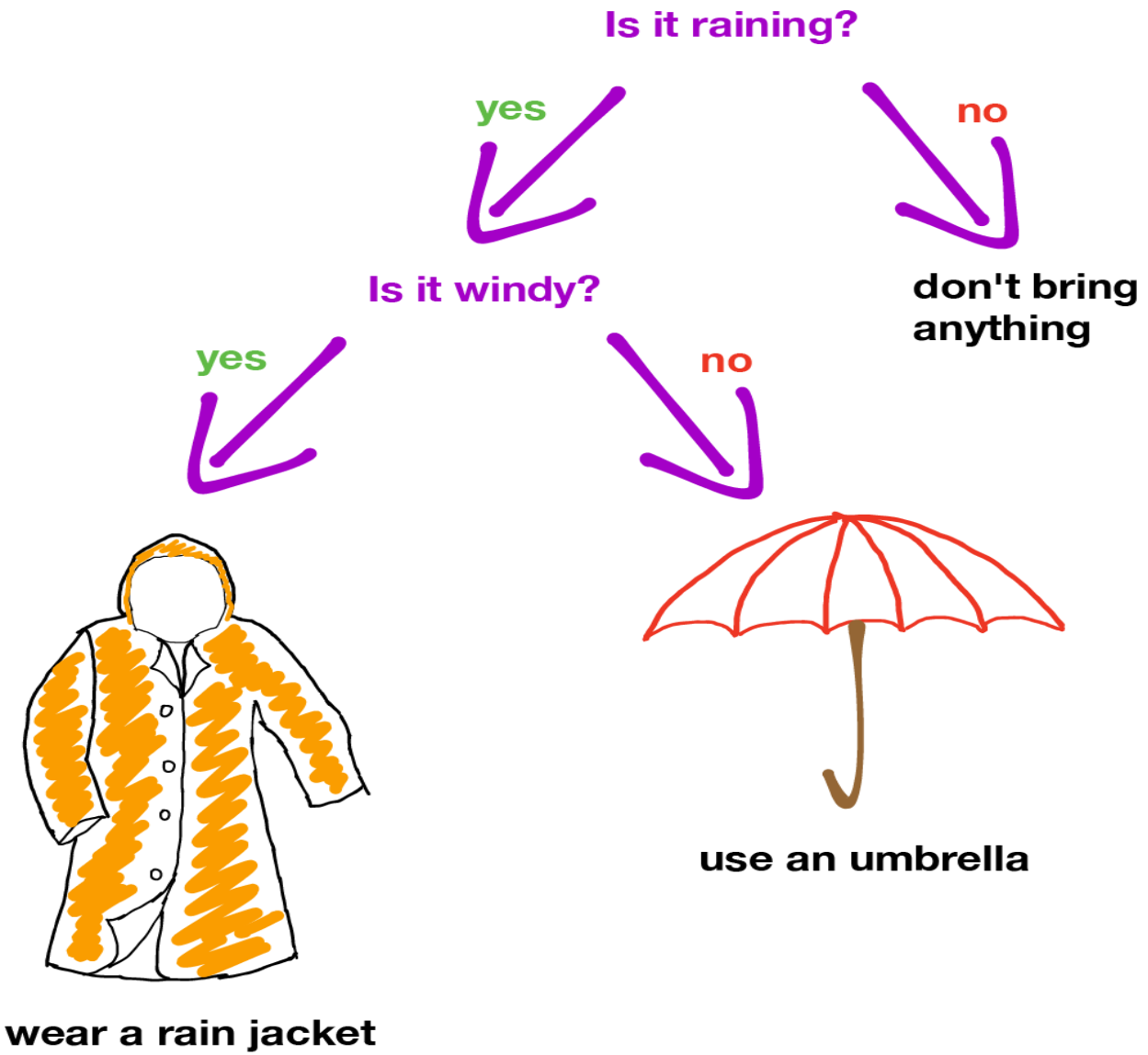
Module 2
Decision Tree Learning

Tejaswini H, Assistant Professor,
Department of Computer Science



Decision Tree

- A decision tree is a flowchart-like structure in which
  - each internal node represents a "test" on an attribute
  - each branch represents the outcome of the test
  - each leaf node represents a class label (decision taken after computing all attributes).
- The paths from root to leaf represent classification rules.

Introduction: Decision Tree Learning

- Decision tree learning is a method for **approximating discrete-valued target functions**, in which the learned function is represented by a **decision tree**.

- Learned trees can also be re-represented as **sets of if-then rules** to improve human readability.

- These learning methods are among the most popular of inductive inference algorithms
- have been successfully applied to a broad range of tasks from learning to diagnose medical cases to learning to assess credit risk of loan applicants.

- It is a method for approximating discrete-valued functions that is robust to noisy data and capable of learning disjunctive expressions.
- widely used algorithms are ID3, ASSISTANT, and **C4.5**
- These decision tree learning methods search a completely expressive hypothesis space and thus avoid the difficulties of restricted hypothesis spaces.
- Their inductive bias is a preference for small trees over large trees.

DECISION TREE REPRESENTATION

- classifies instances by sorting them down the tree from the **root to some leaf node**, which provides the classification of the instance.
- **Each node** in the tree specifies a test of some attribute of the instance
- **branch descending from that node** corresponds to one of the possible values for this attribute

- An instance is classified by
  - **starting at the root node** of the tree,
  - **testing the attribute** specified by this node,
  - then **moving down** the tree branch corresponding to the value of the attribute in the given example.
- This process is then repeated for the subtree rooted at the new node.

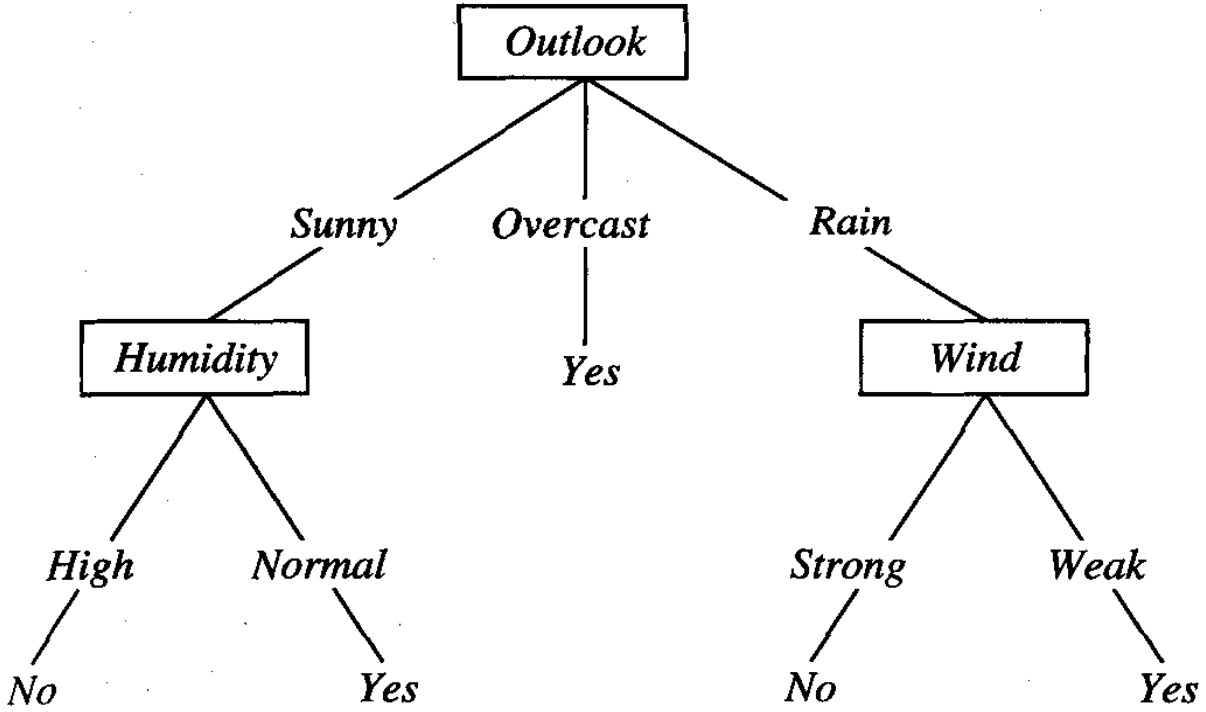


FIGURE 3.1
A decision tree for the concept *PlayTennis*. An example is classified by sorting it through the tree to the appropriate leaf node, then returning the classification associated with this leaf (in this case, *Yes* or *No*). This tree classifies Saturday mornings according to whether or not they are suitable for playing tennis.

- In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.
- Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions.

(Outlook = Sunny ^ Humidity = Normal)

∨ ***(Outlook = Overcast)***

∨ ***(Outlook = Rain ^ Wind = Weak)***

decision tree learning is generally best suited to problems with the following characteristics:

1. *Instances are represented by attribute-value pairs.*

- Instances are described by a fixed set of attributes and their values.
- Attribute can take
  - on a small number of disjoint possible values
  - real-values

2. The target function has discrete output values.

- The decision tree generally assigns a boolean classification (e.g., ***yes*** or ***no)*** to each example.
- Can have more than two possible output values
- Also real-valued outputs (though the application of decision trees in this setting is less common)

3. Disjunctive descriptions may be required.

- As noted above, decision trees naturally represent disjunctive expressions.

4. The training data may contain errors.

- Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples.

5. The training data may contain missing attribute values

- Decision tree methods can be used even when some training examples have unknown values
- (e.g., if the ***Humidity*** of the day is known for only some of the training examples)

- ***classification problems:***
  - Problems in which the task is to classify examples into one of a discrete set of possible categories, are often referred to as ***classification problems.***
- Decision tree learning has therefore been applied to problems such as
  - learning medical patients by their disease
  - equipment malfunctions by their cause
  - loan applicants by their likelihood of defaulting on payments

THE BASIC DECISION TREE LEARNING ALGORITHM

- ID3 algorithm
  - **Iterative Dichotomiser 3**
  - algorithm invented by Ross Quinlan

ID3(*Examples*, *Target_attribute*, *Attributes*)

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = −
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
  - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
  - The decision attribute for *Root* $\leftarrow A$
  - For each possible value, v_i, of A,
    - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
    - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for A
    - If $Examples_{v_i}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
      - Else below this new branch add the subtree
        ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* − {A}))
- End
- Return *Root*

* The best attribute is the one with highest *information gain*, as defined in Equation (3.4).

