

INTRODUCTORY APPLIED MACHINE LEARNING

Yan-Fu Kuo

Dept. of Bio-industrial Mechatronics Engineering

National Taiwan University

Today:

- Decision tree

Outline

- Goal of the lecture
- Classification
- Decision tree induction
- Attribute selection
- Decision tree pruning
- Missing attribute

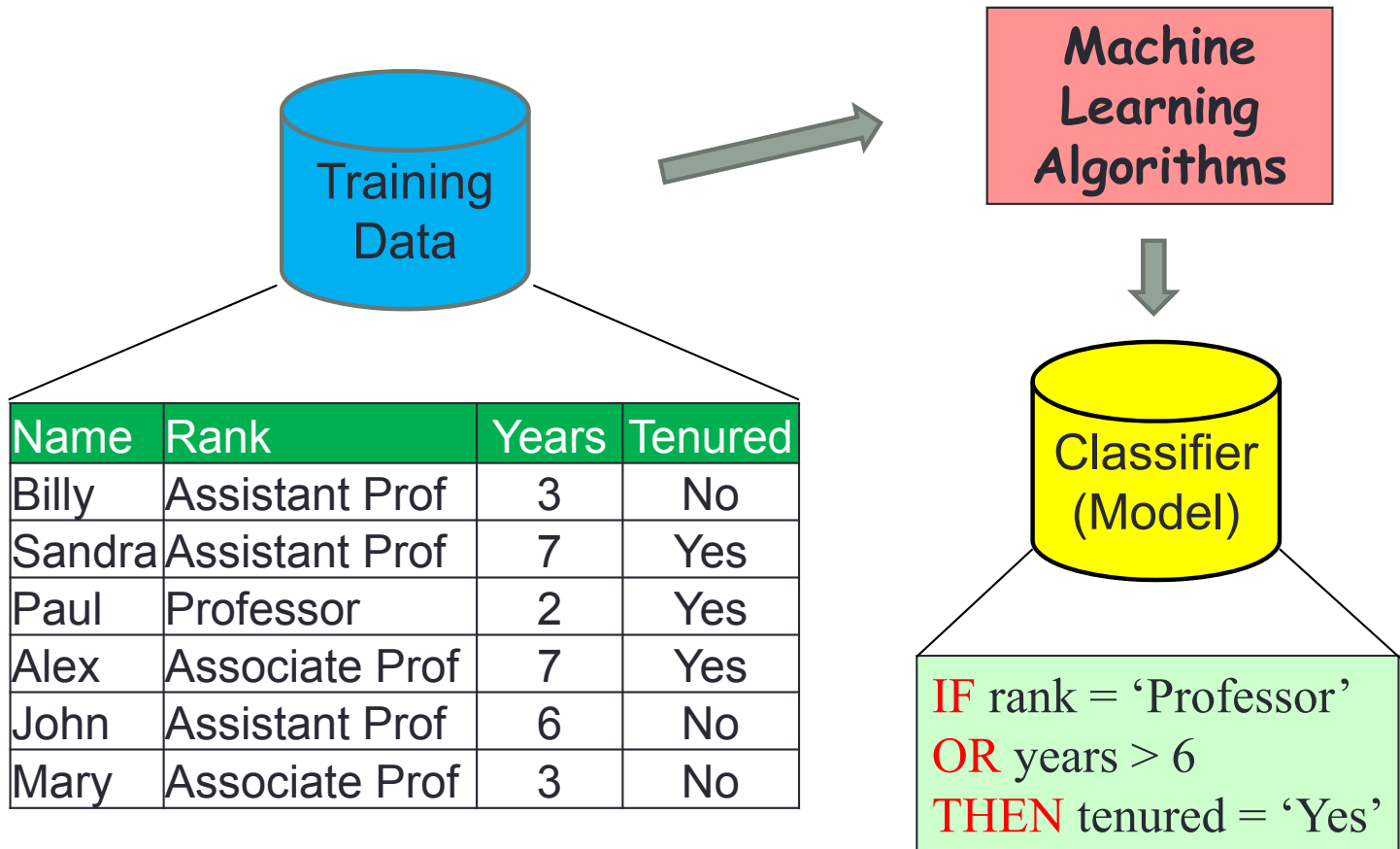
Goals

- After this, you should be able to:
 - Understand how decision tree works
 - Quantitatively assess the split performance of different attributes
 - Develop decision trees for given a set of data
 - Prune decision trees to avoid overfitting

