini(S) = 1 - \sum_i {p_i}^2$$

$$misc\_error = 1 - \max_i p_i$$

- Consistency



Comparison among the impurity measures for binary classification

Comparison among Splitting Criteria

- Splitting Criteria (impurity measures)

Node N_1	Count
Class=0	0
Class=1	6

$$\text{Gini} = 1 - (0/6)^2 - (6/6)^2 = 0$$
$$\text{Entropy} = -(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$$
$$\text{Error} = 1 - \max[0/6, 6/6] = 0$$

Node N_2	Count
Class=0	1
Class=1	5

$$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$$
$$\text{Entropy} = -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$$
$$\text{Error} = 1 - \max[1/6, 5/6] = 0.167$$

Node N_3	Count
Class=0	3
Class=1	3

$$\text{Gini} = 1 - (3/6)^2 - (3/6)^2 = 0.5$$
$$\text{Entropy} = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$$
$$\text{Error} = 1 - \max[3/6, 3/6] = 0.5$$

Comparison among Splitting Criteria

- Splitting Criteria (impurity measures)

Node N_1	Count
Class=0	0
Class=1	6

$$\text{Gini} = 1 - (0/6)^2 - (6/6)^2 = 0$$
$$\text{Entropy} = -(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$$
$$\text{Error} = 1 - \max[0/6, 6/6] = 0$$

Node N_2	Count
Class=0	1
Class=1	5

$$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$$
$$\text{Entropy} = -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$$
$$\text{Error} = 1 - \max[1/6, 5/6] = 0.167$$

Node N_3	Count
Class=0	3
Class=1	3

$$\text{Gini} = 1 - (3/6)^2 - (3/6)^2 = 0.5$$
$$\text{Entropy} = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$$
$$\text{Error} = 1 - \max[3/6, 3/6] = 0.5$$

- Despite their consistency, the attribute chosen as the test condition may vary depending on the choice of impurity measure.

Algorithm for Decision Tree Induction

- A skeleton decision tree induction algorithm *TreeGrowth* is shown.
- The input consists of the training records *E* and the attribute set *F*.

Algorithm 4.1 A skeleton decision tree induction algorithm.

```
TreeGrowth (E, F)
 1: if stopping_cond(E,F) = true then
 2:    leaf = createNode().
 3:    leaf.label = Classify(E).
 4:    return leaf.
 5: else
 6:    root = createNode().
 7:    root.test_cond = find_best_split(E, F).
 8:    let V = {v|v is a possible outcome of root.test_cond }.
 9:    for each v ∈ V do
10:       E_v = {e | root.test_cond(e) = v and e ∈ E}.
11:       child = TreeGrowth(E_v, F).
12:       add child as descendent of root and label the edge (root → child) as v.
13:    end for
14: end if
15: return root.
```

Algorithm for Decision Tree Induction

- A skeleton decision tree induction algorithm *TreeGrowth* is shown.
- The input consists of the training records *E* and the attribute set *F*.

Algorithm 4.1 A skeleton decision tree induction algorithm.

```
TreeGrowth (E, F)
 1: if stopping_cond(E,F) = true then
 2:    leaf = createNode().
 3:    leaf.label = Classify(E).
 4:    return leaf.
 5: else
 6:    root = createNode().
 7:    root.test_cond = find_best_split(E, F).
 8:    let V = {v|v is a possible outcome of root.test_cond }.
 9:    for each v ∈ V do
10:       E_v = {e | root.test_cond(e) = v and e ∈ E}.
11:       child = TreeGrowth(E_v, F).
12:       add child as descendent of root and label the edge (root → child) as v.
13:    end for
14: end if
15: return root.
```

- *Classify(E)* determines the class label to be assigned to a leaf node.
- In most cases, the leaf node is assigned by the majority voting.

Industrial-strength algorithms

- For an algorithm to be useful in a wide range of real-world applications it must:
  - Permit numeric attributes
  - Allow missing values
  - Be robust in the presence of noise

  → ID3 needs to be extended to be able to deal with real-world data

- Result: **C4.5**
  - (Probably) Most widely-used learning algorithm
    - http://www.rulequest.com/Personal/
  - Re-implementation of C4.5 Release 8 in Weka: J4.8
  - Commercial successor: C5.0

Numerical Attributes

Numeric (Continuous) Attributes

- Standard method: **binary partition**
  - E.g. temp < 45 and temp ≥ 45
- Solution is straightforward:
  - Sort the attribute values $\{a^1, a^2, \dots, a^n\}$
  - Evaluate info gain (or other measure) for every possible split point of attribute
  - Split points can be places between values, i.e., $(a^i + a^{i+1})/2$, or directly at values a^i
  - Choose the "best" split point
  - Info gain for best split point is info gain for attribute

$$Gain(D, a) = \max_{t \in T_a} Gain(D, a, t) \qquad T_a = \{\frac{a^i + a^{i+1}}{2} | 1 \leq i \leq n-1\}$$

Split candidates

- Computationally more demanding

Example

- Assume a numerical attribute for Temperature
- First step:
  - **Sort** all examples according to the value of this attribute
  - Could look like this:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

Example

- Assume a numerical attribute for Temperature
- First step:
  - **Sort** all examples according to the value of this attribute
  - Could look like this:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

- One possible split
  - E.g., Temperature < 71.5:
    Temperature ≥ 71.5:

Example

- Assume a numerical attribute for Temperature
- First step:
  - **Sort** all examples according to the value of this attribute
  - Could look like this:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

- One possible split
  - E.g., Temperature < 71.5: yes/4, no/2
    Temperature ≥ 71.5: yes/5, no/3

$$I(Temp@71.5) = \frac{6}{14}E(Temp < 71.5) + \frac{8}{14}E(Temp \geq 71.5) = 0.939$$

Efficient Computation

- Efficient computation needs only one scan through the values!
  - Linearly scan the sorted values, each time updating the count matrix and computing the evaluation measure
  - Choose the split position that has the best value

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Taxable Income										
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72	80	87	92	97	110	122	172	230
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >
Yes	0 3	0 3	0 3	0 3	1 2	2 1	3 0	3 0	3 0	3 0	3 0
No	0 7	1 6	2 5	3 4	3 4	3 4	3 4	4 3	5 2	6 1	7 0
Gini	0.420	0.400	0.375	0.343	0.417	0.400	*0.300*	0.343	0.375	0.400	0.420

Efficient Computation

- Can be further optimized by considering only candidate split positions located between two adjacent records with different class labels.
- This reduces the number of candidate positions from 11 to 2.

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Taxable Income										
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72	80	87	92	97	110	122	172	230

	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		*0.300*		0.343		0.375		0.400		0.420	

Missing Data

Missing Data

Day	Temperature	Outlook	Humidity	Windy	Play Golf?
07-05	hot	sunny	high	false	no
07-06	hot	sunny	high	true	no
07-07	hot	overcast	high	false	yes
07-09	cool	rain	?	false	yes
07-10	cool	overcast	normal	true	yes
07-12	mild	sunny	high	false	no
07-14	cool	sunny	normal	false	yes
07-15	mild	rain	normal	false	yes
07-20	mild	sunny	normal	true	yes
07-21	mild	overcast	high	true	yes
07-22	hot	overcast	normal	false	yes
07-23	mild	?	high	true	no
07-26	cool	rain	normal	true	no
07-30	mild	rain	high	false	yes

today	cool	sunny	normal	?	?
tomorrow	mild	sunny	normal	false	?

Missing Values

- Assume that attribute a has V possible values $\{a^1, a^2, \ldots, a^V\}$.
- $\tilde{S}$: subset of instances in S whose values of a are not missing.
- $\tilde{S}_i$: subset of instances in $\tilde{S}$ whose values of attribute a is a_i.
- $\tilde{S}^k$: subset of instances in $\tilde{S}$ belonging to the k-th class (k=1,…, $|\mathcal{Y}|$).
- Originally,

$$Gain(S,a) = E(S) - I(S,a) = E(S) - \sum_i \frac{|S_i|}{|S|} E(S_i)$$

- Now we modify the information gain as follows,

$$Gain(S,a) = \rho \times Gain(\tilde{S},a) = \rho \times \left(E(\tilde{S}) - \sum_i \tilde{r}_i E(\tilde{S}_i) \right)$$

proportionally $\rho = \frac{|\tilde{S}|}{|S|}$ $\tilde{r}_i = \frac{|\tilde{S}_i|}{|\tilde{S}|}$

Ignore all instances whose values of attribute a are unknown

$$E(\tilde{S}) = -\sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k \log_2 \tilde{p}_k \qquad \tilde{p}_k = \frac{|\tilde{S}^k|}{|\tilde{S}|}$$

Missing Values

- In the case of gain ratio, the denominator should be calculated as if the missing values represent an additional value in the attribute domain.

$$GR(S,a) = \frac{Gain(S,a)}{IntI(S,a)} \qquad IntI(S,a) = -\sum_i \frac{|S_i|}{|S|} \log\left(\frac{|S_i|}{|S|}\right)$$

$$GR(S,a) = \frac{\rho \times Gain(\tilde{S},a)}{-\frac{|S\setminus\tilde{S}|}{|S|} \log\left(\frac{|S\setminus\tilde{S}|}{|S|}\right) - \sum_i \frac{|\tilde{S}_i|}{|S|} \log\left(\frac{|\tilde{S}_i|}{|S|}\right)}$$

C4.5 can induce from a training set that incorporates missing values by using the modified gain ratio criteria.