Training examples for the target concept *PlayTennis*

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

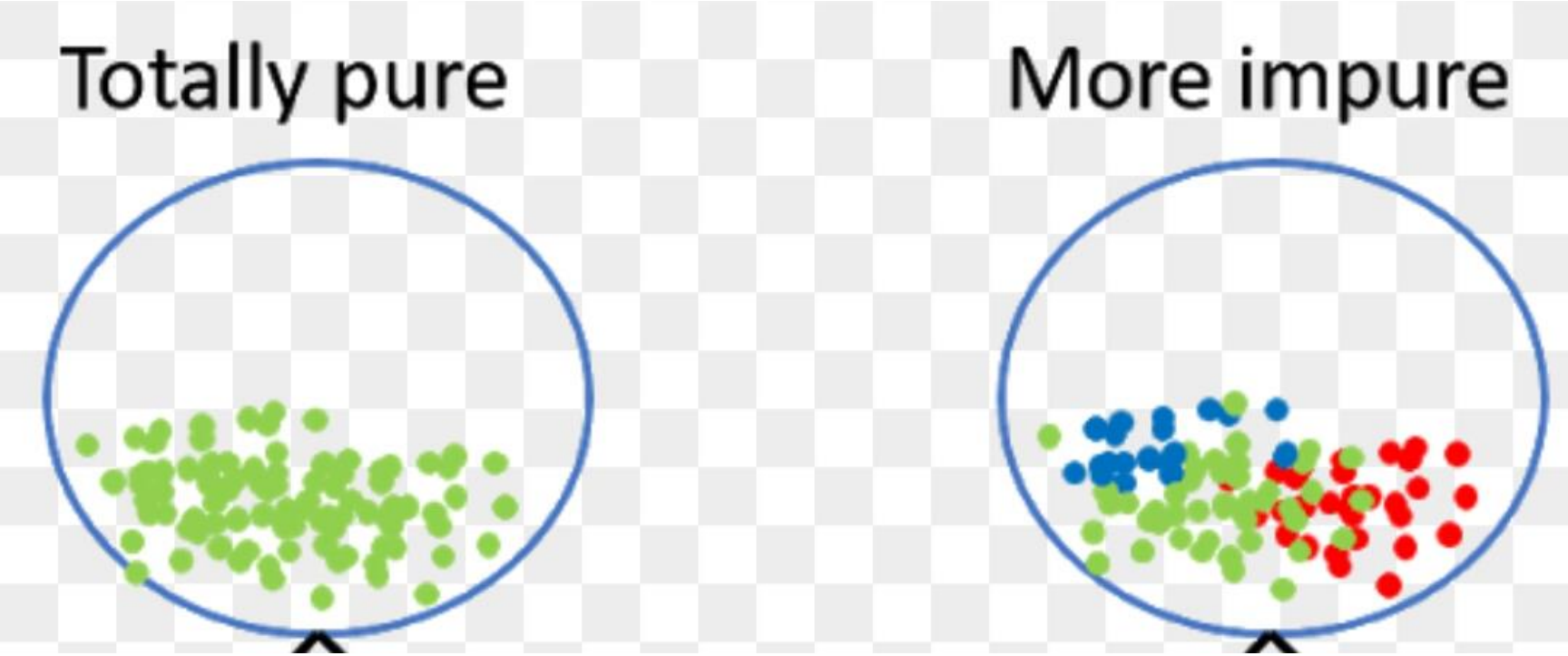
- ID3 algorithm learns decision trees by constructing them **top-down**, beginning with the question "which attribute should be tested at the root of the tree?‘
  - To answer this question, each **instance attribute is evaluated** using a statistical test to determine how well it alone classifies the training examples.
- The **best attribute** is selected and used as the test at the **root node** of the tree.

- A **descendant of the root node** is then created for **each possible value of this attribute**, and the **training examples are sorted to the appropriate descendant node**.
- The **entire process is then repeated** using the training examples **associated with each descendant node** to **select the best attribute** to test at that point in the tree.
- This forms a **greedy search** for an acceptable decision tree, in which the algorithm never backtracks to reconsider earlier choices.

Which Attribute Is the Best Classifier?

- The central choice in the ID3 algorithm is selecting **which attribute to test at each node** in the tree.
- We would like to **select the attribute** that is most useful for classifying examples.

- **ENTROPY:** MEASURES HOMOGENEITY OF EXAMPLES
- **INFORMATION GAIN:** MEASURES THE EXPECTED REDUCTION IN ENTROPY


Totally pure
More impure

ENTROPY

- ***Entropy:*** characterizes the (im)purity of an arbitrary collection of examples.
- Given a collection S, containing positive and negative examples of some target concept, the entropy of S relative to this boolean classification is:

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

- Where
- $\mathbf{p_+}$: is the proportion of positive examples in S
  $\mathbf{p_-}$: is the proportion of negative examples in S
- In all calculations involving entropy we define 0 log 0 to be 0.

- Notice that the entropy is 0 if all members of S belong to the same class.
- For example,
  - if all members are positive ($\mathbf{p_+} = 1$), then $\mathbf{p_-}$ is 0
  - Entropy(S)

  $= -1 \cdot \log_2(1) - 0 \cdot \log_2 0$

  $= -1 \cdot 0 - 0 \cdot \log_2 0$

  $= 0$

- **Entropy is 1**
  - If the collection contains an equal number of positive and negative examples.
- **Entropy is between 0 and 1**
  - If the collection contains unequal numbers of positive and negative examples.
- **Entropy is 0**
  - If the examples belongs to same class.

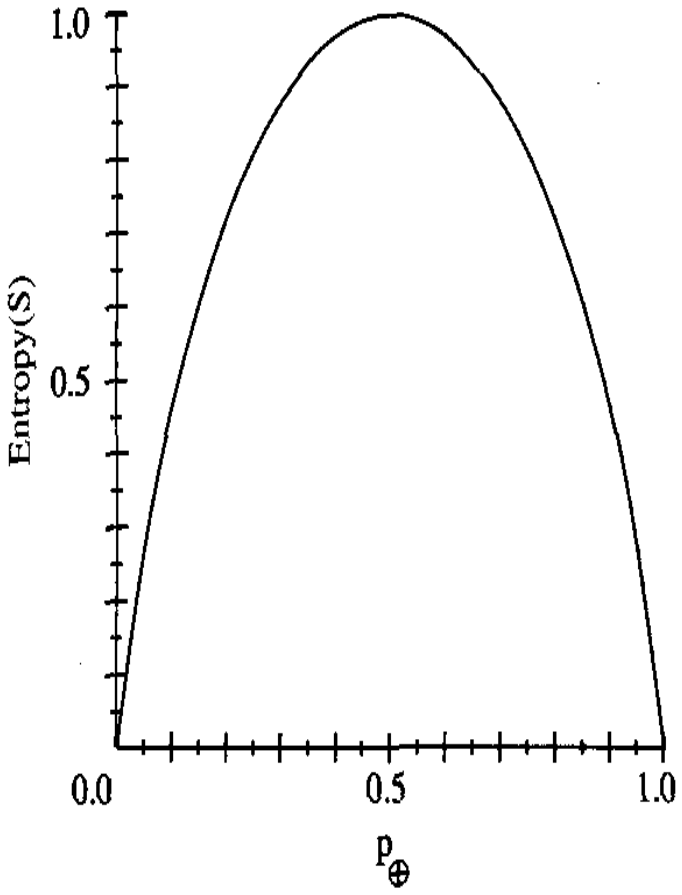


FIGURE 3.2
The entropy function relative to a boolean classification, as the proportion, $p_{\oplus}$, of positive examples varies between 0 and 1.

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

- p_i is the proportion of S belonging to class i

- Let's define a statistical property, called ***information gain***, that measures how well a given attribute separates the training examples according to their target classification.
- ID3 uses this information gain measure to select among the candidate attributes at each step while growing the tree.

INFORMATION GAIN

- measures the effectiveness of an attribute in classifying the training data.
- ***information gain*** is the expected reduction in entropy caused by partitioning the examples according to this attribute.

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- **S*:*** collection of examples
- **Values(A)*:*** set of all possible values for attribute A
- **S_v*:*** subset of S for which attribute **A** has value **v** i.e.,

  $S_v = \{s \in S \mid A(s) = v\}$
- **Entropy(S) *:*** entropy of the original collection ***S***
- second term: the expected value of the entropy after ***S*** is partitioned using attribute ***A.***
- The expected entropy is the sum of the entropies of each subset ***S_v*** weighted by the fraction of examples $|S_v|/|S|$ that belong to ***S_v***

- ***Gain(S, A)*** is therefore the expected reduction in entropy caused by knowing the value of attribute A.
- Put another way, ***Gain(S, A)*** is the information provided about the ***target &action value,*** given the value of some other attribute ***A.***

ID3(*Examples*, *Target_attribute*, *Attributes*)

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = −
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
  - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
  - The decision attribute for *Root* $\leftarrow A$
  - For each possible value, v_i, of A,
    - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
    - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for A
    - If $Examples_{v_i}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
      - Else below this new branch add the subtree
        ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* − {A}))
- End
- Return *Root*

* The best attribute is the one with highest *information gain*, as defined in Equation (3.4).

Training examples for the target concept *PlayTennis*

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Consider:

- **S**: collection of training example
- Days are described by attributes
  - ***Outlook***
  - ***Temperature***
  - ***Humidity***
  - ***Wind***
- Target attribute: ***PlayTennis***

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- Now, |S| = 14 (number of training example)
- Of these 14 examples,
  - **9 are positive (i.e. with target value Yes)**
  - **5 are negative (i.e with target value No)**

So,

S = [9+, 5-]

Entropy

- **Entropy(S) = $-p_+ \log_2 p_+ \; - \; p_- \log_2 p_-$**

- S = [9+, 5-]
- |S| = 14
- p_{yes} = (9/14) (probability that target value is yes)
- p_{no} = (5/14) (probability that target value is no)

Entropy(S) $= - p_{yes} \log_2 p_{yes} - p_{no} \log_2 p_{no}$

Entropy([9+, 5-]) $= - (9/14) \log_2(9/14) - (5/14) \log_2 (5/14)$

$= 0.940$

So, Inititally,

S= [9+, 5-] and

E = 0.940

Information Gain

$$\text{Gain}(S, A) =$$

$$\text{Entropy}(S) - \sum_{V \in \text{Values}(A)} (|S_v| / |S|) \text{Entropy}(S_v)$$

1.1: Attribute A = Wind

- Consider attribute ***Wind***
- ***Value(wind) = {weak, strong}***

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- $|S_{wind=weak}| = 8$
  - Yes = 6
  - No = 2
  - **$S_{wind=weak}$ ← [6+, 2-]**
- $|S_{wind=strong}| = 6$
  - Yes = 3
  - No = 3
  - **$S_{wind=strong}$ ← [3+, 3-]**

- **Entropy($S_{wind=weak}$)**

  = Entropy([6+, 2-])

  = $-(6/8)\log_2(6/8) - (2/8)\log_2(2/8)$

  = 0.81125

- **Entropy($S_{wind=strong}$)**

  = Entropy([3+, 3-])

  = $-(3/6)\log_2(3/6) - (3/6)\log_2(3/6)$

  = 1

Gain(S, Wind)

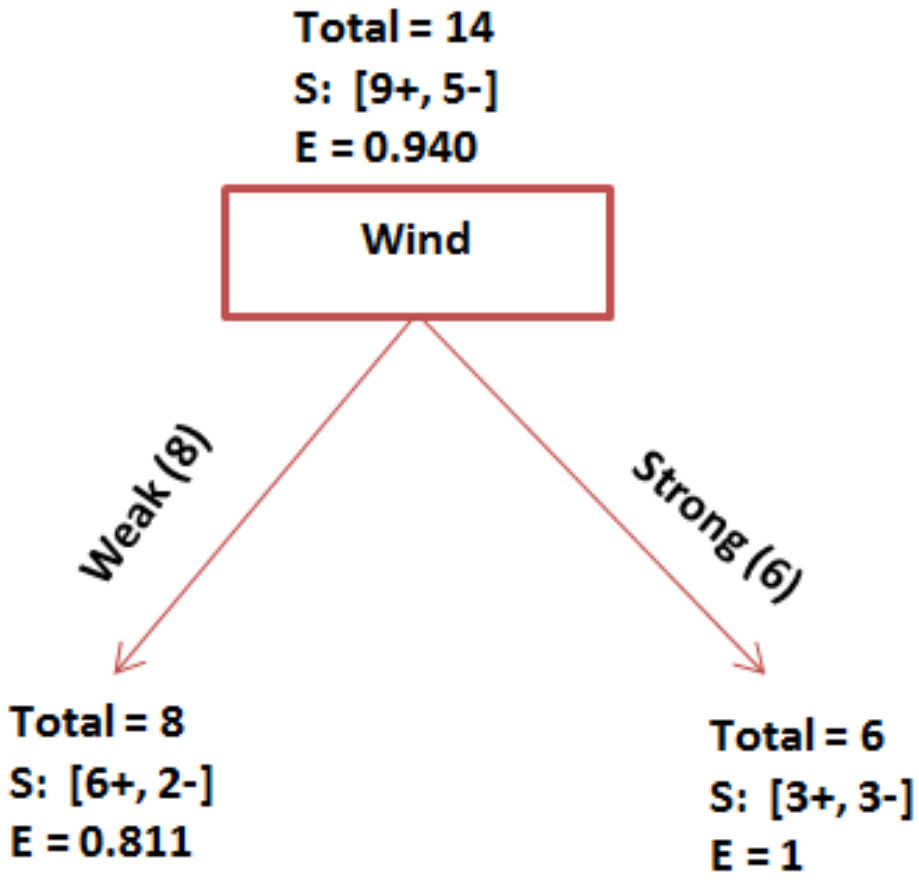
=Entropy(S) –

[($|S_{wind=weak}|$ / $|S|$) Entropy($S_{wind=weak}$) +

($|S_{wind=strong}|$ / $|S|$) Entropy($S_{wind=strong}$)]