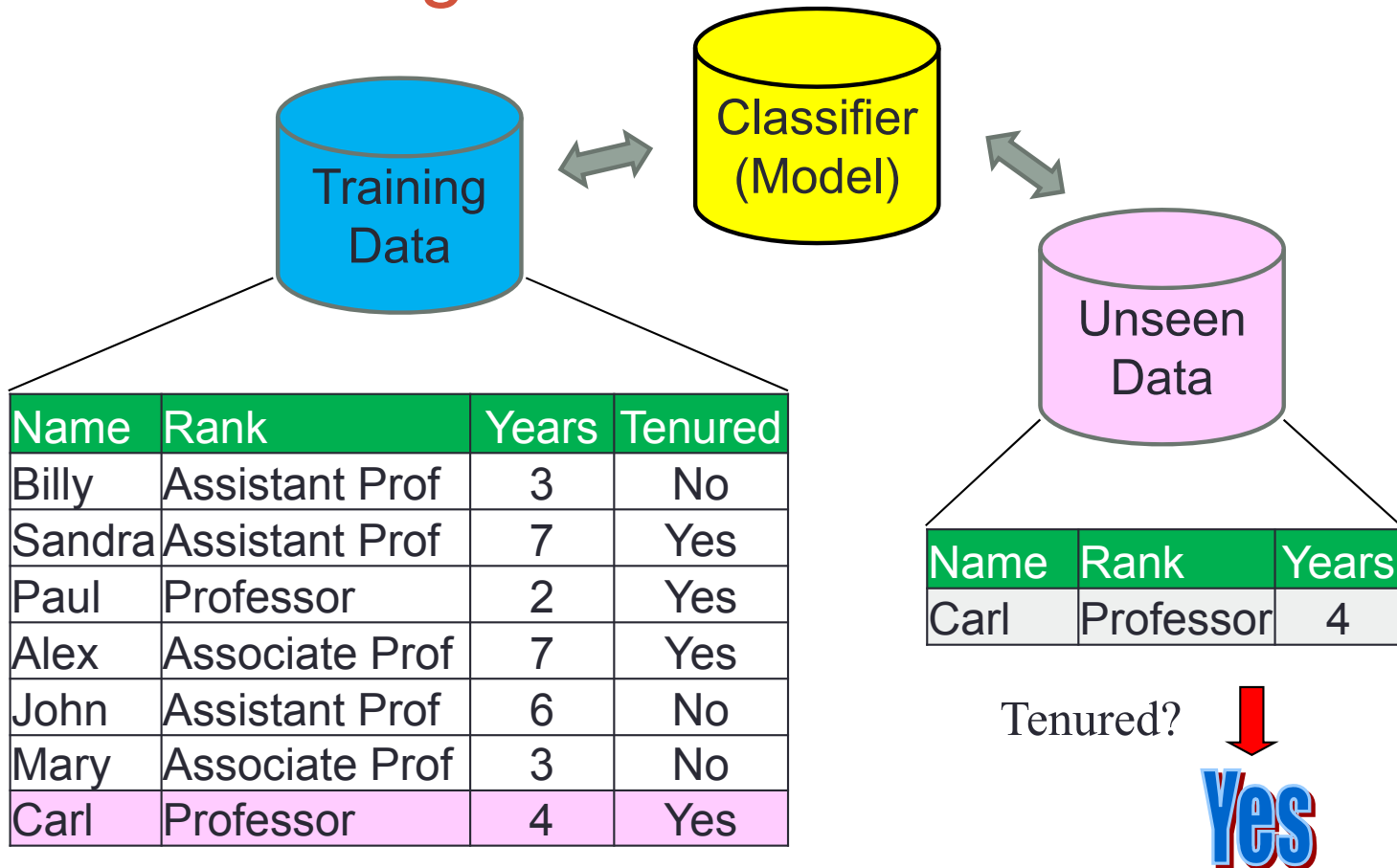
Classification – A Two-step Process

- Step I – model construction:
 - To describe a set of samples with predetermined classes (namely training sample set)
 - Each training sample belongs to a predefined class
 - The model is represented as classification rules, decision trees, or mathematical formula
- Step II – model usage:
 - To classify future or unknown objects
 - Needs to estimate accuracy of the model using a set of test samples, which is independent of the training sample set (otherwise overfitting)