Missing Values

- If an instance with a missing value needs to be tested:
  - Split the instance into fractional instances (*pieces*)
  - One piece for each outgoing branch of the node
  - A piece going down a branch receives a weight proportional to the popularity of the branch
  - Weights sum to 1

$$\tilde{r}_i = \frac{|\tilde{S}_i|}{|\tilde{S}|}$$

Quiz 1

Q: Is a tree with only pure leafs always the best classifier you can have?

Quiz 1

Q: Is a tree with only pure leafs always the best classifier you can have?

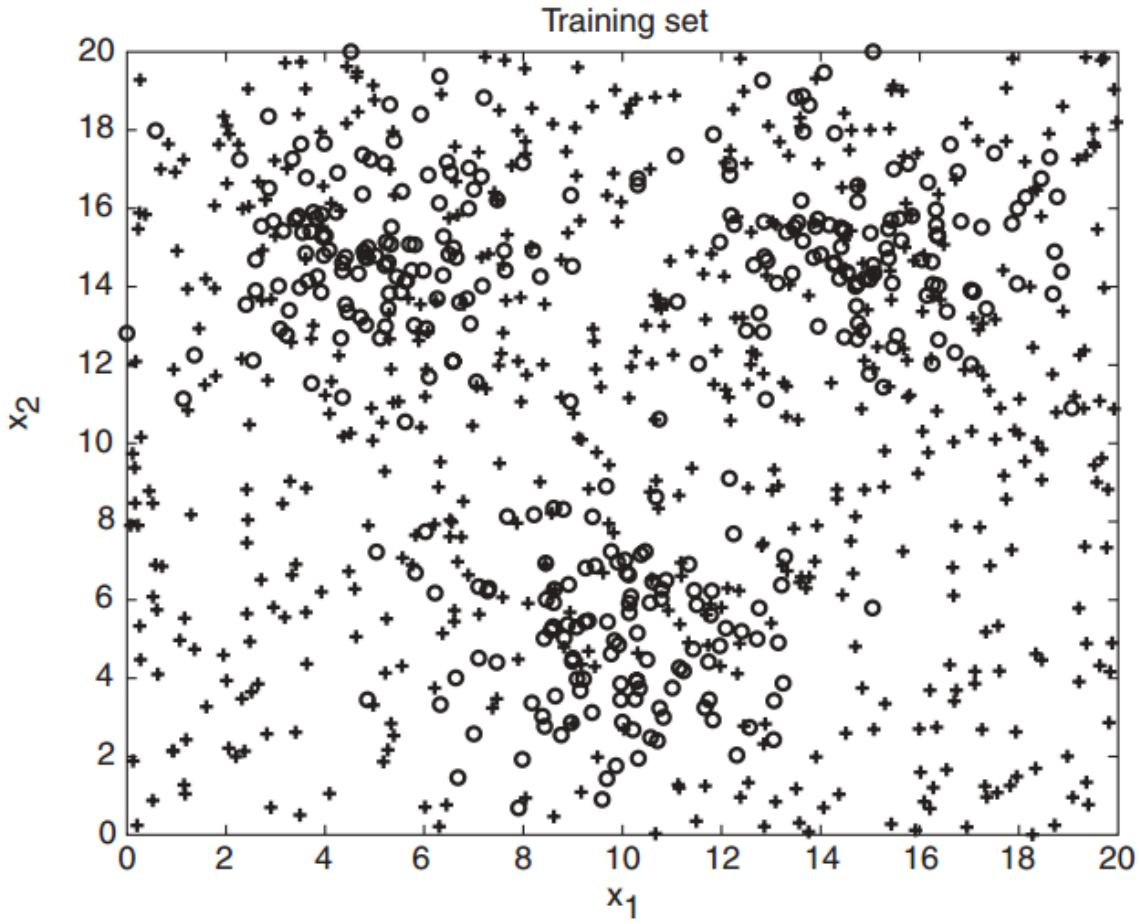
A: No.

This tree is the best classifier on the training set, but possibly not on new and unseen data. Because of overfitting, the tree may not generalize very well.

Overfitting

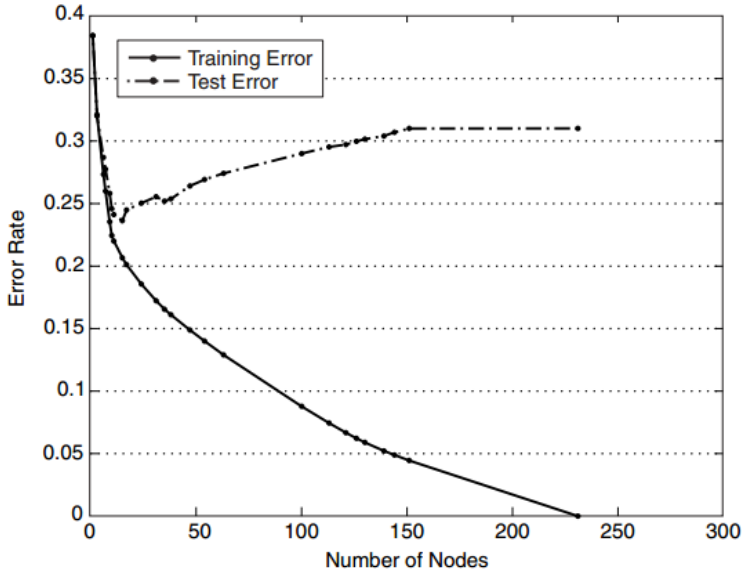
Overfitting Example in 2-D Data

- The dataset contains data points of two classes: ‘o’ and ‘+’.
- ‘o’ are generated from a mixture of three Gaussian distributions.
- ‘+’ are generated from a uniform distribution



Overfitting Example in 2-D Data

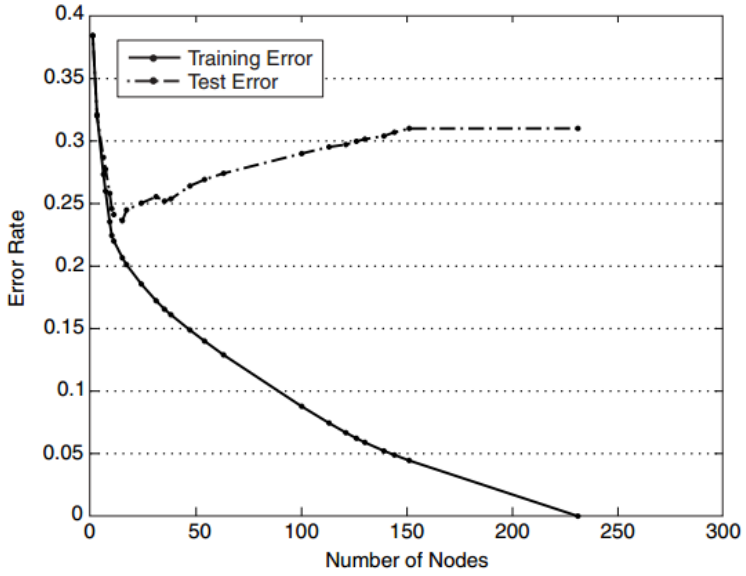
- Underfitting occurs because the model has yet to learn the true structure of the data.
- Overfitting occurs once the tree becomes too large.
- The test error begins to increase even though the training error continues to decrease.



Training and test error.

Overfitting Example in 2-D Data

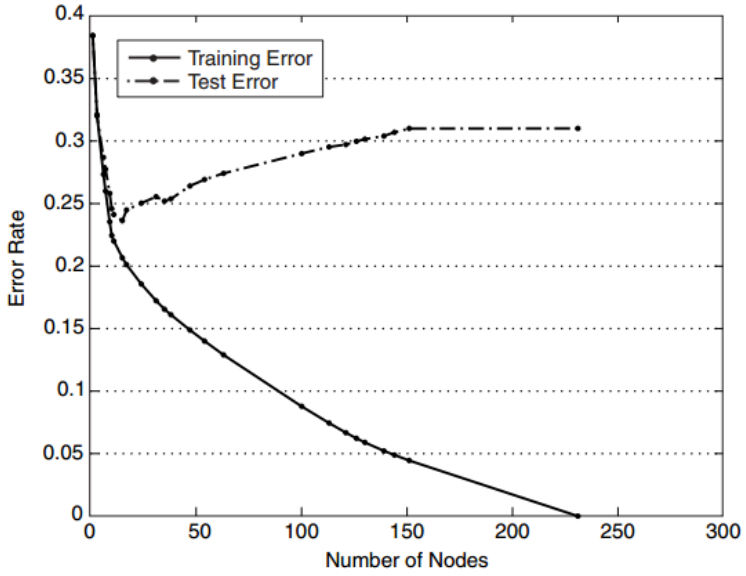
- The leaf nodes of the tree can be expanded until it perfectly fits the training data.
- Note a “perfect“ fit on the training data can always be found for a decision tree! (except when data are contradictory)



Training and test error.