= 0.940 - [(8 / 14) * 0.811 + (6 / 14) * 1]

= 0.048



Gain(S, Wind) = 0.048

1.2: Attribute A = Outlook

- Consider attribute ***Outlook***
- ***Value(outlook) = {sunny, rain, overcast}***

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- $|S_{outlook=sunny}| = 5$
  - Yes = 2
  - No = 3
  - **$S_{outlook=sunny}$ ← [2+, 3-]**
- $|S_{outlook=overcast}| = 4$
  - Yes = 4
  - No = 0
  - **$S_{outlook=overcast}$ ← [4+, 0-]**
- $|S_{outlook=rain}| = 5$
  - Yes = 3
  - No = 2
  - **$S_{outlook=rain}$ ← [3+, 2-]**

- **Entropy($S_{\text{outlook=sunny}}$)**
  = Entropy([2+, 3-])
  = $-(2/5)\log_2(2/5) - (3/5)\log_2(3/5)$
  = ***0.9696***
- **Entropy($S_{\text{outlook=overcast}}$)** = 0
  (all belongs to class yes)
- **Entropy($S_{\text{outlook=rain}}$)**
  = Entropy([3+, 2-])
  = $-(3/5)\log_2(3/5) - (2/5)\log_2(2/5)$
  = ***0.9696***

Gain(S, Outlook)

= Entropy(S) –

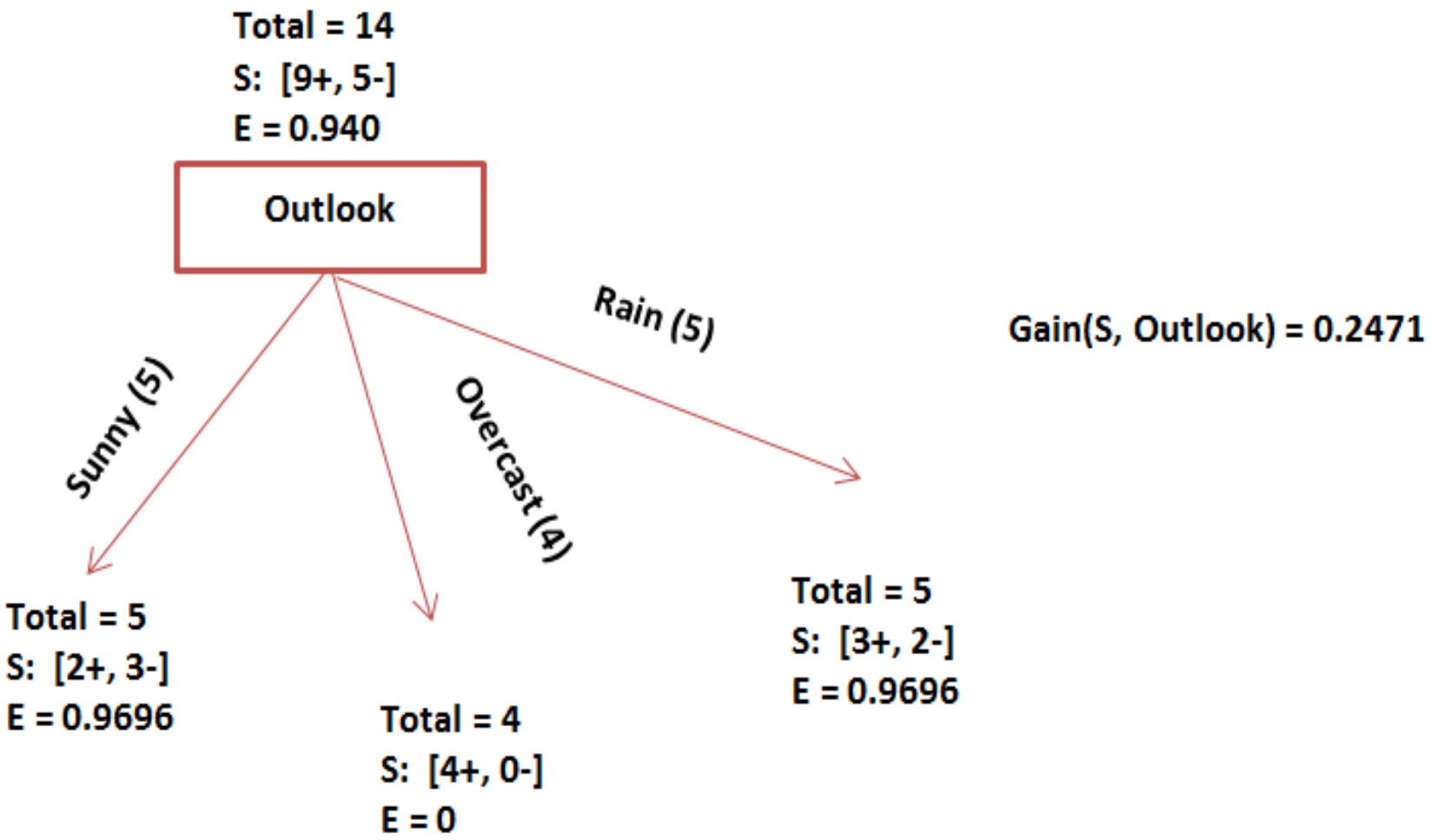
[(| $S_{outlook=sunny}$ | / |S|) Entropy($S_{outlook=sunny}$) +

(| $S_{outlook=overcast}$ | / |S|) Entropy($S_{outlook=overcast}$) +

(| $S_{outlook=rain}$ | / |S|) Entropy($S_{outlook=rain}$)]

= 0.940 - [(5 / 14) * 0.9696 + (4/14) * 0 + (5/14)* 0.9696]

= ***0.2471***


Total = 14
S: [9+, 5-]
E = 0.940
Outlook
Sunny (5)
Overcast (4)
Rain (5)
Gain(S, Outlook) = 0.2471
Total = 5
S: [2+, 3-]
E = 0.9696
Total = 4
S: [4+, 0-]
E = 0
Total = 5
S: [3+, 2-]
E = 0.9696

1.3: Attribute A = Temperature

- Consider attribute ***Temperature***
- ***Value(Temperature) = {hot, cool, mild}***

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- $|S_{temperature=hot}| = 4$
  - Yes = 2
  - No = 2
  - **$S_{temperature=hot}$ ← [2+, 2-]**
- $|S_{temperature=cool}| = 4$
  - Yes = 3
  - No = 1
  - **$S_{temperature=cool}$ ← [3+, 1-]**
- $|S_{temperature=mild}| = 6$
  - Yes = 4
  - No = 2
  - **$S_{temperature=mild}$ ← [4+, 2-]**

- **Entropy($S_{\text{temperature=hot}}$)**
  = Entropy([2+, 2-])
  = $-(2/4)\log_2(2/4) - (2/4)\log_2(2/4)$
  = ***1***
- **Entropy($S_{\text{temperature=cool}}$)**
  = Entropy([3+, 1-])
  = $-(3/4)\log_2(3/4) - (1/4)\log_2(1/4)$
  = ***0.81125***
- **Entropy($S_{\text{temperature=mild}}$)**
  = Entropy([4+, 2-])
  = $-(4/6)\log_2(4/6) - (2/6)\log_2(2/6)$
  = ***0.9164***

Gain(S, Temperature)

= Entropy(S) –

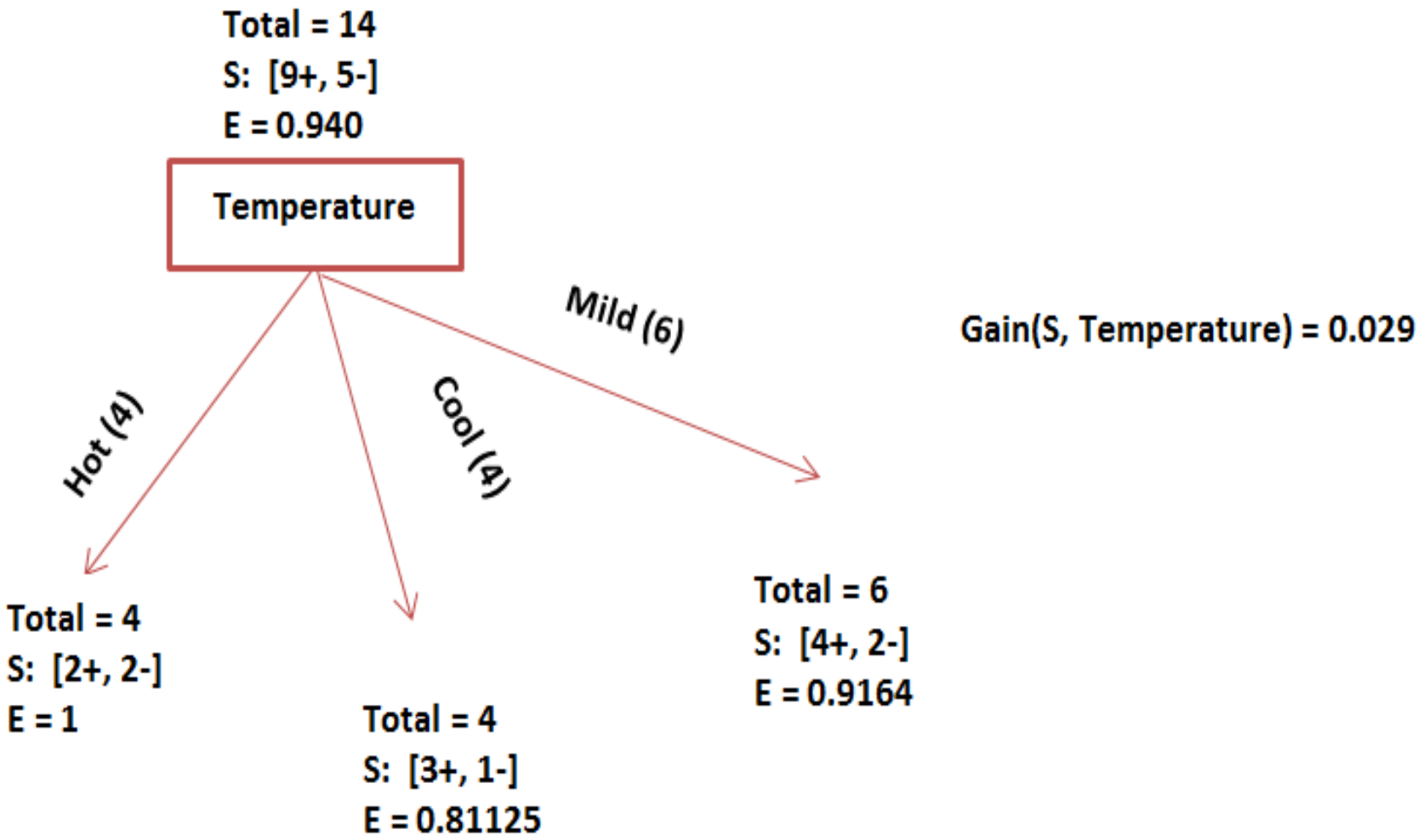
[(| $S_{temperature=hot}$ | / |S|) Entropy($S_{temperature=hot}$) +