Model Construction



Model Usage



Classification Problem Statement

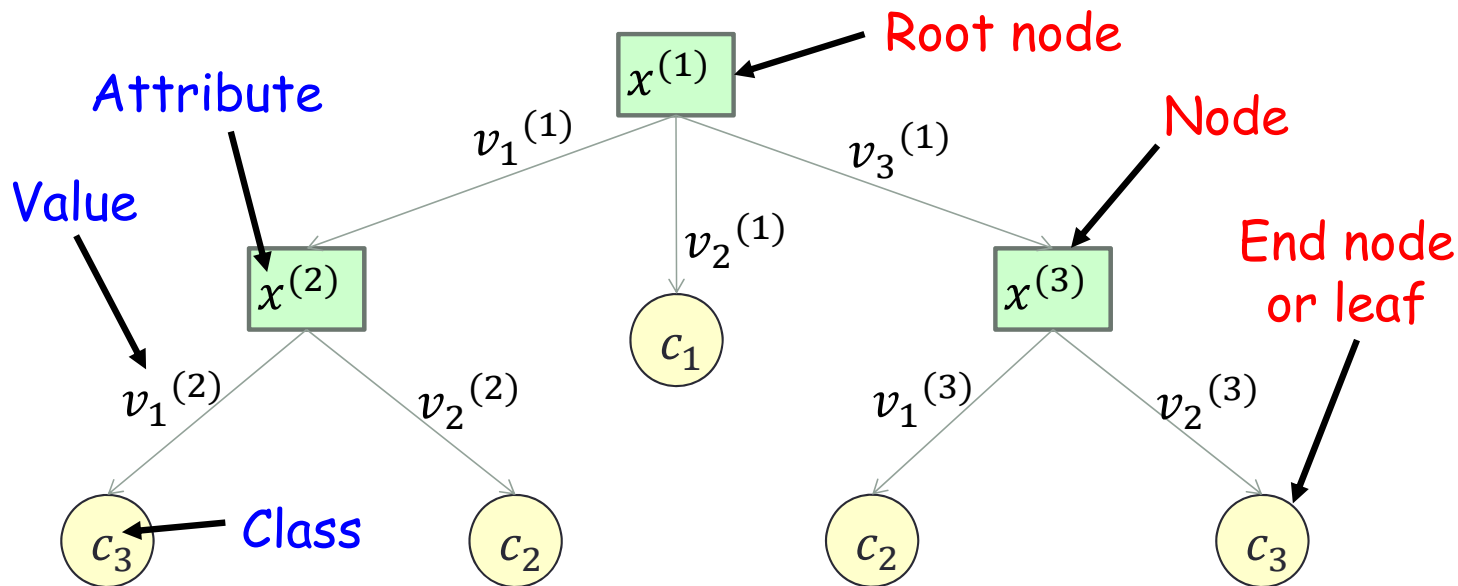
- Problem:

Given a set of training data points $(\mathbf{x}_1, y_1) \dots (\mathbf{x}_N, y_N)$, find a classifier $\hat{y}_i = f(\mathbf{x}_i, \boldsymbol{\alpha})$ that can be used to predict unseen data

- The $\boldsymbol{\alpha}$ are the parameters to be learned
- The $\mathbf{x}_i = [x^{(1)} \dots x^{(M)}]_i \in \mathcal{X}$ are called variables or attributes
- The $y_i \in \mathcal{Y}$ is called class
- Usually there are a limited number of different classes, $y_i \in \{c_1 \dots c_p\}$

Decision Trees

- A decision support tool that uses a tree-like graph or model of decisions and their possible consequences
- Best for cases with discrete-valued attributes