Overfitting Example in 2-D Data

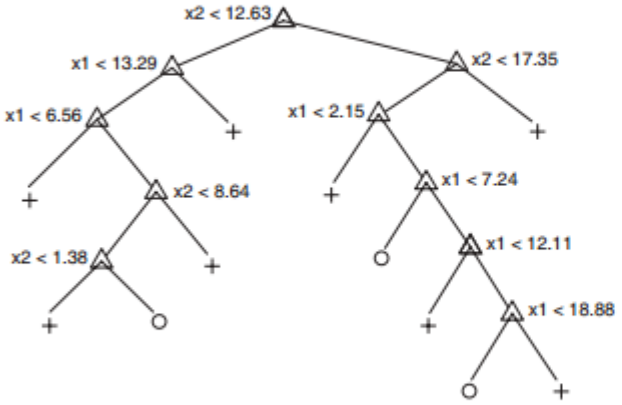
- Although the training error for a complex tree can be zero, its test error can be large because that the tree may contain nodes that accidently fit some of the noise points.
- Such nodes lead to the poor generalization performance.



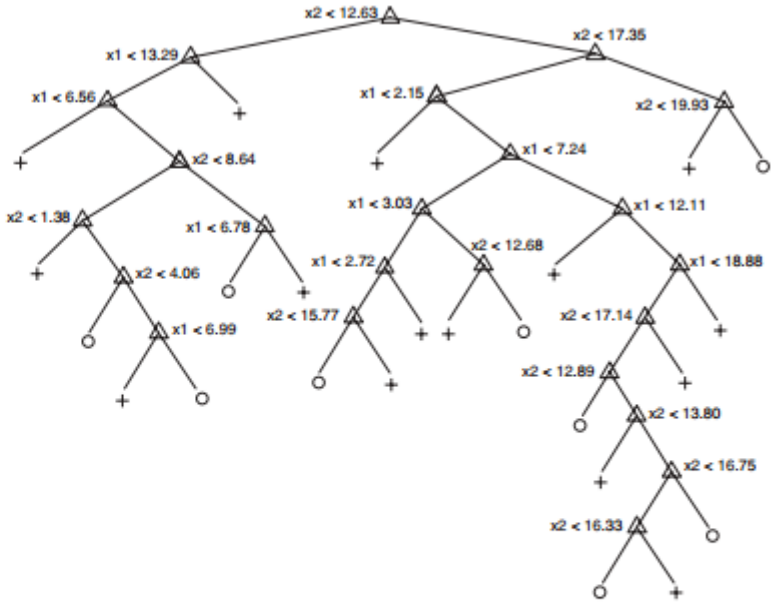
Training and test error.

Overfitting Example in 2-D Data

- The tree that contains the smaller number of nodes has a higher training error rate, but a lower test error rate compared to the more complex tree.



(a) Decision tree with 11 leaf nodes.



(b) Decision tree with 24 leaf nodes.

Decision trees with different model complexities.

Overfitting Due to the Presence of Noise

- Consider the following training and test dataset for the mammal classification problem.
- Two training records are mislabeled: bats and whales.

An example training and test set for classifying mammals.

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
porcupine	warm-blooded	yes	yes	yes	yes
cat	warm-blooded	yes	yes	no	yes
bat	warm-blooded	yes	no	yes	no*
whale	warm-blooded	yes	no	no	no*
salamander	cold-blooded	no	yes	yes	no
komodo dragon	cold-blooded	no	yes	no	no
python	cold-blooded	no	no	yes	no
salmon	cold-blooded	no	no	no	no
eagle	warm-blooded	no	no	no	no
guppy	cold-blooded	yes	no	no	no

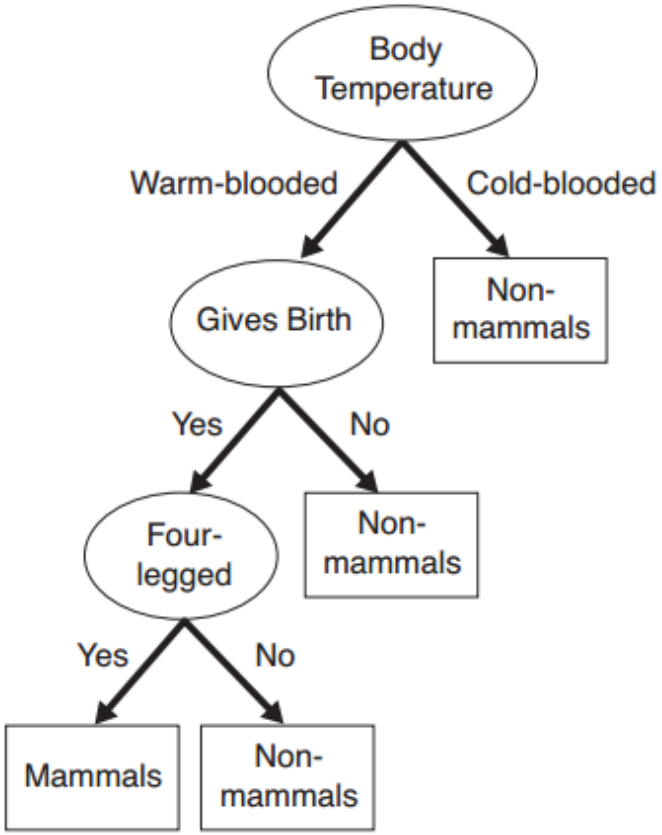
Training

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
human	warm-blooded	yes	no	no	yes
pigeon	warm-blooded	no	no	no	no
elephant	warm-blooded	yes	yes	no	yes
leopard shark	cold-blooded	yes	no	no	no
turtle	cold-blooded	no	yes	no	no
penguin	cold-blooded	no	no	no	no
eel	cold-blooded	no	no	no	no
dolphin	warm-blooded	yes	no	no	yes
spiny anteater	warm-blooded	no	yes	yes	yes
gila monster	cold-blooded	no	yes	yes	no

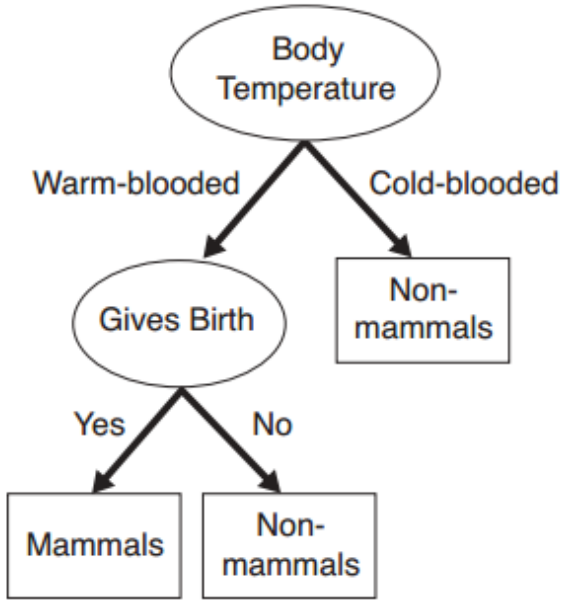
Test

Overfitting Due to the Presence of Noise

- A decision tree (M1) perfectly fits the training data. However, its error rate on the test set is 30%.



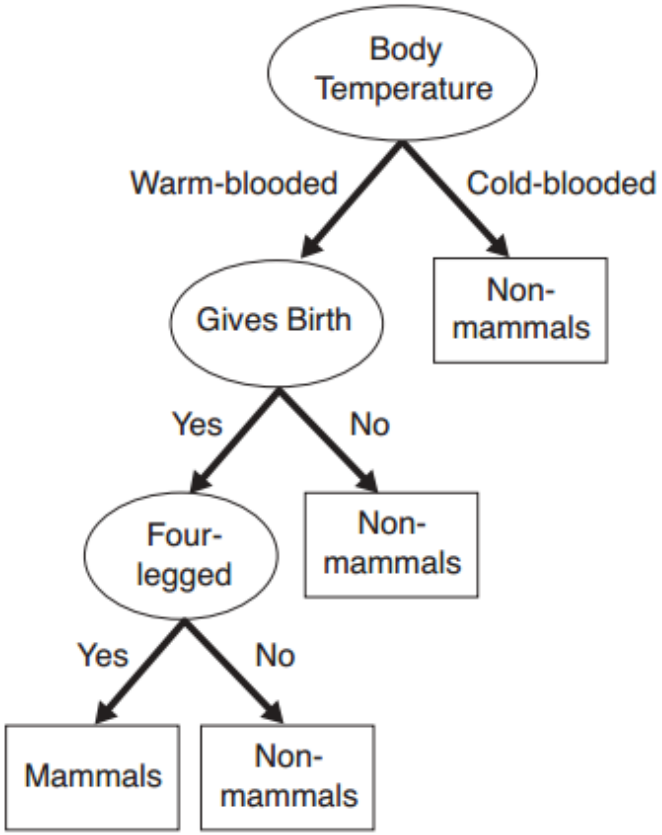
(a) Model M1



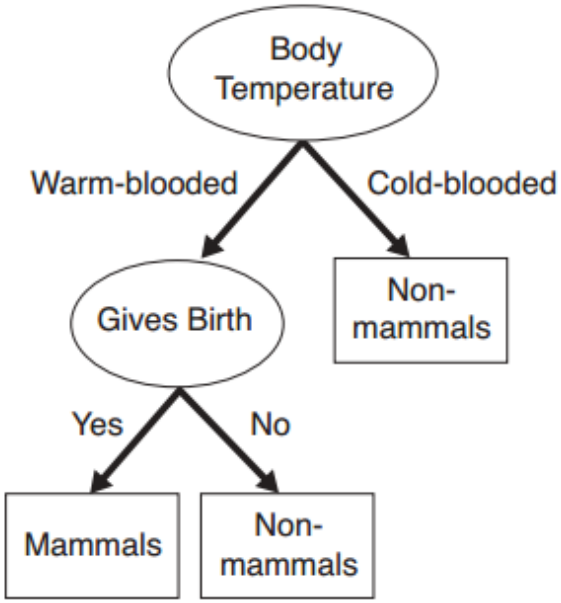
(b) Model M2

Overfitting Due to the Presence of Noise

- A decision tree (M2) has a lower test error rate (10%) even though its training error rate is somewhat higher 20%.