(| $S_{temperature=cool}$ | / |S|) Entropy($S_{temperature=cool}$) +

(| $S_{temperature=mild}$ | / |S|) Entropy($S_{temperature=mild}$)]

= 0.940 - [(4 / 14) * 1 + (4/14) * 0.81125 +
(6/14)* 0.9164]

= ***0.029***


Total = 14
S: [9+, 5-]
E = 0.940
Temperature
Hot (4)
Cool (4)
Mild (6)
Gain(S, Temperature) = 0.029
Total = 4
S: [2+, 2-]
E = 1
Total = 4
S: [3+, 1-]
E = 0.81125
Total = 6
S: [4+, 2-]
E = 0.9164

1.4: Attribute A = Humidity

- Consider attribute ***Humidity***
- ***Value(humidity)= {normal, high}***

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- $|S_{humidity=normal}| = 7$
  - Yes = 6
  - No = 1
  - **$S_{humidity=normal}$ ← [6+, 1-]**
- $|S_{humidity=high}| = 7$
  - Yes = 3
  - No = 4
  - **$S_{humidity=high}$ ← [3+, 4-]**

- **Entropy($S_{\text{humidity=normal}}$)**

  = Entropy([6+, 1-])

  = $-(6/7)\log_2(6/7) - (1/7)\log_2(1/7)$

  = ***0.5888***

- **Entropy($S_{\text{humidity=high}}$)**

  = Entropy([3+, 4-])

  = $-(3/7)\log_2(3/7) - (4/7)\log_2(4/7)$

  = ***0.9849***

Gain(S, Humidity)

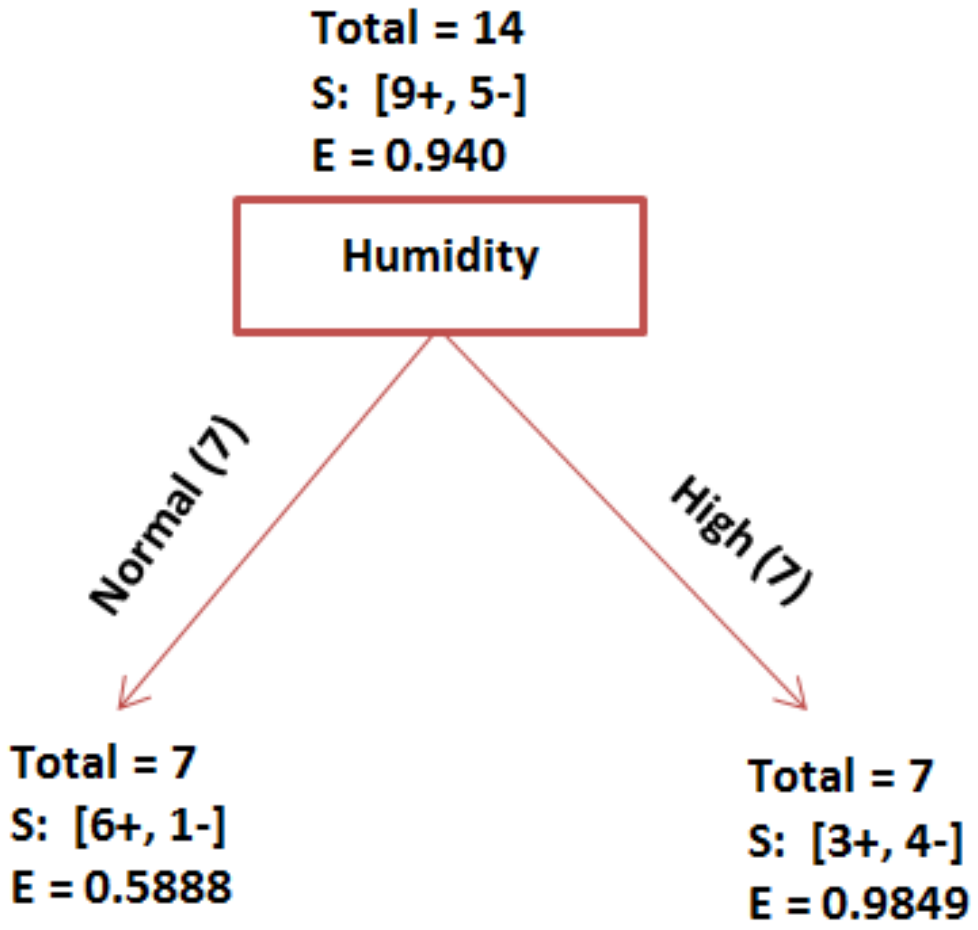
= Entropy(S) –

[(| $S_{humidity=normal}$ | / |S|) Entropy($S_{humidity=normal}$) +

(| $S_{humidity=high}$ | / |S|) Entropy($S_{humidity=high}$)]

= 0.940 - [(7 / 14) * 0.5888 + (7 / 14) * 0.9849]

= ***0.153***

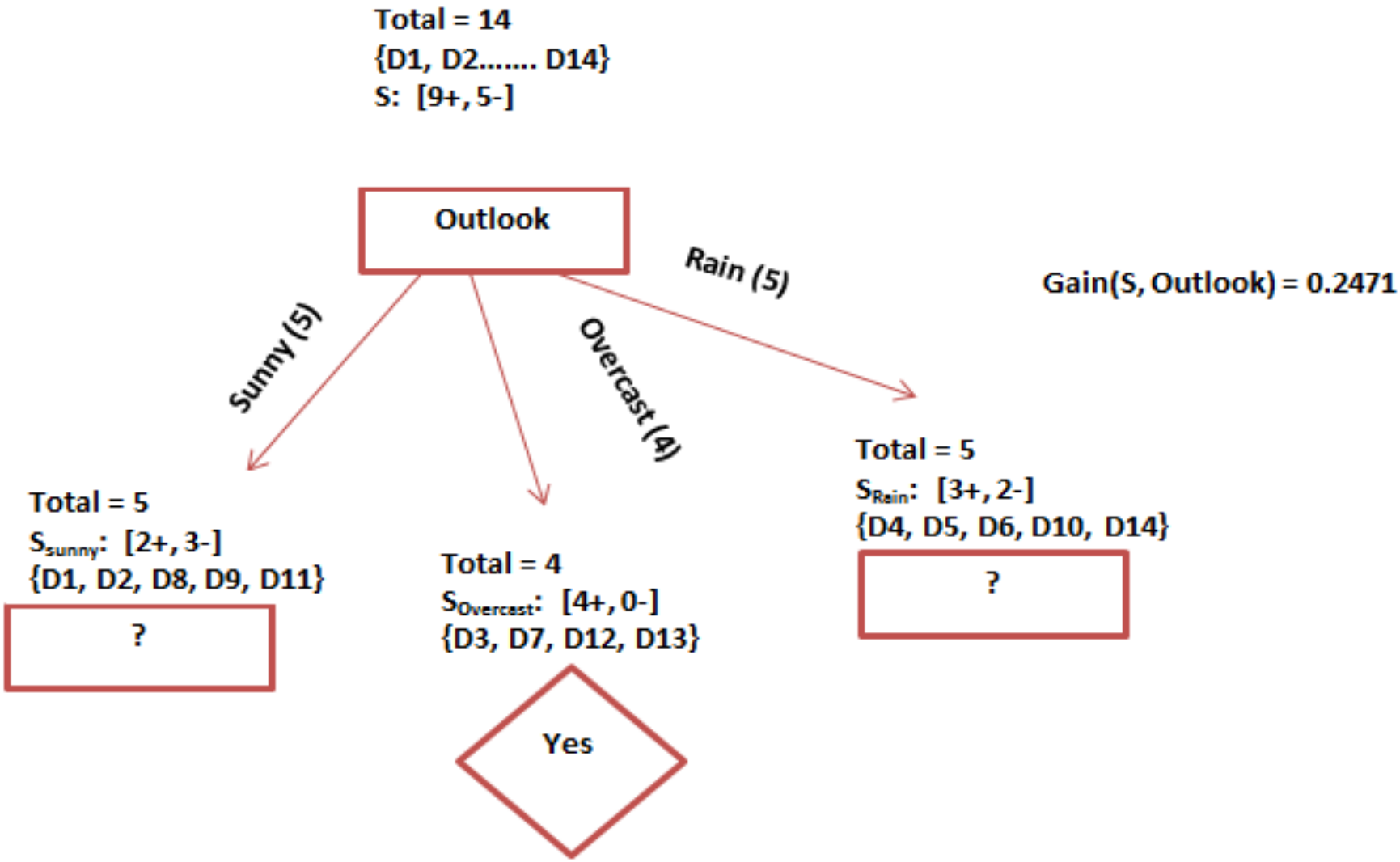


Gain(S, Humidity) = 0.153

- Gain(S, Wind) = 0.048 **(1.1)**
- ***Gain(S, Outlook) = 0.246*** **(1.2)**
- Gain(S, Temperature) = 0.029 **(1.3)**
- Gain(S, Humidity) = 0.153 **(1.4)**

- ***Outlook attribute is having the highest gain. So it becomes root node***

- Note: For $\mathbf{S}_{\text{outlook=}\mathbf{overcast}}$ all the records are associated with class label as **Yes**.
- So leaf node is created for **Outlook = Overcast** with class **label = Yes**.