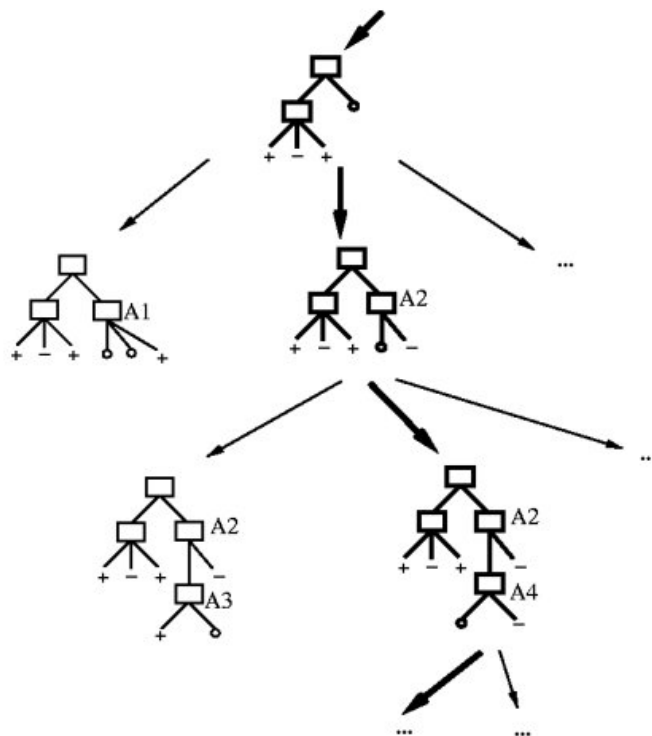


Decision Tree Example – Play Tennis

<i>Day</i>	<i>Outlook</i>	<i>Temperature</i>	<i>Humidity</i>	<i>Windy</i>	<i>Play Tennis</i>
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

Constructing Decision Tree

- Strategy: top down, recursive divide-and-conquer fashion

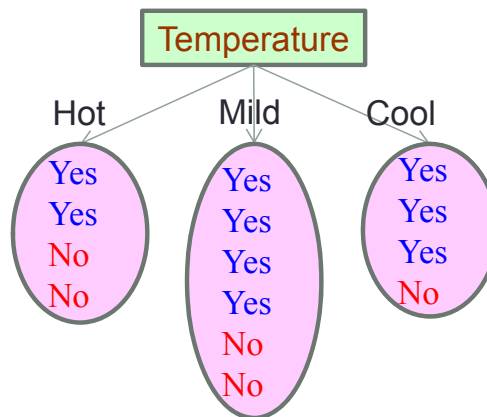
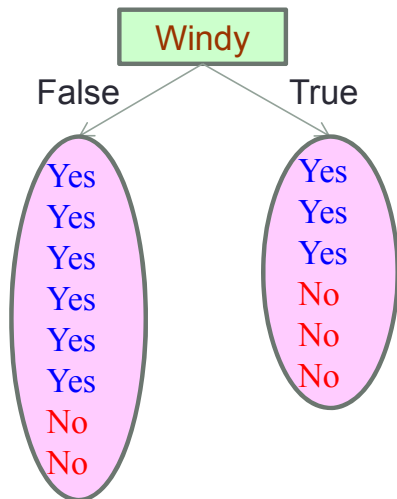
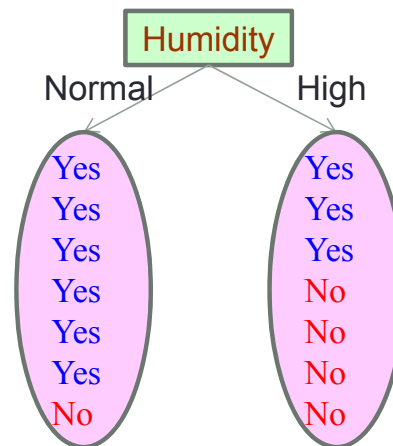
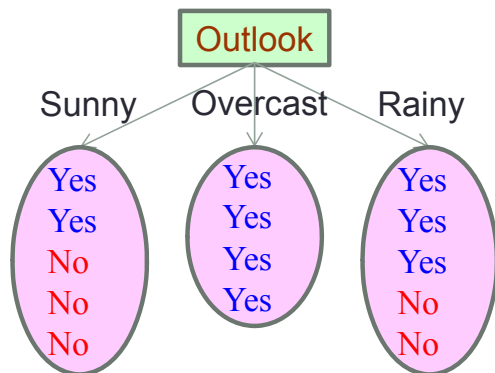