(a) Model M1



(b) Model M2

Overfitting Due to Lack of Representative Samples

- Models that make their classification decisions based on a small number of training records are also susceptible to overfitting.
- Consider the following five training records, which are labeled correctly.

An example training and test set for classifying mammals.

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
salamander	cold-blooded	no	yes	yes	no
guppy	cold-blooded	yes	no	no	no
eagle	warm-blooded	no	no	no	no
poorwill	warm-blooded	no	no	yes	no
platypus	warm-blooded	no	yes	yes	yes

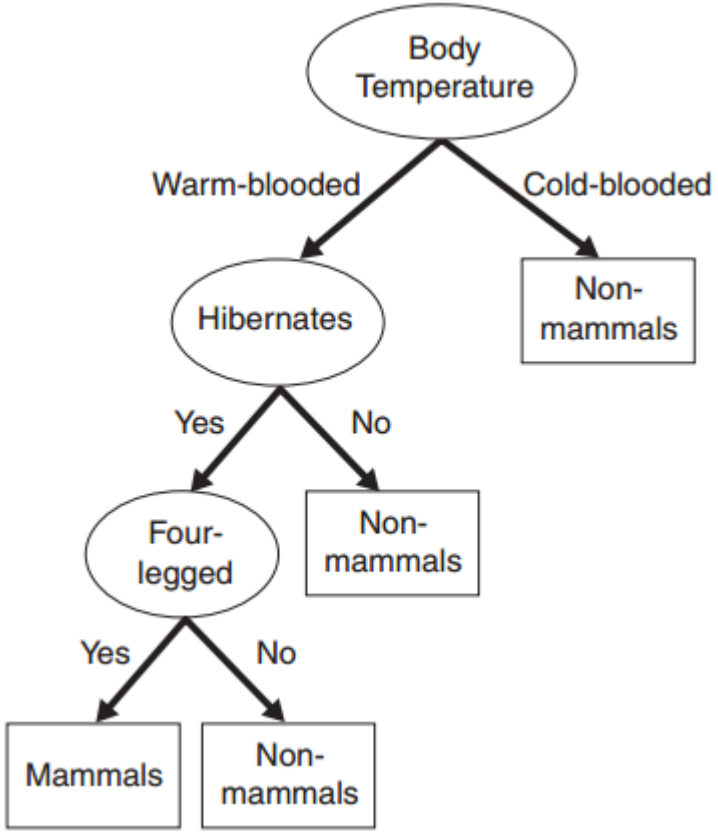
Training

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
human	warm-blooded	yes	no	no	yes
pigeon	warm-blooded	no	no	no	no
elephant	warm-blooded	yes	yes	no	yes
leopard shark	cold-blooded	yes	no	no	no
turtle	cold-blooded	no	yes	no	no
penguin	cold-blooded	no	no	no	no
eel	cold-blooded	no	no	no	no
dolphin	warm-blooded	yes	no	no	yes
spiny anteater	warm-blooded	no	yes	yes	yes
gila monster	cold-blooded	no	yes	yes	no

Test

Overfitting Due to Lack of Representative Samples

- Decision tree induced from the dataset.
- Although the training error is zero, the test error is 30%.



Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
human	warm-blooded	yes	no	no	yes
pigeon	warm-blooded	no	no	no	no
elephant	warm-blooded	yes	yes	no	yes
leopard shark	cold-blooded	yes	no	no	no
turtle	cold-blooded	no	yes	no	no
penguin	cold-blooded	no	no	no	no
eel	cold-blooded	no	no	no	no
dolphin	warm-blooded	yes	no	no	yes
spiny anteater	warm-blooded	no	yes	yes	yes
gila monster	cold-blooded	no	yes	yes	no

Test

Two Strategies to Avoid the Overfitting

- **Pre-Pruning:**
  - The tree growing algorithm is halted before generating a fully grown tree that perfectly fits all training data
- **Post-Pruning:**
  - Grow a decision tree that correctly classifies all training data
  - Simplify it later by replacing some nodes with leafs

Post-pruning preferred in practice—pre-pruning can "stop early"

Pre-pruning (Early Stopping)

- Typical stopping conditions for a node:
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same

Pre-pruning (Early Stopping)

- Typical stopping conditions for a node:
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same

- More restrictive conditions:
  - Stop if the number of instances is less than some threshold
  - Stop if class distribution of instances are independent of the available attributes (i.e., there is no *statistically significant* association between any attribute and the class at a node)
    - Based on statistical significance test (e.g., χ^2 test)
    - **ID3** used χ^2 test in addition to information gain
      - Only statistically significant attributes were allowed to be selected by information gain procedure.
  - Stop if expanding the current node, the observed gain in impurity measure falls below a certain threshold.

ID3 Uses χ^2 Test

- Classic problem: XOR/Parity-problem
  - No individual attribute exhibits any significant association to the class

	a	b	class
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

→ In a dataset that contains XOR attributes a and b, and several irrelevant attributes, **ID3** cannot distinguish between relevant and irrelevant attributes.

→ Such pre-pruning will not expand the root node

ID3 Uses χ^2 Test

- Classic problem: XOR/Parity-problem
  - No individual attribute exhibits any significant association to the class

	a	b	class
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

→ In a dataset that contains XOR attributes a and b, and several irrelevant attributes, **ID3** cannot distinguish between relevant and irrelevant attributes.

→ Such pre-pruning will not expand the root node

- But:
  - XOR-type problems rare in practice
  - Pre-pruning is faster than post-pruning

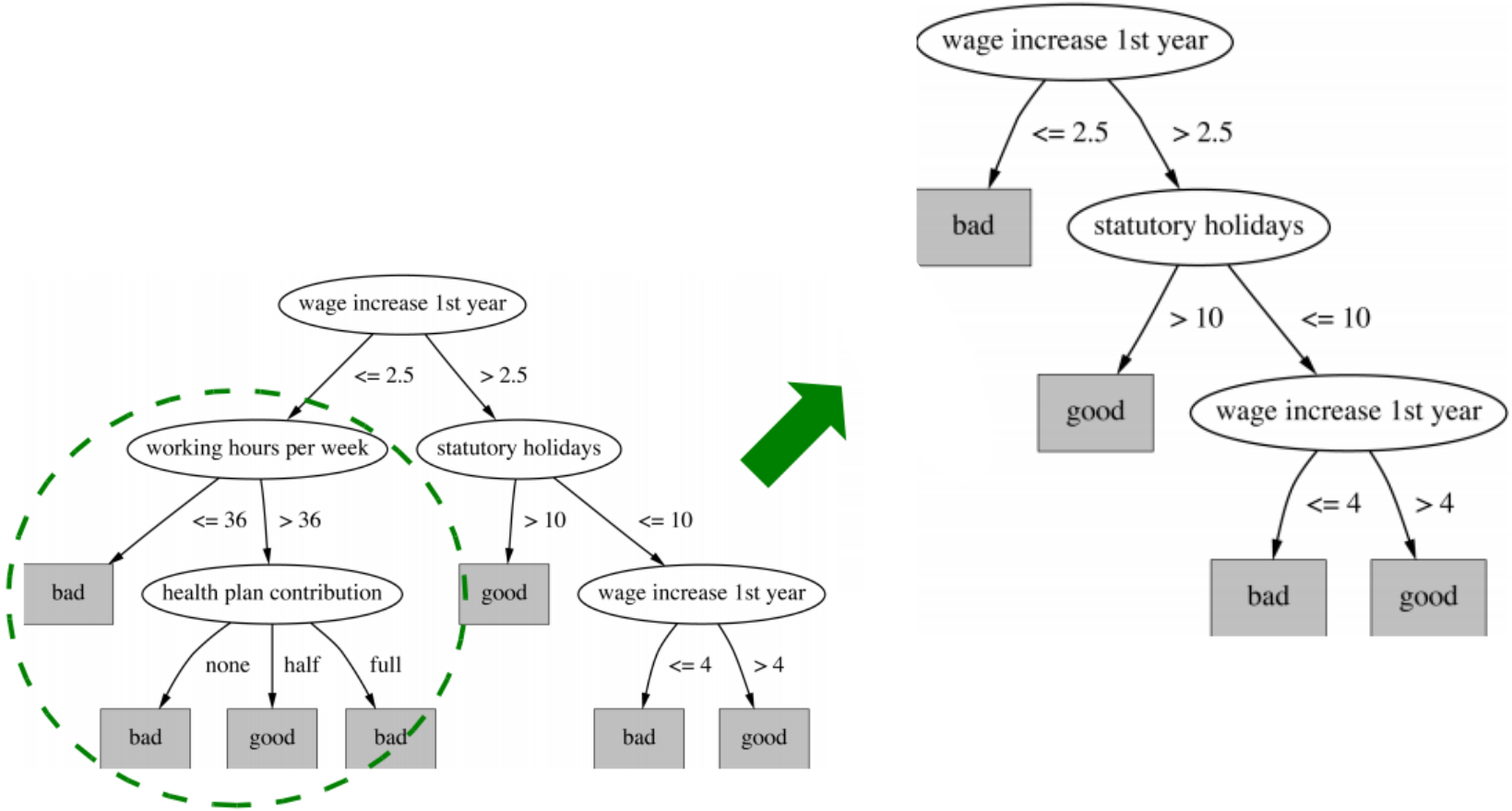
Post-pruning

- Basic idea
  - first grow a full tree to capture all possible attribute interactions
  - later trim the fully grown tree in a bottom-up fashion

1. If generalization error improves after trimming, replace subtree by a leaf node.
2. Class label of leaf node is determined from majority class of instances in the subtree.