Total = 14
{D1, D2……. D14}
S: [9+, 5-]
Outlook
Rain (5)
Gain(S, Outlook) = 0.2471
Sunny (5)
Overcast (4)
Total = 5
S_Rain: [3+, 2-]
{D4, D5, D6, D10, D14}
?
Total = 5
S_sunny: [2+, 3-]
{D1, D2, D8, D9, D11}
?
Total = 4
S_Overcast: [4+, 0-]
{D3, D7, D12, D13}
Yes

Records associated with Outlook = Sunny

$S_{outlook=sunny}$ = {D1 , D2, D8, D9, D11}

Records associated with Outlook = Overcast

$S_{outlook=overcast}$ = {D3, D7, D12, D13}

Records associated with Outlook = Rain

$S_{outlook=rain}$ = {D4, D5, D6, D10, D14}

Records associated with Outlook = Sunny

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Records associated with Outlook = Rain

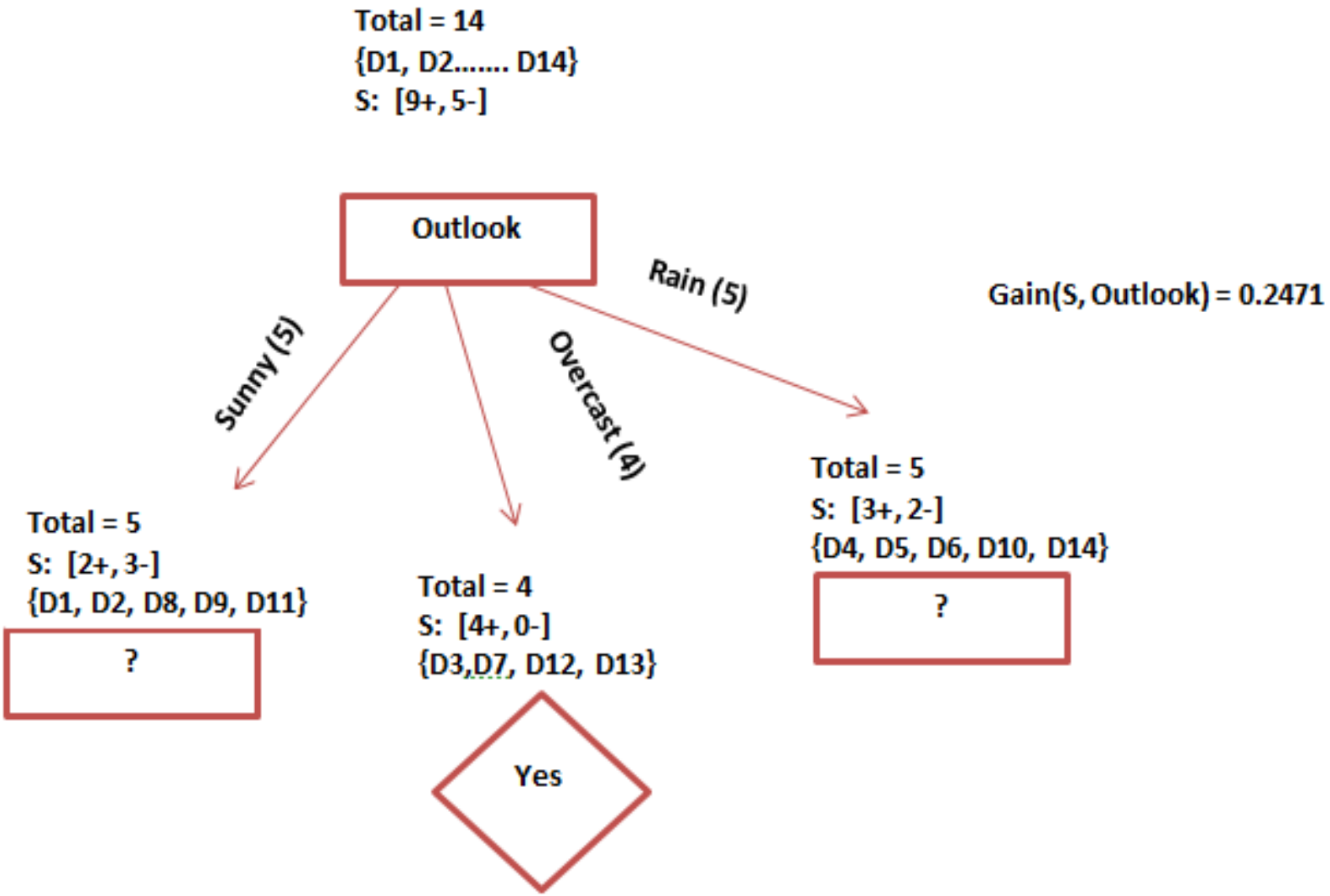
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Records associated with Outlook = Overcast

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- Remaining attributes are: Humidity ,Wind Temperature
- Now test $\mathbf{S_{sunny}}$ with respect to above three attributes. Later $\mathbf{S_{rain}}$ with the remaining attributes. Note: no need to test $\mathbf{S_{overcast}}$ as all the records associated with has class label as **Yes**.
- Now calculate:
  - Gain($S_{outlook=sunny}$, Humidity)
  - Gain($S_{outlook=sunny}$, Wind)
  - Gain($S_{outlook=sunny}$, Temperature)

Outlook becomes root node.



- Remaining attributes are:
  - Humidity
  - Wind
  - Temperature
- Now test $\mathbf{S}_{\text{outlook=}\mathbf{sunny}}$ with respect to above three attributes.
- Later $\mathbf{S}_{\text{outlook=}\mathbf{rain}}$ with the remaining attributes.
- Note: no need to test $\mathbf{S}_{\text{outlook=}\mathbf{overcast}}$ as all the records associated with it has class label as **Yes**.

- Now calculate:
  - **Gain($S_{outlook=sunny}$, Humidity) (2.1)**
  - **Gain($S_{outlook=sunny}$, Temperature) (2.2)**
  - **Gain($S_{outlook=sunny}$, Wind) (2.3)**

Records associated with Outlook = Sunny

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

2.1 To compute Gain ($S_{outlook=sunny}$, Humidity)

Records associated with Outlook = Sunny

S_{sunny} = {D1 , D2, D8, D9, D11}

- Lets add attribute **Humidity**

Records associated with Outlook = Sunny and Attribute A = Humidity

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

Outlook = Sunny and Attribute = Humidity

- **Values (Humidity) = {Normal, High}**
- Gain($S_{outlook=sunny}$, Humidity)
- $|S_{outlook=\mathbf{sunny}}| = 5$
  - $S_{sunny} \leftarrow [2+, 3-]$
- Values(Humidity) = {Normal, High}
- $|S_{\mathbf{humidity=normal}}| = 2$
  - $S_{\mathbf{humidity=}normal} \leftarrow [2+, 0-]$
- $|S_{\mathbf{humidity=high}}| = 3$
  - $S_{\mathbf{humidity=}high} \leftarrow [0+, 3-]$

- **Entropy($S_{humidity=normal}$)** = 0
- **Entropy($S_{humidity=high}$)** = 0

- **Gain($S_{outlook=sunny}$, Humidity) =**

Entropy($S_{outlook=sunny}$) –

[(| $S_{humidity=normal}$ | / | $S_{outlook=sunny}$ |)Entropy($S_{humidity=normal}$)
+ (| $S_{humidity=high}$ | / | $S_{outlook=sunny}$ |) Entropy($S_{humidity=high}$)]

= 0.9696 - [(2 / 5) * 0 + (3 / 5) * 0]

= ***0.9696***

2.2 To compute Gain ($S_{outlook=sunny}$, Temperature)

Records associated with Outlook = Sunny and Attribute A = Temperature

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

Outlook = Sunny and Attribute A = Temperature

- **Values (Temperature) = {Hot, Mild, Cold }**
- **$|S_{outlook=sunny}| = 5$**
- **$S_{outlook=sunny}$ ← [2+, 3-]**
- **Entropy($S_{outlook=sunny}$) = 0.9696**
- **$|S_{temperature=hot}| = 2$**
  - **$S_{temperature=hot}$ ← [0+, 2-]**
- **$|S_{temperature=mild}| = 2$**
  - **$S_{temperature=mild}$ ← [1+, 1-]**
- **$|S_{temperature=cold}| = 1$**
  - **$S_{temperature=cold}$ ← [1+, 0-]**

- **Entropy($S_{\text{temperature=}\mathbf{hot}}$)** = 0 (all negative)
- **Entropy($S_{\text{temperature=}\mathbf{mild}}$)** = 1 (equal)
- **Entropy($S_{\text{temperature=}\mathbf{cold}}$)** = 0 (all positive)

- **Gain($S_{outlook=sunny}$, Temperature) =**

Entropy($S_{outlook=sunny}$) –

[(| $S_{temperature=hot}$ | / | $S_{outlook=sunny}$ |) Entropy($S_{temperature=hot}$) +

(| $S_{temperature=mild}$ | / | $S_{outlook=sunny}$ |) Entropy($S_{temperature=mild}$)+

(| $S_{temperature=cold}$ | / | $S_{outlook=sunny}$ |) Entropy($S_{temperature=cold}$)]

= 0.9696 - [(2 / 5) * 0 + (2 / 5) * 1 + (1 / 5) * 0]

= ***0.5696***

2.3 To compute Gain ($S_{outlook=sunny}$, Wind)

Records associated with Outlook = Sunny and Attribute A = Wind

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

Outlook = Sunny and Attribute A = Wind

- **Values (Wind) = {weak, strong}**
- **$|S_{outlook=sunny}| = 5$**
- **$S_{outlook=sunny} \leftarrow [2+, 3-]$**
- **$Entropy(S_{outlook=sunny}) = 0.9696$**
- **$|S_{wind=weak}| = 3$**
  - **$S_{wind=weak} \leftarrow [1+, 2-]$**
- **$|S_{wind=strong}| = 2$**
  - **$S_{wind=strong} \leftarrow [1+, 1-]$**

- **Entropy($S_{wind=\textbf{weak}}$)**

  = Entropy([1+, 2-])

  = $-(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$

  = ***0.9164***

- **Entropy($S_{wind=\textbf{strong}}$)** = 1 (equal)

- **Gain($S_{outlook=sunny}$, Wind) =**

Entropy($S_{outlook=sunny}$) –

[(| $S_{wind=weak}$ | / | $S_{outlook=sunny}$ |) Entropy($S_{wind=weak}$) +

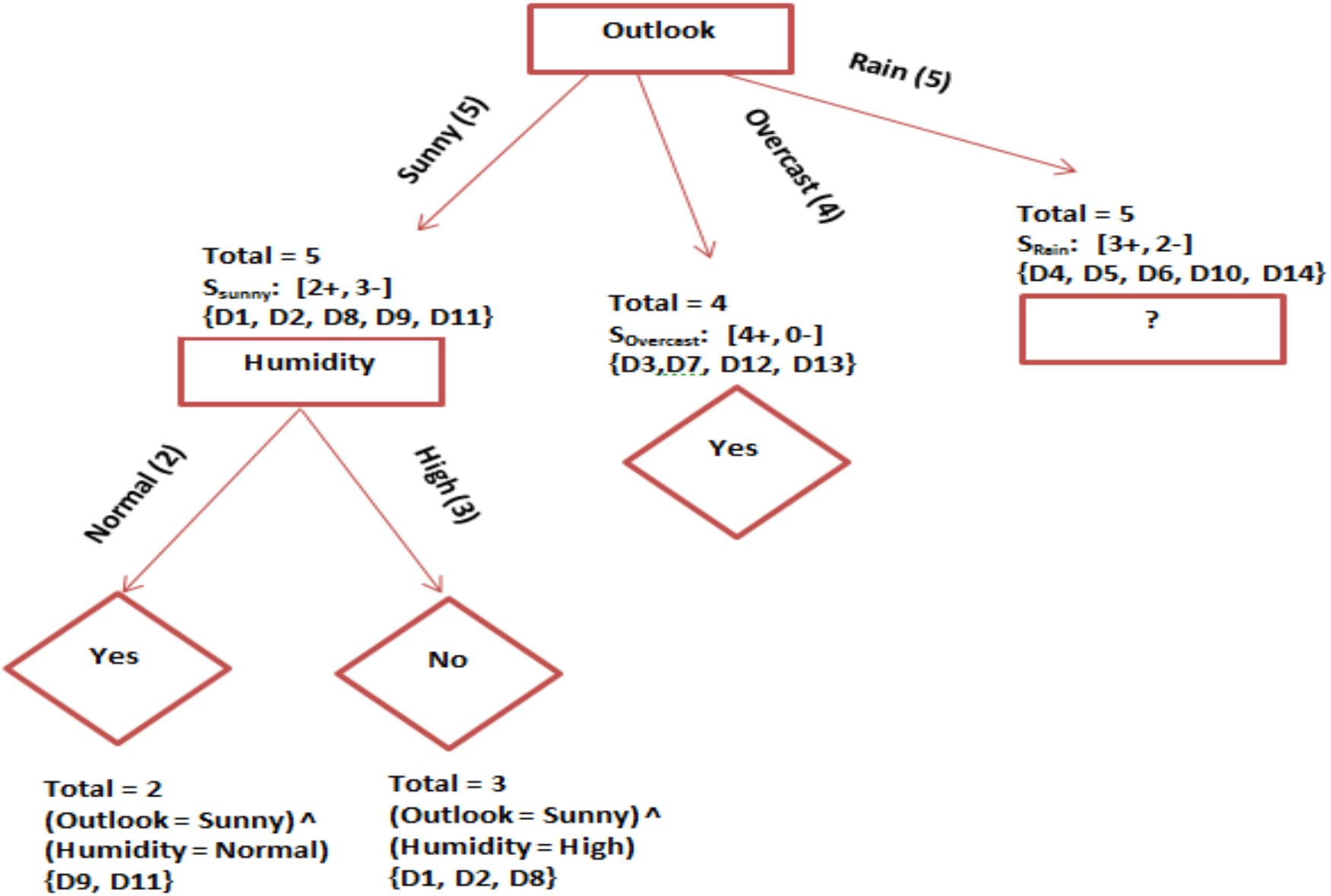
(| $S_{wind=strong}$ | / | $S_{outlook=sunny}$ |) Entropy($S_{wind=strong}$)]