Tom Mitchell.
Machine Learning

Constructing Decision Tree (Cont'd)

1. Start with a set of training samples
2. Create a node and its branches by following these steps:
 - Select attribute for the node
 - Create branch for each possible attribute value
 - Split instances into subsets
 - One subset for each branch extending from the node
3. Repeat recursively for each branch, using only instances that reach the branch
4. Stop if all instances have the same class


First to Choose An Attribute for the Root Node



Attribute Selection

- Desired to create the smallest tree (why?)
- Which is the best attribute? In what sense?
- Intuition: choose the attribute that produces the “purest” subsets
- Properties required for a purity measure
 - When node is pure, measure should be zero
 - When impurity is maximal (i.e. all classes equally likely), measure should be maximal

Shannon's Entropy

- Proposed by Claude Shannon in 1948 at Bell Labs in the work [A Mathematical Theory of Communication](#)
 - A measure of the uncertainty associated with a random variable on a data subset
- 
- The entropy helps us to choose an attribute for a test (in a node) so that it divides samples into subsets with less uncertainty (as oppose to low purity)

Entropy Definition

- The entropy $En(K) \in \mathbb{R}$ of a data set K is:

$$En(K) = - \sum_{i=1}^q p(K, c_i) \cdot \log_2(p(K, c_i))$$

where $p(K, c_i)$ denotes proportion of samples in K that are associated with the class c_i , $i = 1 \dots q$, and $\sum p(K, c_i) = 1$

- Example:

The entropy of the play tennis data set K

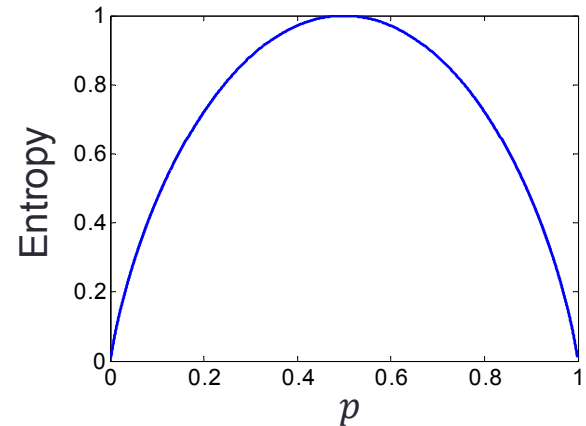
$$En(K) = - \left(\underbrace{\frac{9}{14} \log_2 \left(\frac{9}{14} \right)}_{\text{Yes}} + \underbrace{\frac{5}{14} \log_2 \left(\frac{5}{14} \right)}_{\text{No}} \right) = 0.9403$$

K

Day	Play?
D1	No
D2	No
D3	Yes
D4	Yes
D5	Yes
D6	No
D7	Yes
D8	No
D9	Yes
D10	Yes
D11	Yes
D12	Yes
D13	Yes
D14	No

Properties of Entropy

- For a set of samples K with binary classes, e.g., yes and no, the Shannon's entropy against $p(K, c_i)$, the proportion of samples in K associated with the class c_i , looks like this:



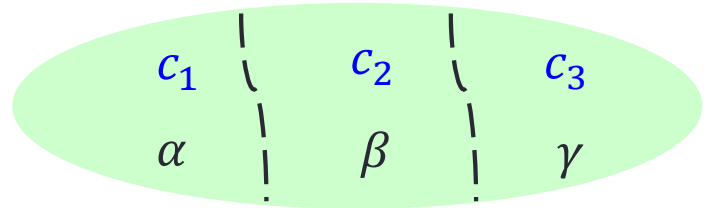
- Property:
$$En(K) = - \sum_{i=1}^2 p(K, c_i) \cdot \log_2(p(K, c_i))$$
 - Always positive
 - Maximum is 1 when $p(K, c_1) = p(K, c_2) = 1/2$
 - Minimum is 0 when $p(K, c_1) = 0$ or $p(K, c_2) = 0$

Why \log_2 ?