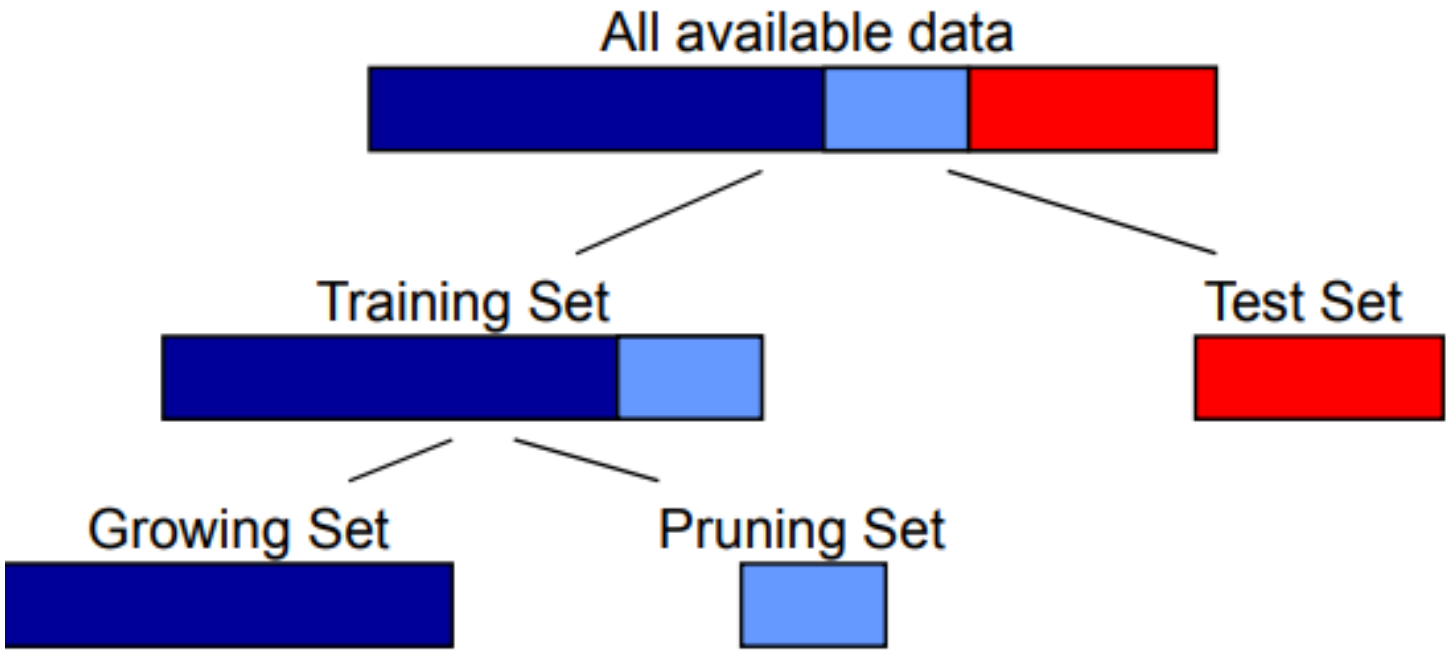
Subtree Replacement

- Replace a subtree with a new leaf node whose class label is determined by the majority class of records with the subtree.



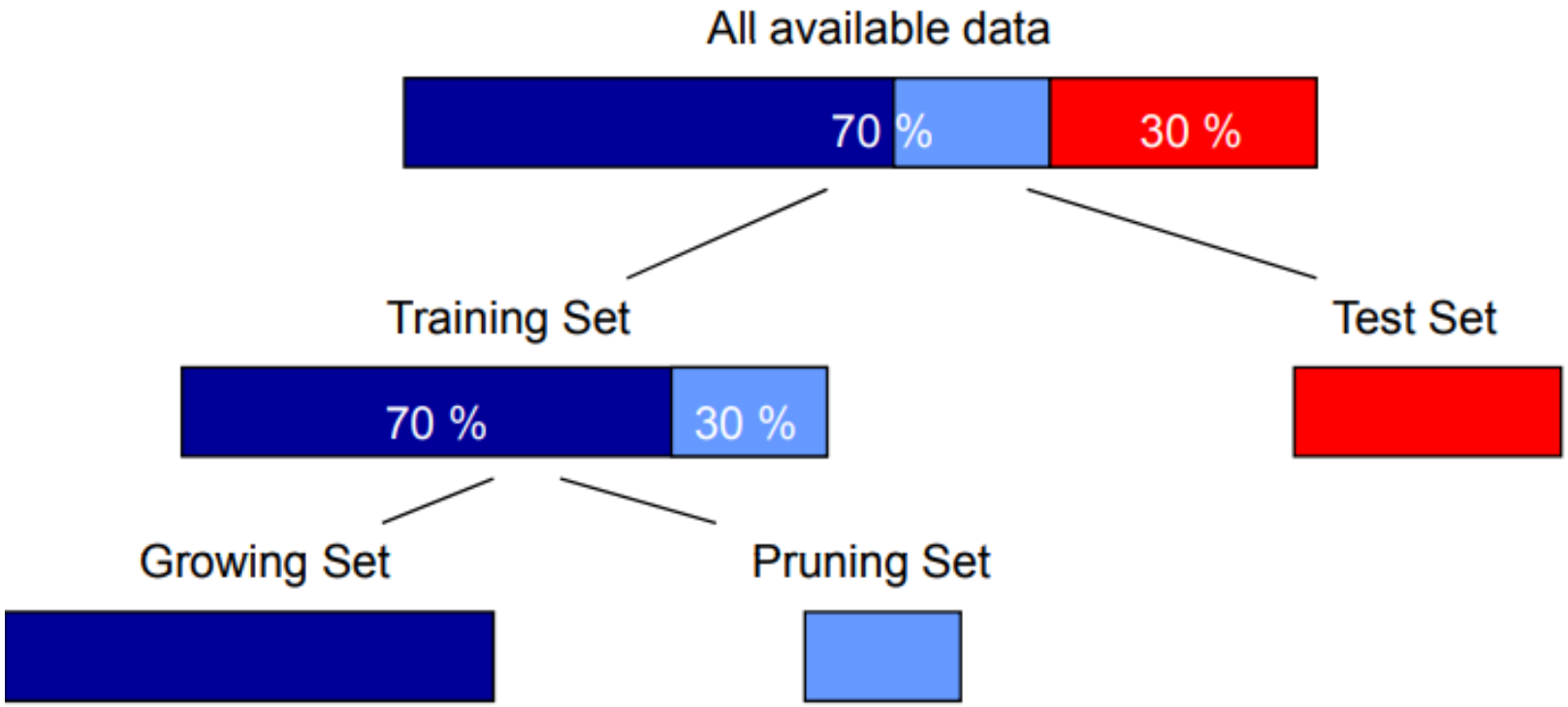
Partitioning Data in Tree Induction

- Estimating accuracy of a tree on new data: “Test Set”
- Some post pruning methods need an independent data set: “Pruning Set”

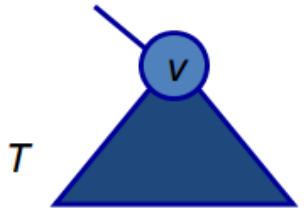


Partitioning Data in Tree Induction

- Problem with using “Pruning Set”: less data for “Growing Set”



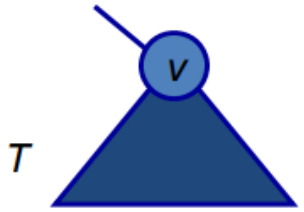
Reduced Error Pruning (REP)



- Use pruning set to estimate accuracy of subtrees and each nodes
- Let T be a subtree rooted at node v. We define:

  Gain from pruning at v = #misclassification in T - #misclassification at v

Reduced Error Pruning (REP)

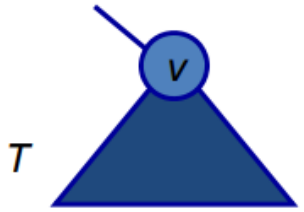


- Use pruning set to estimate accuracy of subtrees and each nodes
- Let T be a subtree rooted at node v. We define:

  Gain from pruning at v = #misclassification in T - #misclassification at v

1. Split training data into a growing and a pruning set
2. Learn a full decision tree
3. As long as the error on the pruning set does not increase
   - try to replace each node v by a leaf (assign the majority class)
   - evaluate the resulting (sub-)tree on the pruning set
   - make the replacement that results in the maximum error reduction (maximum gain from pruning at v)
4. Return the resulting decision tree

Reduced Error Pruning (REP)



- Use pruning set to estimate accuracy of subtrees and each nodes
- Let T be a subtree rooted at node v. We define:

  Gain from pruning at v = #misclassification in T - #misclassification at v

1. Split training data into a growing and a pruning set
2. Learn a full decision tree
3. As long as the error on the pruning set does not increase
   - try to replace each node v by a leaf (assign the majority class)
   - evaluate the resulting (sub-)tree on the pruning set
   - make the replacement that results in the maximum error reduction (maximum gain from pruning at v)
4. Return the resulting decision tree

Note: "Bottom-up restriction": T can only be pruned if it does not contain a sub-tree with lower error than T

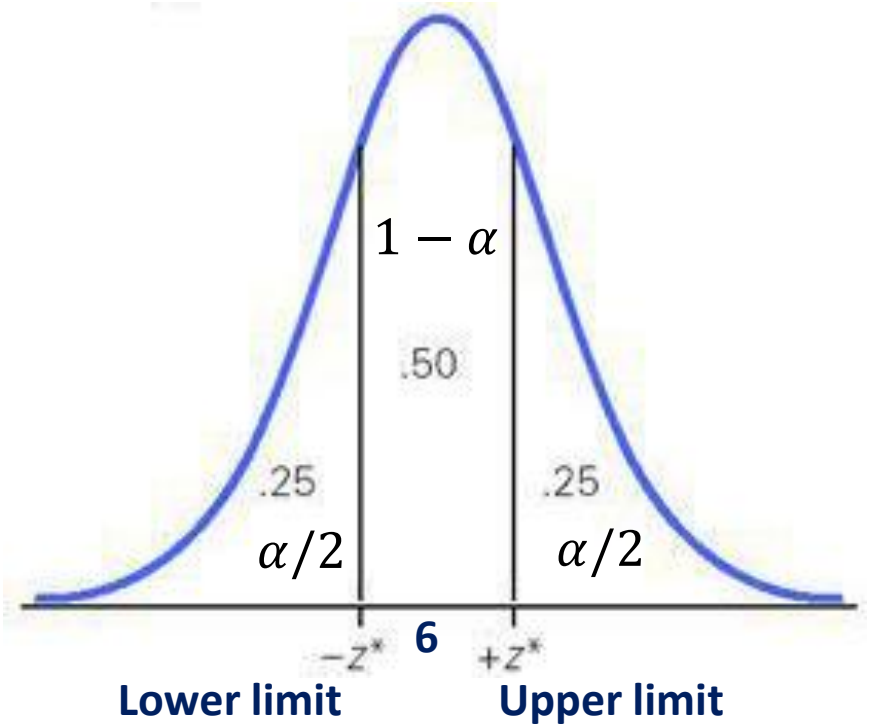
Pessimistic Error Pruning

- Error on the training data is NOT a useful estimator
- Use hold-out (pruning) set for pruning
  - ("reduced-error pruning")
- C4.5's method
  - Avoids the need of pruning set or cross validation
  - Derive confidence interval from training data
    - Use a user-specified confidence level
  - Use a heuristic limit for pruning
    - Assume that the true error is on the upper bound of this confidence interval (pessimistic error estimate)

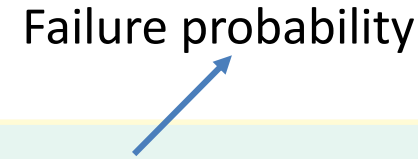
Pessimistic Error Rates

- Consider classifying S examples incorrectly out of N examples as observing S events in N trials in the binomial distribution.
- For a given confidence level $100(1-\alpha)$ (α is the significant level), we aim to find the upper limit on the error rate.

- Example:
  - 100 examples in a leaf
  - 6 examples misclassified
- How large is the true error assuming the pessimistic estimate with a confidence of 50%?



Pessimistic Error Rates



- Mean and variance for a Bernoulli trial: $e, e(1-e)$
- Mean and variance for the error rate $f = S/N$: $e, e(1-e)/N$

Pessimistic Error Rates

Failure probability

WHY?

- Mean and variance for a Bernoulli trial: $e, e(1-e)$
- Mean and variance for the error rate $f = S/N$: $e, e(1-e)/N$

Pessimistic Error Rates

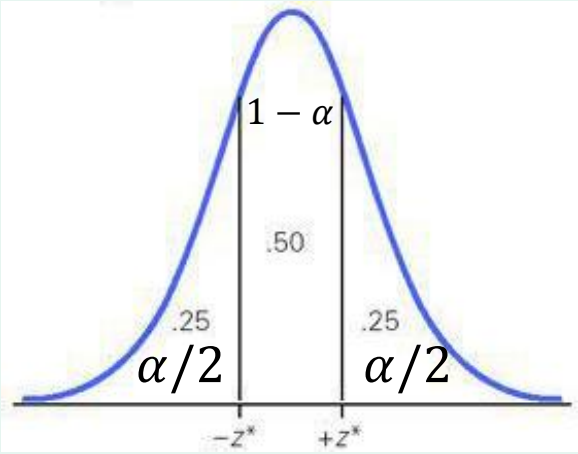
Failure probability

- Mean and variance for a Bernoulli trial: $e, e(1-e)$
- Mean and variance for the error rate $f = S/N$: $e, e(1-e)/N$
- For large enough N, f follows a Normal distribution
- For confidence level $100(1-\alpha)$, the confidence interval $[-z \leq x \leq z]$ for random variable with 0 mean is given by:

$$Pr[-z \leq x \leq z] = 1 - \alpha$$

z is the upper bound

$-z$ is the lower bound

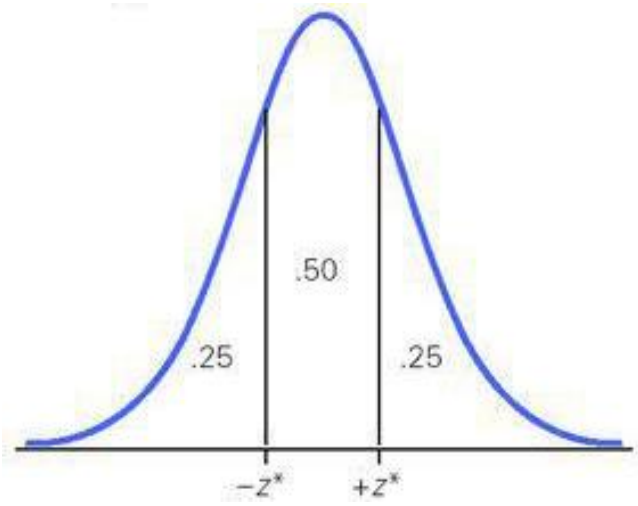


- With a symmetric distribution:

$$Pr[-z \leq x \leq z] = 1 - 2Pr[x \geq z] = 1 - 2Pr[x \leq -z]$$

Pessimistic Error Rates

- Confidence limits for the normal distribution with mean 0 and a variance of 1:

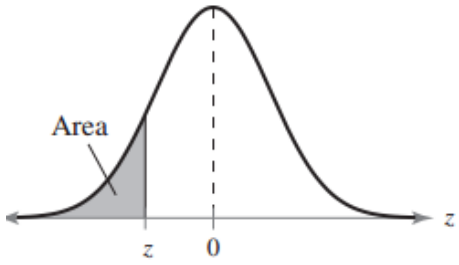


Pr[X ≥ z]	z
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
25%	0.69
40%	0.25

- Thus: $Pr[-0.69 \leq x \leq 0.69] = 50\%$

- To use this we have to reduce our random variable f to have 0 mean and unit variance.

Pessimistic Error Rates



Pr[$X \geq z$]	z
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
25%	0.69
40%	0.25

z	.09	.08	.07	.06	.05	.04	.03	.02	.01	.00
–3.4	.0002	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0003
–3.3	.0003	.0004	.0004	.0004	.0004	.0004	.0004	.0005	.0005	.0005
–3.2	.0005	.0005	.0005	.0006	.0006	.0006	.0006	.0006	.0007	.0007
–3.1	.0007	.0007	.0008	.0008	.0008	.0008	.0009	.0009	.0009	.0010
–3.0	.0010	.0010	.0011	.0011	.0011	.0012	.0012	.0013	.0013	.0013
–2.9	.0014	.0014	.0015	.0015	.0016	.0016	.0017	.0017	.0018	.0019
–2.8	.0019	.0020	.0021	.0021	.0022	.0023	.0023	.0024	.0025	.0026
–2.7	.0026	.0027	.0028	.0029	.0030	.0031	.0032	.0033	.0034	.0035
–2.6	.0036	.0037	.0038	.0039	.0040	.0041	.0043	.0044	.0045	.0047
–2.5	.0048	.0049	.0051	.0052	.0054	.0055	.0057	.0059	.0060	.0062
–2.4	.0064	.0066	.0068	.0069	.0071	.0073	.0075	.0078	.0080	.0082
–2.3	.0084	.0087	.0089	.0091	.0094	.0096	.0099	.0102	.0104	.0107
–2.2	.0110	.0113	.0116	.0119	.0122	.0125	.0129	.0132	.0136	.0139
–2.1	.0143	.0146	.0150	.0154	.0158	.0162	.0166	.0170	.0174	.0179
–2.0	.0183	.0188	.0192	.0197	.0202	.0207	.0212	.0217	.0222	.0228
–1.9	.0233	.0239	.0244	.0250	.0256	.0262	.0268	.0274	.0281	.0287
–1.8	.0294	.0301	.0307	.0314	.0322	.0329	.0336	.0344	.0352	.0359
–1.7	.0367	.0375	.0384	.0392	.0401	.0409	.0418	.0427	.0436	.0446
–1.6	.0455	.0465	.0475	.0485	.0495	.0505	.0516	.0526	.0537	.0548
–1.5	.0559	.0571	.0582	.0594	.0606	.0618	.0630	.0643	.0655	.0668
–1.4	.0681	.0694	.0708	.0722	.0735	.0749	.0764	.0778	.0793	.0808
–1.3	.0823	.0838	.0853	.0869	.0885	.0901	.0918	.0934	.0951	.0968
–1.2	.0985	.1003	.1020	.1038	.1056	.1075	.1093	.1112	.1131	.1151
–1.1	.1170	.1190	.1210	.1230	.1251	.1271	.1292	.1314	.1335	.1357
–1.0	.1379	.1401	.1423	.1446	.1469	.1492	.1515	.1539	.1562	.1587
–0.9	.1611	.1635	.1660	.1685	.1711	.1736	.1762	.1788	.1814	.1841
–0.8	.1867	.1894	.1922	.1949	.1977	.2005	.2033	.2061	.2090	.2119
–0.7	.2148	.2177	.2206	.2236	.2266	.2296	.2327	.2358	.2389	.2420
–0.6	.2451	.2483	.2514	.2546	.2578	.2611	.2643	.2676	.2709	.2743
–0.5	.2776	.2810	.2843	.2877	.2912	.2946	.2981	.3015	.3050	.3085
–0.4	.3121	.3156	.3192	.3228	.3264	.3300	.3336	.3372	.3409	.3446
–0.3	.3483	.3520	.3557	.3594	.3632	.3669	.3707	.3745	.3783	.3821
–0.2	.3859	.3897	.3936	.3974	.4013	.4052	.4090	.4129	.4168	.4207
–0.1	.4247	.4286	.4325	.4364	.4404	.4443	.4483	.4522	.4562	.4602
–0.0	.4641	.4681	.4721	.4761	.4801	.4840	.4880	.4920	.4960	.5000