= 0.9696 - [(3 / 5) * 0.9164 + (2 / 5) * 1]

= ***0.01976***

- **So,**
- ***Gain($S_{outlook=sunny}$, Humidity) = 0.9696***
- Gain($S_{outlook=sunny}$, Wind) = 0.570
- Gain($S_{outlook=sunny}$, Temperature) = 0.019

- The attribute **Humidity** has the highest gain. So attach it as a child node for **Outlook=Sunny**.
- For **Outlook=Sunny and Humidity = Normal** all the records belongs to **label = yes**, so attach it as leaf node for **Humidity = Normal**
- For **Outlook=Sunny and Humidity = High** all the records belongs to **label = No**, so attach it as leaf node for **Humidity = High**


Outlook
Sunny (5)
Overcast (4)
Rain (5)
Total = 5
S_{sunny}: [2+, 3-]
{D1, D2, D8, D9, D11}
Humidity
Total = 4
$S_{Overcast}$: [4+, 0-]
{D3,D7, D12, D13}
Yes
Total = 5
S_{Rain}: [3+, 2-]
{D4, D5, D6, D10, D14}
?
Normal (2)
High (3)
Yes
No
Total = 2
(Outlook = Sunny) ^
(Humidity = Normal)
{D9, D11}
Total = 3
(Outlook = Sunny) ^
(Humidity = High)
{D1, D2, D8}

- Remaining attributes are: Wind, Temperature
- Now test $\mathbf{S_{rain}}$ with respect to above two attributes.

- Now calculate:
  - Gain($\mathbf{S_{rain}}$, Temperature) **(3.1)**
  - Gain($\mathbf{S_{rain}}$, Wind) **(3.2)**
  - Gain($\mathbf{S_{rain}}$, Humidity) **(3.3)**

- **Records associated with Outlook = Rain is S_{rain} = {D4, D5, D6, D10, D14}**

3.1 To compute Gain($S_{outlook=rain}$, Temperature)

Records associated with Outlook = Rain and Attribute A = Temperature

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Outlook = Rain and Attribute A = Temperature.

- **Values (Temperature) = {Hot, Mild, Cold }**
- **$|S_{outlook=rain}| = 5$**
- **$S_{outlook=rain} \leftarrow [3+, 2-]$**
- **$Entropy(S_{outlook=rain}) = 0.9696$**
- **$|S_{temperature=hot}| = 0$**
  - **$S_{temperature=hot} \leftarrow [0+, 0-]$**
- **$|S_{temperature=mild}| = 3$**
  - **$S_{temperature=mild} \leftarrow [2+, 1-]$**
- **$|S_{temperature=cold}| = 2$**
  - **$S_{temperature=cold} \leftarrow [1+, 1-]$**

- **Entropy($S_{\text{temperature=}\mathbf{hot}}$)** = 0 (all negative)
- **Entropy($S_{\text{temperature=}\mathbf{mild}}$)**

  = Entropy([2+, 1-])

  = $-(2/3)\log_2(2/3) - (1/3)\log_2(1/3)$

  = ***0.9164***
- **Entropy($S_{\text{temperature=}\mathbf{cold}}$)** = 1 (equal)

- **Gain($S_{outlook=rain}$, Temperature) =**

Entropy($S_{outlook=rain}$) –

[(| $S_{temperature=hot}$ | / | $S_{outlook=rain}$ |) Entropy($S_{temperature=hot}$) +

(| $S_{temperature=mild}$ | / | $S_{outlook=rain}$ |) Entropy($S_{temperature=mild}$) +

(| $S_{temperature=cold}$ | / | $S_{outlook=rain}$ |) Entropy($S_{temperature=cold}$)]

= 0.9696 - [(0 / 5) * 0 + (3 / 5) * 0.9164 + (2 / 5) * 1]

= ***0.01976***

3.2 To compute Gain($S_{outlook=rain}$, Wind)

Records associated with Outlook = Rain and Attribute A = Wind

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Outlook = Rain and Attribute A = Wind

- **Values (Wind) = {weak, strong}**
- **$|S_{outlook=rain}| = 5$**
- **$S_{outlook=rain} \leftarrow [2+, 3-]$**
- **$Entropy(S_{outlook=rain}) = 0.9696$**
- **$|S_{wind=weak}| = 3$**
  - **$S_{wind=weak} \leftarrow [3+, 0-]$**
- **$|S_{wind=strong}| = 2$**
  - **$S_{wind=strong} \leftarrow [0+, 2-]$**

- **Entropy($S_{wind=\textbf{weak}}$)** = 0 (all yes)
- **Entropy($S_{wind=\textbf{strong}}$)** = 0 (all no)

- **Gain($S_{outlook=rain}$, Wind) =**

Entropy($S_{outlook=rain}$) –

[(| $S_{wind=weak}$ | / | $S_{outlook=rain}$ |) Entropy($S_{wind=weak}$) +
(| $S_{wind=strong}$ | / | $S_{outlook=rain}$ |) Entropy($S_{wind=strong}$)]

= 0.9696 - [(3 / 5) * 0 + (2 / 5) * 0]

= 0.**9696**

3.3 To compute Gain($S_{outlook=rain}$, Humidity)

Records associated with Outlook = Rain and Attribute A = Humidity

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Outlook = Rain and Attribute A = Humidity

- **Values (Humidity) = {high, normal}**
- **$|S_{outlook=rain}| = 5$**
- **$S_{outlook=rain} \leftarrow [2+, 3-]$**
- **$Entropy(S_{outlook=rain}) = 0.9696$**
- **$|S_{humidity=high}| = 2$**
  - **$S_{humidity=high} \leftarrow [1+, 1-]$**
- **$|S_{humidity=normal}| = 3$**
  - **$S_{humidity=normal} \leftarrow [2+, 1-]$**

- **Entropy($S_{\text{humidity=}\mathbf{high}}$)** = 1
- **Entropy($S_{\text{humidity=}\mathbf{normal}}$)**

  = Entropy([2+, 1-])

  = $-(2/3)\log_2(2/3) - (1/3)\log_2(1/3)$

  = ***0.9164***

- **Gain($S_{outlook=rain}$, Humidity) =**

Entropy($S_{outlook=rain}$) –

[(| $S_{humidity=normal}$ | / | $S_{outlook=rain}$ |) Entropy($S_{humidity=normal}$)
+(| $S_{humidity=high}$ | / | $S_{outlook=rain}$ |) Entropy($S_{humidity=high}$)]

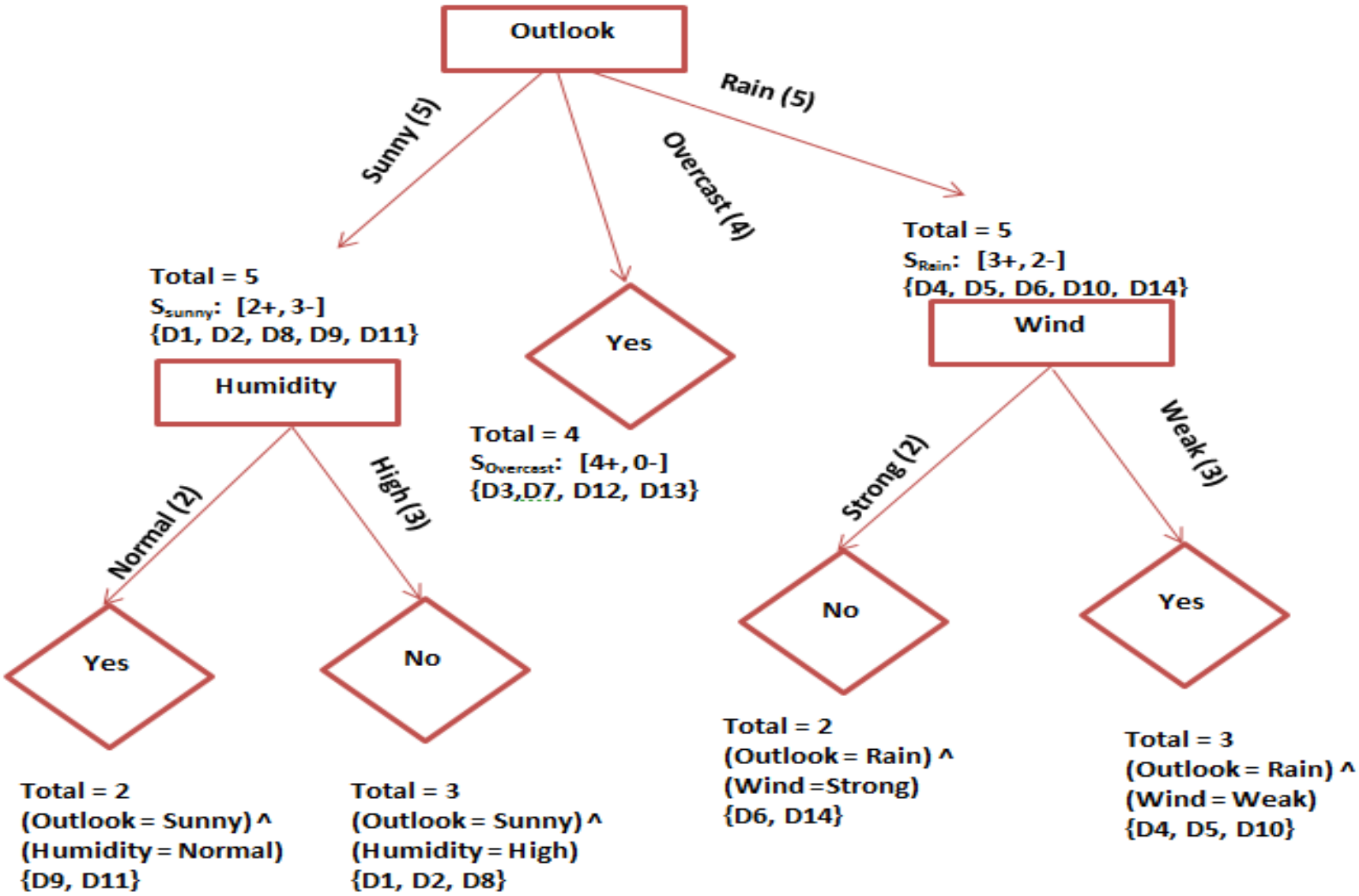
= 0.9696 - [(3 / 5) * 0.9164 + (2 / 5) * 1]

= 0.01976

- So,
- ***Gain($S_{outlook=rain}$, Wind) = 0.9696***
- Gain($\mathbf{S}_{outlook=\mathbf{rain}}$, Temperature) = 0.01976
- Gain($\mathbf{S}_{outlook=\mathbf{rain}}$, Humidity) = 0.01976

- The attribute **Wind** has the highest gain. So attach it as a child node for **Outlook=Rain**.
- For **Outlook= Rain and Wind = Strong** all the records belongs to **class label = No**, so attach it as leaf node for **Wind = Strong**
- For **Outlook= Rain and Wind = Weak** all the records belongs to **class label = Yes**, so attach it as leaf node for **Wind = Weak**

So, a final decision tree is:



- The process of selecting a new attribute and partitioning the training examples is now repeated for each nonterminal descendant node, this time using only the training examples associated with that node.
- Attributes that have been incorporated higher in the tree are excluded, so that any given attribute can appear at most once along any path through the tree.