Pessimistic Error Rates

- Transformed value for f: $\frac{f-e}{\sqrt{e(1-e)/N}}$ (i.e., subtract the mean and divide by the standard deviation)
- Let $Pr[x \leq -z] = Pr[x \geq z] = c$

$$Pr\left[-z \leq \frac{f-e}{\sqrt{e(1-e)/N}} \leq z\right] = 1-2c$$

- Solving $\frac{f-e}{\sqrt{e(1-e)/N}} = z$ for e:

$$x = \frac{-b \pm \sqrt{\Delta}}{2a} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$$

$$e = \frac{f + \frac{z^2}{2N} \pm z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}}$$

C4.5

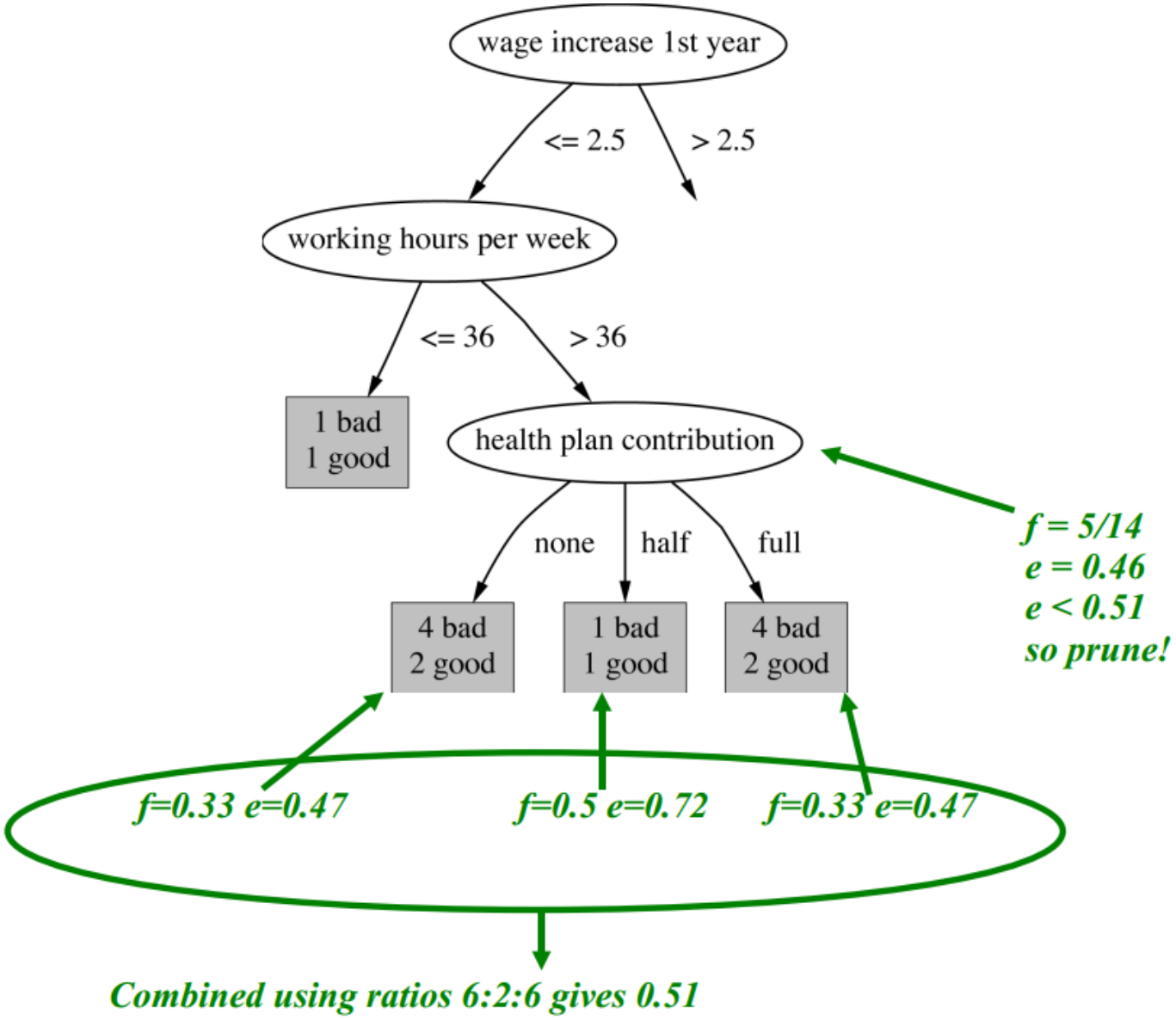
- Pessimistic error estimate for a node

$$e = \frac{f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}}$$

  - z is derived from the desired confidence value
    - If $c = 25\%$ then $z = 0.69$ (from normal distribution)
  - f is the error on the training data
  - N is the number of instances covered by the leaf

- Error estimate for subtree is weighted sum of error estimates or all its leaves.

Example


wage increase 1st year
<= 2.5
> 2.5
working hours per week
<= 36
> 36
1 bad
1 good
health plan contribution
none
half
full
4 bad
2 good
1 bad
1 good
4 bad
2 good
f = 5/14
e = 0.46
e < 0.51
so prune!
f=0.33 e=0.47
f=0.5 e=0.72
f=0.33 e=0.47
Combined using ratios 6:2:6 gives 0.51

C4.5：Choices And Options

C4.5 has several parameters

- -c confidence value (default 25%)
  - Lower values incur heavier pruning
- -m In all tests, at least two branches must contain a minimum number of objects (default 2)

Sample Experimental Evaluation

Typical behavior with growing m and decreasing c:

- Tree size and training accuracy always decrease
- Predictive accuracy first increases then decreases
- Ideal value *on this data* set near
  - $m = 30$
  - $c = 10$

Parameters	*Tree Size*	*Purity*	*Predictive Accuracy*
No Pruning (`C4.5 -m1`)	547	99.7%	60.3% (± 4.8)
`C4.5 -m2`	314	91.8%	60.1% (± 3.3)
`C4.5 -m5`	170	82.3%	60.4% (± 5.7)
`C4.5 -m10`	90	76.6%	60.0% (± 5.2)
`C4.5 -m15`	62	74.1%	61.6% (± 4.7)
`C4.5 -m20`	47	71.9%	62.7% (± 2.0)
`C4.5 -m25`	37	71.3%	63.0% (± 2.2)
`C4.5 -m30`	26	70.1%	65.1% (± 2.5)
`C4.5 -m35`	22	69.9%	65.0% (± 4.2)
`C4.5 -m40`	20	69.2%	64.8% (± 2.6)
`C4.5 -m50`	24	69.1%	64.5% (± 3.5)
`C4.5 -c75`	524	99.7%	61.0% (± 4.5)
`C4.5 -c50`	357	95.3%	60.2% (± 3.6)
`C4.5 -c25`	257	91.2%	62.3% (± 4.4)
`C4.5 -c15`	137	81.8%	64.8% (± 4.6)
`C4.5 -c10`	75	76.9%	65.9% (± 4.9)
`C4.5 -c5`	53	74.7%	63.8% (± 6.0)
`C4.5 -c1`	27	70.2%	63.4% (± 5.8)
`C4.5` Default	173	86.2%	62.5% (± 5.2)
`C4.5 -m30 -c10`	20	69.6%	66.7% (± 3.7)
Mode Prediction	1	56.8%	56.8%

training accuracy

Characteristics of Decision Tree

Characteristics of Decision Tree

1. Nonparametric approach for building classification models.
   - No prior assumptions regarding the type of probability distributions satisfied by the class and attributes.
2. Heuristic-based approach to guide the search in the vast hypothesis space.
3. Computationally inexpensive, good for the large dataset.
   - Once a decision tree has been built, classifying a test record is extremely fast, with a worst-case complexity of $O(w)$, w is the maximum depth of the tree.
4. Easy to interpret, especially smaller ones.
5. Provide an expressive representation for learning discrete-valued functions.