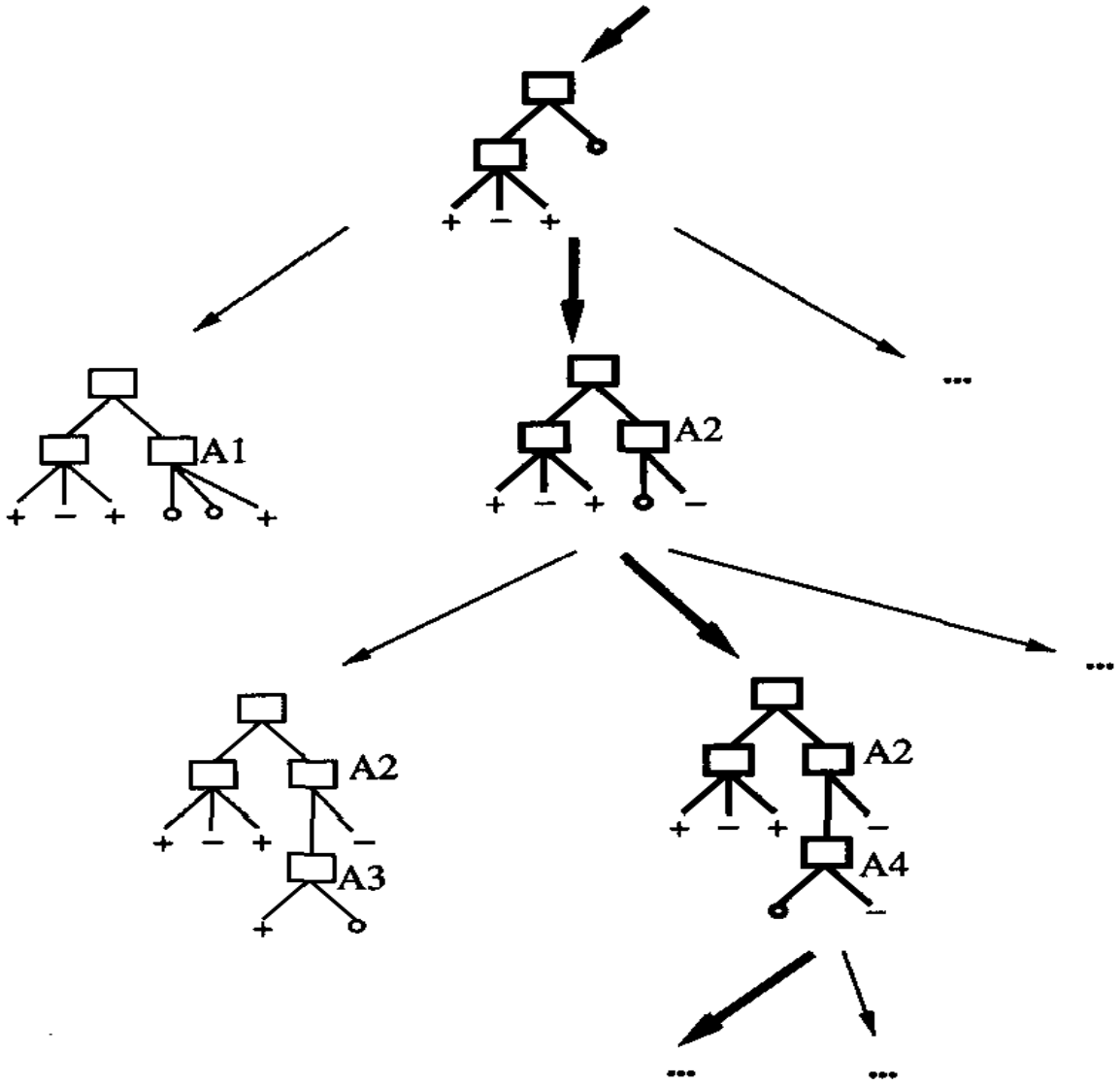
- This process continues for each new leaf node until either of two conditions is met:
  - every attribute has already been included along this path through the tree, or
  - the training examples associated with this leaf node all have the same target attribute value (i.e., their entropy is zero).

3.5 HYPOTHESIS SPACE SEARCH IN DECISION TREE LEARNING

- As with other inductive learning methods, ID3 can be characterized as searching a space of hypotheses for one that fits the training examples.
- The hypothesis space searched by ID3 is the set of possible decision trees.

- ID3 performs a simple-to complex, hill-climbing search through this hypothesis space, beginning with the empty tree, then considering progressively more elaborate hypotheses in search of a decision tree that correctly classifies the training data.

- The evaluation function that guides this hill-climbing search is the information gain measure.


A1
A2
A2
A3
A2
A4

Insights into ID3: capabilities and limitations

- ID3's hypothesis space of all decision trees is a ***complete*** space of finite discrete-valued functions, relative to the available attributes.
  - Because every finite discrete-valued function can be represented by some decision tree, ID3 avoids one of the major risks of methods that search incomplete hypothesis spaces (such as methods that consider only conjunctive hypotheses): that the hypothesis space might not contain the target function.

- ID3 maintains only a single current hypothesis as it searches through the space of decision trees.
  - This contrasts, for example, with the earlier version space candidate-elimination, which maintains the set of ***all*** hypotheses consistent with the available training examples.
  - By determining only a single hypothesis, ID3 loses the capabilities that follow from explicitly representing all consistent hypotheses. For example, it does not have the ability to determine how many alternative decision trees are consistent with the available training data, or to pose new instance queries that optimally resolve among these competing hypotheses.

- **ID3** in its pure form performs no backtracking in its search.
  - Once it, selects an attribute to test at a particular level in the tree, it never backtracks to reconsider this choice.
  - Therefore, it is susceptible to the usual risks of hill-climbing search without backtracking: converging to locally optimal solutions that are not globally optimal.
  - In the case of **ID3,** a locally optimal solution corresponds to the decision tree it selects along the single search path it explores.
  - However, this locally optimal solution may be less desirable than trees that would have been encountered along a different branch of the search.

- **ID3** uses all training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis.
- This contrasts with methods that make decisions incrementally, based on individual training examples (e.g., FIND-S or CANDIDATE-ELIMINATION). One advantage of using statistical properties of all the examples (e.g., information gain) is that the resulting search is much less sensitive to errors in individual training examples.
- **ID3** can be easily extended to handle noisy training data by modifying its termination criterion to accept hypotheses that imperfectly fit the training data.

ISSUES IN DECISION TREE LEARNING

1. Avoiding Overfitting the Data
   - REDUCED ERROR PRUNING
   - RULE POST-PRUNING
2. Incorporating Continuous-Valued Attributes
3. Alternative Measures for Selecting Attributes
4. Handling Training Examples with Missing Attribute Values
5. Handling Attributes with Differing Costs

1. Avoiding Overfitting the Data

- The algorithm grows each branch of the tree just deeply enough to perfectly classify the training examples.
- While this is sometimes a reasonable strategy, in fact it can lead to difficulties when
  - there is noise in the data,
  - or when the number of training examples is too small to produce a representative sample of the true target function.
- In either of these cases, this simple algorithm can produce trees that ***overfit*** the training examples.

- We will say that a hypothesis overfits the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances (i.e., including instances beyond the training set).

Definition: Given a hypothesis space H, a hypothesis $h \in H$ is said to **overfit** the training data if there exists some alternative hypothesis $h' \in H$, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.

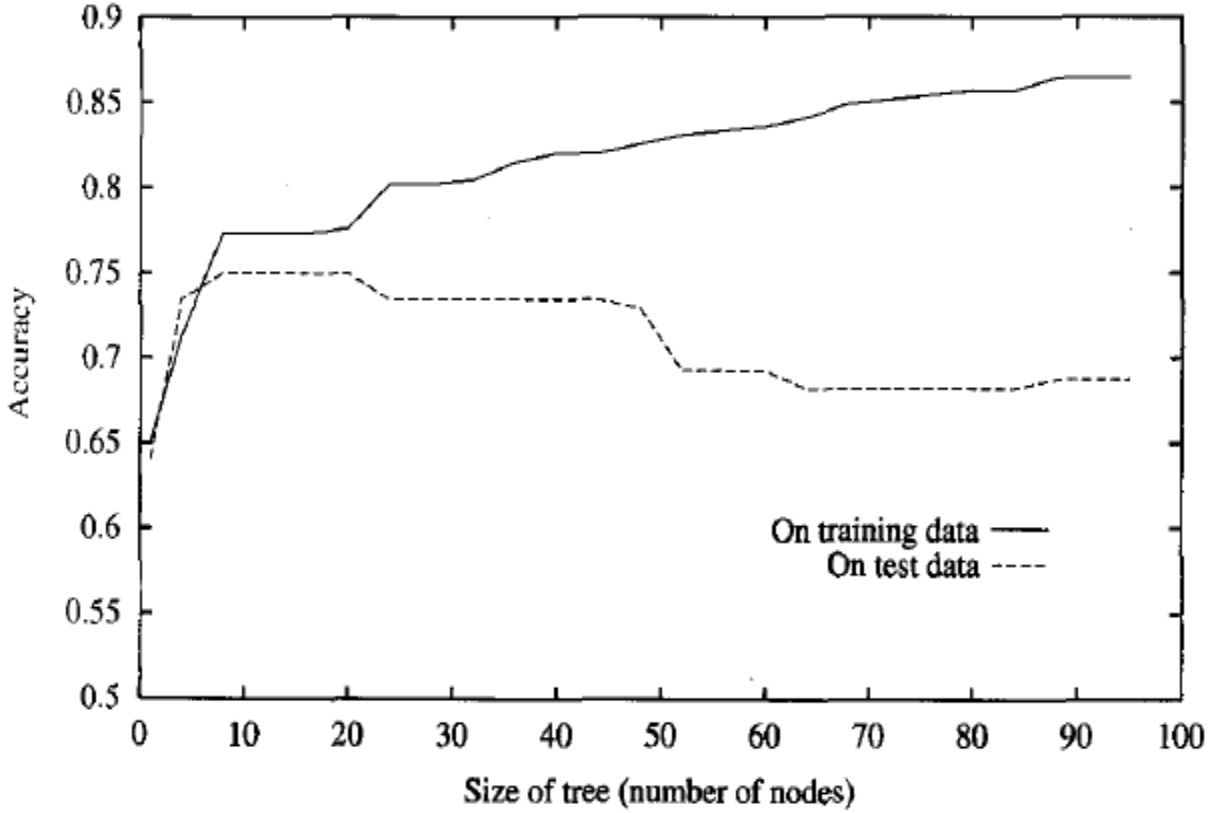


FIGURE 3.6

Overfitting in decision tree learning. As ID3 adds new nodes to grow the decision tree, the accuracy of the tree measured over the training examples increases monotonically. However, when measured over a set of test examples independent of the training examples, accuracy first increases, then decreases.

- Figure 3.6 illustrates the impact of overfitting in a typical application of decision tree learning.
- In this case, the ID3 algorithm is applied to the task of learning which medical patients have a form of diabetes.
- The horizontal axis of this plot indicates the total number of nodes in the decision tree, as the tree is being constructed.
- The vertical axis indicates the accuracy of predictions made by the tree.
- The solid line shows the accuracy of the decision tree over the training examples, whereas the broken line shows accuracy measured over an independent set of test examples (not included in the training set).

- Predictably, the accuracy of the tree over the training examples increases monotonically as the tree is grown.
- However, the accuracy measured over the independent test examples first increases, then decreases.
- As can be seen, once the tree size exceeds approximately *25* nodes, further elaboration of the tree decreases its accuracy over the test examples despite increasing its accuracy on the training examples.

- How can it be possible for tree h to fit the training examples better than h', but for it to perform more poorly over subsequent examples?
- One way this can occur is when the training examples contain random errors or noise.
- To illustrate, consider the effect of adding the following positive training example, incorrectly labeled as negative, to the (otherwise correct) examples in Table

$$\langle Outlook = Sunny, Temperature = Hot, Humidity = Normal,$$
$$Wind = Strong, PlayTennis = No \rangle$$

- Given the original error-free data, ID3 produces the decision tree shown in Figure 3.1.
- However, the addition of this incorrect example will now cause ID3 to construct a more complex tree.
- In particular, the new example will be sorted into the second leaf node from the left in the learned tree of Figure 3.1, along with the previous positive examples D9 and D11.
- Because the new example is labeled as a negative example, ID3 will search for further refinements to the tree below this node.

- Of course as long as the new erroneous example differs in some arbitrary way from the other examples affiliated with this node, ID3 will succeed in finding a new decision attribute to separate out this new example from the two previous positive examples at this tree node.
- The result is that ID3 will output a decision tree (h) that is more complex than the original tree from Figure 3.1 (h').
- Of course h will fit the collection of training examples perfectly, whereas the simpler h' will not.
- However, given that the new decision node is simply a consequence of fitting the noisy training example, we expect h to outperform h' over subsequent data drawn from the same instance distribution.

- The above example illustrates how random noise in the training examples can lead to overfitting.
- In fact, overfitting is possible even when the training data are noise-free, especially when small numbers of examples are associated with leaf nodes.
- In this case, it is quite possible for coincidental regularities to occur, in which some attribute happens to partition the examples very well, despite being unrelated to the actual target function.
- Whenever such coincidental regularities exist, there is a risk of overfitting.

- There are several approaches to avoiding overfitting in decision tree learning.These can be grouped into two classes:
  - approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data,
  - approaches that allow the tree to overfit the data, and then post-prune the tree.

- a key question is what criterion is to be used to determine the correct final tree size. Approaches include:
  - Use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.
  - Use all the available data for training, but apply a statistical test to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set. (Chi-square test)
  - Use an explicit measure of the complexity for encoding the training examples and the decision tree, halting growth of the tree when this encoding size is minimized. (Minimum Description Length methoid)

- The first of the above approaches is the most common and is referred to as a training and validation set approach. Two main variants:
- In this approach, the available data are separated into two sets of examples: a training set, which is used to form the learned hypothesis, and a separate validation set, which is used to evaluate the accuracy of this hypothesis over subsequent data and, in particular, to evaluate the impact of pruning this hypothesis.

- The motivation is this: Even though the learner may be misled by random errors and coincidental regularities within the training set, the validation set is unlikely to exhibit the same random fluctuations.
- Therefore, the validation set can be expected to provide a safety check against overfitting the spurious characteristics of the training set.
- Of course, it is important that the validation set be large enough to itself provide a statistically significant sample of the instances.
- One common heuristic is to withhold one-third of the available examples for the validation set, using the other two-thirds for training.

REDUCED ERROR PRUNING

- How exactly might we use a validation set to prevent overfitting?
- One approach, called reduced-error pruning (Quinlan 1987), is to consider each of the decision nodes in the tree to be candidates for pruning.
- Pruning a decision node consists of removing the subtree rooted at that node, making it a leaf node, and assigning it the most common classification of the training examples affiliated with that node.

- Nodes are removed only if the resulting pruned tree performs no worse than-the original over the validation set.
- This has the effect that any leaf node added due to coincidental regularities in the training set is likely to be pruned because these same coincidences are unlikely to occur in the validation set.
- Nodes are pruned iteratively, always choosing the node whose removal most increases the decision tree accuracy over the validation set.
- Pruning of nodes continues until further pruning is harmful (i.e., decreases accuracy of the tree over the validation set).

RULE POST-PRUNING

- In practice, one quite successful method for finding high accuracy hypotheses is a technique we shall call rule post-pruning.
- A variant of this pruning method is used by C4.5 (Quinlan 1993), which is an outgrowth of the original ID3 algorithm.

- Rule post-pruning involves the following steps:
  - **1.** Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing overfitting to occur.
  - 2. Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.
  - **3.** Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.
  - 4. Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

- To illustrate, consider again the decision tree in Figure 3.1.
- In rule postpruning, one rule is generated for each leaf node in the tree.
- Each attribute test along the path from the root to the leaf becomes a rule antecedent (precondition)
- and the classification at the leaf node becomes the rule consequent (postcondition).

- For example, the leftmost path of the tree in Figure 3.1 is translated into the rule

$$\text{IF} \quad (Outlook = Sunny) \wedge (Humidity = High)$$

$$\text{THEN} \quad PlayTennis = No$$

- Next, each such rule is pruned by removing any antecedent, or precondition, whose removal does not worsen its estimated accuracy.
- Given the above rule, for example, rule post-pruning would consider removing the preconditions (Outlook = Sunny) and (Humidity = High).
- It would select whichever of these pruning steps produced the greatest improvement in estimated rule accuracy, then consider pruning the second precondition as a further pruning step.
- No pruning step is performed if it reduces the estimated rule accuracy

Why convert the decision tree to rules before pruning?

- There are three main advantages.
- Converting to rules allows distinguishing among the different contexts in which a decision node is used. Because each distinct path through the decision tree node produces a distinct rule, the pruning decision regarding that attribute test can be made differently for each path. In contrast, if the tree itself were pruned, the only two choices would be to remove the decision node completely, or to retain it in its original form.
- Converting to rules removes the distinction between attribute tests that occur near the root of the tree and those that occur near the leaves. Thus, we avoid messy bookkeeping issues such as how to reorganize the tree if the root node is pruned while retaining part of the subtree below this test.
- Converting to rules improves readability. Rules are often easier for to understand.

2. Incorporating Continuous-Valued Attributes

- Our initial definition of ID3 is restricted to attributes that take on a discrete set of values.
- First, the target attribute whose value is predicted by the learned tree must be discrete valued.
- Second, the attributes tested in the decision nodes of the tree must also be discrete valued.
- This second restriction can easily be removed so that continuous-valued decision attributes can be incorporated into the learned tree.

- This can be accomplished by dynamically defining new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals.
- In particular, for an attribute A that is continuous-valued, the algorithm can dynamically create a new boolean attribute Ac, that is true if A < c and false otherwise.
- The only question is how to select the best value for the threshold c.

3. Alternative Measures for Selecting Attributes

- Attribute Date has so many possible values that it is bound to separate the training examples into very small subsets.
- Because of this, it will have a very high information gain relative to the training examples, despite being a very poor predictor of the target function over unseen instances.
- One way to avoid this difficulty is to select decision attributes based on some measure other than information gain.

- One alternative measure that has been used successfully is the gain ratio.
- The gain ratio measure penalizes attributes such as Date by incorporating a term, called split information, that is sensitive to how broadly and uniformly the attribute splits the data:

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_1 through Sc, are the ***c*** subsets of examples resulting from partitioning S by the c-valued attribute A.

- The Gain Ratio measure is defined in terms of the earlier Gain measure, as well as this Split information, as follows

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

4. Handling Training Examples with Missing Attribute Values

- In certain cases, the available data may be missing values for some attributes.
- For example, in a medical domain in which we wish to predict patient outcome based on various laboratory tests, it may be that the lab test Blood-Test-Result is available only for a subset of the patients.
- In such cases, it is common to estimate the missing attribute value based on other examples for which this attribute has a known value.

- Consider the situation in which ***Gain(S, A)*** is to be calculated at node ***n*** in the decision tree to evaluate whether the attribute ***A*** is the best attribute to test at this decision node.
- Suppose that ***(x, c(x))*** is one of the training examples in ***S*** and that the value ***A(x)*** is unknown.

- One strategy for dealing with the missing attribute value is to assign it the value that is most common among training examples at node ***n.***
- Alternatively, we might assign it the most common value among examples at node ***n*** that have the classification ***c(x)***

5. Handling Attributes with Differing Costs

- In some learning tasks the instance attributes may have associated costs.
- For example, in learning to classify medical diseases we might describe patients in terms of attributes such **as** Temperature, BiopsyResult, Pulse, BloodTestResults, etc.
- These attributes vary significantly in their costs, both in terms of monetary cost and cost to patient comfort.
- In such tasks, we would prefer decision trees that use low-cost attributes where possible, relying on high-cost attributes only when needed to produce reliable classifications.

- ID3 can be modified to take into account attribute costs by introducing a cost term into the attribute selection measure.
- For example, we might divide the ***Gain*** by the cost of the attribute, so that lower-cost attributes would be preferred.

$$\frac{Gain^2(S, A)}{Cost(A)}$$

$$\frac{2^{Gain(S,A)} - 1}{(Cost(A) + 1)^w}$$

INDUCTIVE BIAS in DECISION TREE LEARNING

- inductive bias:
  - the set of assumptions that, together with the training data, deductively justify the classifications assigned by the learner to future instances.

- For a given collection of training examples, there are many decision trees consistent with these examples.
- ID3 search strategy selects
  - (a) in favor of shorter trees over longer ones, and
  - (b)trees that place the attributes with highest information gain closest to the root.

- **Approximate inductive bias of ID3:**
  - Shorter trees are preferred over larger trees.

- **A closer approximation to the inductive bias of ID3:**
  - Shorter trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred over those that do not.

Restriction Biases and Preference Biases: ID3 v/s Candidate-Elimination

Tejaswini H, Assistant Professor,
Department of Computer Science

ID3

- ID3 searches a **complete** hypothesis space.
- It searches **incompletely** through this space, from simple to complex hypotheses, until its termination condition is met.
- Its inductive bias is solely a consequence of the ordering of hypotheses by its search strategy.
- Its hypothesis space introduces **no additional bias**.

Candidate-Elimination

- The version space CANDIDATE-ELIMINATION algorithm searches an **incomplete hypothesis space**.
- It searches this space **completely**, finding every hypothesis consistent with the training data.
- Its inductive bias is solely a consequence of the expressive power of its hypothesis representation.
- Its search strategy introduces no additional bias.

- In brief, the inductive bias of ID3 follows from its search strategy,
- whereas the inductive bias of the C-E follows from the definition of its search space.

- The inductive bias of ID3 is thus a preference for certain hypotheses over others (e.g., for shorter hypotheses), with no hard restriction on the hypotheses that can be eventually enumerated.
- This form of bias is typically called a **preference bias** (or, alternatively, a search bias).

- In contrast, the bias of the C-E is in the form of a categorical restriction on the set of hypotheses considered.
- This form of bias is typically called a **restriction bias** (or, alternatively, a language bias).

- Typically, a preference bias is more desirable than a restriction bias, because it allows the learner to work within a complete hypothesis space that is assured to contain the unknown target function.
- In contrast, a restriction bias that strictly limits the set of potential hypotheses is generally less desirable, because it introduces the possibility of excluding the unknown target function altogether

Why Prefer Short Hypotheses?

- **Occam's razor**: Prefer the simplest hypothesis that fits the data.