Characteristics of Decision Tree

6. Quite robust to the presence of noise (with pruning).
7. Robust to the redundant attributes.
   - An attributes is *redundant* if it is strongly correlated with another attribute in the data.
   - One of the two redundant attributes will not be used for splitting once the other one has been chosen.
8. Not robust to the irrelevant attributes, i.e., attributes that are not useful for the classification task.
   - The irrelevant attributes may be accidently chosen during the tree-growing process, which results in the overfitting.
   - Feature selection techniques can help by eliminating the irrelevant attributes during preprocessing.

Characteristics of Decision Tree

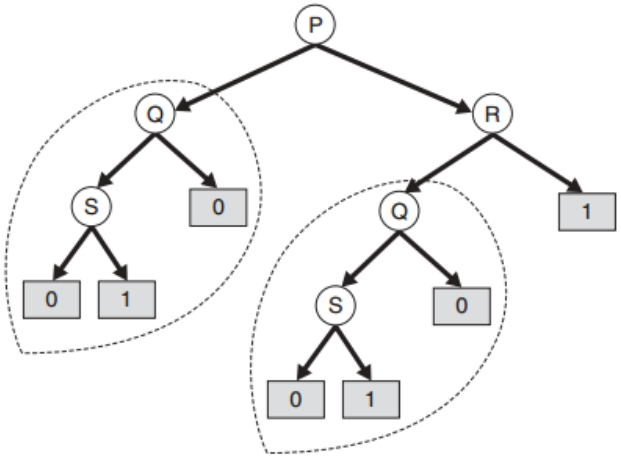
9. **Data fragmentation** (remember the high-branching attribute)
   - The recursive partition approach → the number of records becomes smaller as we traverse down the tree.
   - The number of records of leaf nodes maybe too small to make a statistically significant decision about the class.
   - **Solution**: disallow further splitting when the number of records falls below a certain threshold.

10. Different criteria for attribute selection rarely make a large difference.
11. Different pruning methods mainly change the size of the resulting pruned tree.

Characteristics of Decision Tree

12. Subtree Replication

- A subtree can be replicated multiple times in a decision tree.
- Due to that decision tree replies on a single attribute test condition at each internal node.
- The same test condition can be applied to different parts of the attribute space (due to the divide-and-conquer strategy).

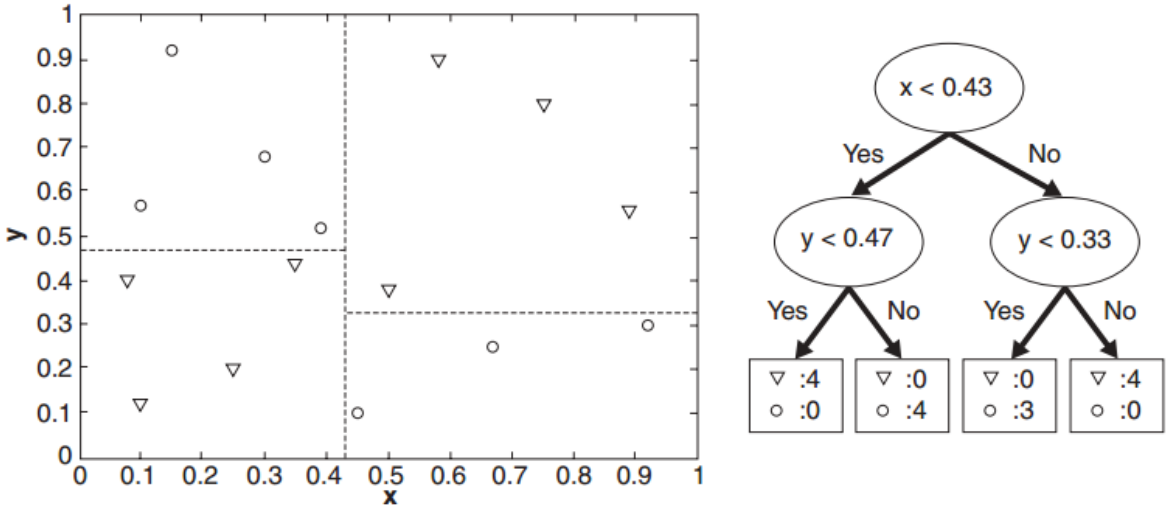


Tree replication problem. The same subtree can appear at different branches.

Characteristics of Decision Tree

13. Decision Boundary

- The decision tree partitions the attribute space into disjoint regions until each region contains records of the same class.
- The border between two neighboring regions of different class is known as a *decision boundary.*
- The decision boundaries of the univariate split are rectilinear; i.e., parallel to the *"coordinate axes".*

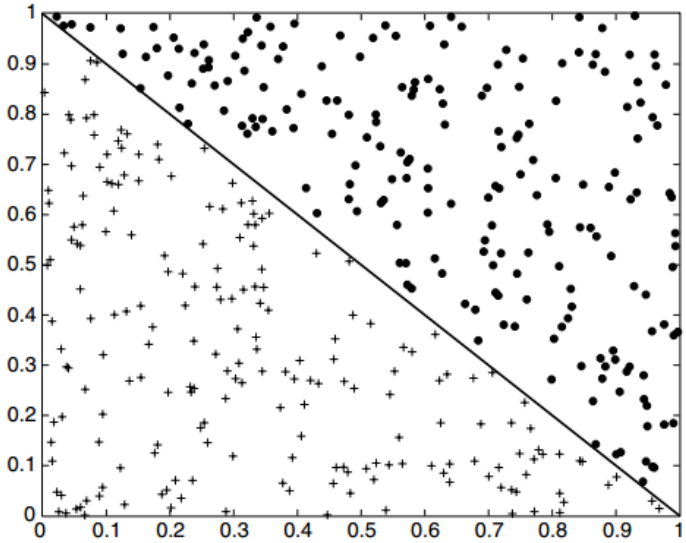


Example of a decision tree and its decision boundaries for a 2-D dataset.

Characteristics of Decision Tree

13. Decision Boundary

- Univariate Split
- Limits the expressiveness of the decision tree for modeling complex relationships among continuous attributes.



Example of dataset that cannot be partitioned optimally using test conditions involving single attributes.

Characteristics of Decision Tree

13. Decision Boundary

- Multivariate Split
  - Several attributes may participate in a single node split.
  - Finding the best multivariate criteria is more complicated than the univariate split.
  - Although the multivariate split may improve the tree's performance dramatically , it is less popular than the univariate split.
  - Mainly based on the linear combination of the input attributes.
  - C4.5 builds univariate decision trees
  - CART can build multivariate trees

Regression Trees

Differences to Decision Trees (Classification Trees)

- Leaf Nodes:

Regression Trees

Differences to Decision Trees (Classification Trees)

- Leaf Nodes:
  - Predict the average value of all instances in this leaf

Regression Trees

Differences to Decision Trees (Classification Trees)

- Leaf Nodes:
  - Predict the average value of all instances in this leaf
- Splitting criterion:

Regression Trees

Differences to Decision Trees (Classification Trees)

- Leaf Nodes:
  - Predict the average value of all instances in this leaf
- Splitting criterion:
  - Minimize the variance of the values in each subset S_i
  - Standard deviation reduction

$$SDR(A,S) = SD(S) - \sum_i \frac{|S_i|}{|S|} SD(S_i)$$

Regression Trees

Differences to Decision Trees (Classification Trees)

- Leaf Nodes:
  - Predict the average value of all instances in this leaf
- Splitting criterion:
  - Minimize the variance of the values in each subset S_i
  - Standard deviation reduction

$$SDR(A,S) = SD(S) - \sum_i \frac{|S_i|}{|S|} SD(S_i)$$

- Termination criteria:
  - Very important! (otherwise only single points in each leaf)
  - lower bound on standard deviation in a node
  - lower bound on number of examples in a node

Regression Trees

Differences to Decision Trees (Classification Trees)

- Leaf Nodes:
  - Predict the average value of all instances in this leaf
- Splitting criterion:
  - Minimize the variance of the values in each subset S_i
  - Standard deviation reduction

$$SDR(A,S) = SD(S) - \sum_i \frac{|S_i|}{|S|} SD(S_i)$$

- Termination criteria:
  - Very important! (otherwise only single points in each leaf)
  - lower bound on standard deviation in a node
  - lower bound on number of examples in a node
- Pruning criterion:
  - Numeric error measures, e.g. Mean-Squared Error

Comparison

	Splitting Criteria	Attribute types	Missing value	Pruning strategy
ID3	Information gain	Handles only categorical value	Cannot handle	No pruning
C4.5	Gain ratio	Handles only categorical and numerical value	Handle	Error based pruning
CART	Gini index	Handles only categorical and numerical value	Handle	Cost-complexity pruning

Summary

- **Decision tree induction**
- **Different criteria for attribute selection**
  - ✓ Information gain, Gain ratio, Gini index
- **Deal with missing values**
- **Deal with numerical attributes**
- **Deal with overfitting**
  - ✓ Pre-pruning (early stopping) and post-pruning
  - ✓ Reduced Error Pruning (REP)
  - ✓ Pessimistic Error Estimate
- **Characteristics of Decision tree**
- **Regression tree**