- A transformation from multiplication to addition (why?)

$$\log_2(ab) = \log_2 a + \log_2 b$$

- Multistage property:
- Exist a set $\{\mathbf{x}_i | \mathbf{x}_i \in c_1, c_2, c_3\}$
- $0 \leq \alpha, \beta, \gamma \leq 1, \alpha + \beta + \gamma = 1$



$$En([\alpha, \beta, \gamma]) = En([\alpha, \beta + \gamma]) + (\beta + \gamma) \cdot En\left(\left[\frac{\beta}{\beta + \gamma}, \frac{\gamma}{\beta + \gamma}\right]\right)$$

Proof of Multistage Property

- Proof:

$$En([\alpha, \beta + \gamma]) + (\beta + \gamma) \cdot En\left(\left[\frac{\beta}{\beta + \gamma}, \frac{\gamma}{\beta + \gamma}\right]\right)$$

$$\begin{aligned} &= -(\alpha \cdot \log_2(\alpha) + (\beta + \gamma) \cdot \log_2(\beta + \gamma)) \\ &\quad -(\beta + \gamma) \cdot \left(\frac{\beta}{\beta + \gamma} \cdot \log_2\left(\frac{\beta}{\beta + \gamma}\right) + \frac{\gamma}{\beta + \gamma} \cdot \log_2\left(\frac{\gamma}{\beta + \gamma}\right)\right) \end{aligned}$$

$$= -(\alpha \cdot \log_2(\alpha) + (\beta + \gamma) \cdot \log_2(\beta + \gamma) + \beta \cdot \log_2\left(\frac{\beta}{\beta + \gamma}\right) + \gamma \cdot \log_2\left(\frac{\gamma}{\beta + \gamma}\right))$$

$$= -(\alpha \cdot \log_2(\alpha) + \beta \cdot \log_2(\beta) + \gamma \cdot \log_2(\gamma)) = En([\alpha, \beta, \gamma])$$

Properties of Entropy (Cont'd)

- Example:

$$En([0.2, 0.3, 0.5]) = En([.2, .8]) + .8 \cdot En\left(\left[\frac{3}{8}, \frac{5}{8}\right]\right)$$

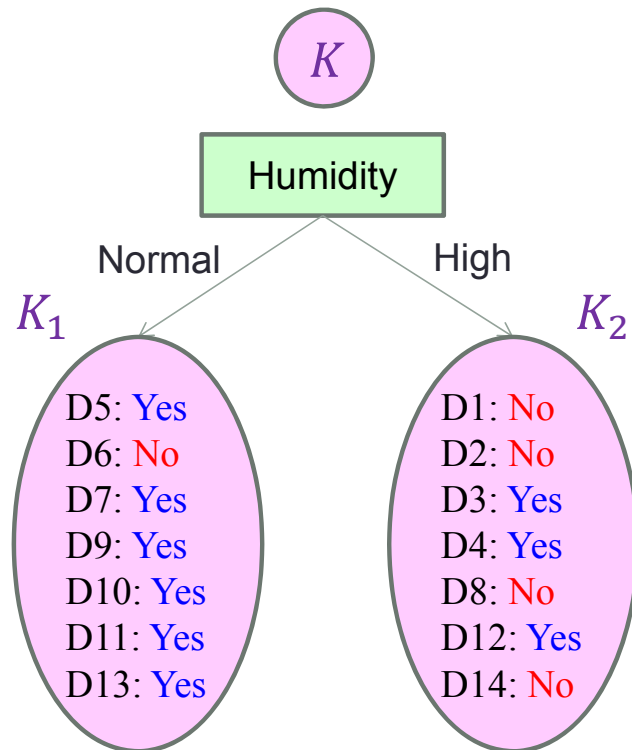
- Note that for 3-class cases, the maximum entropy is not 1

$$En = -\left(\frac{1}{3} \log_2 \left(\frac{1}{3}\right) + \frac{1}{3} \log_2 \left(\frac{1}{3}\right) + \frac{1}{3} \log_2 \left(\frac{1}{3}\right)\right) = 1.585$$

- Other properties hold though

Play Tennis Example (1)

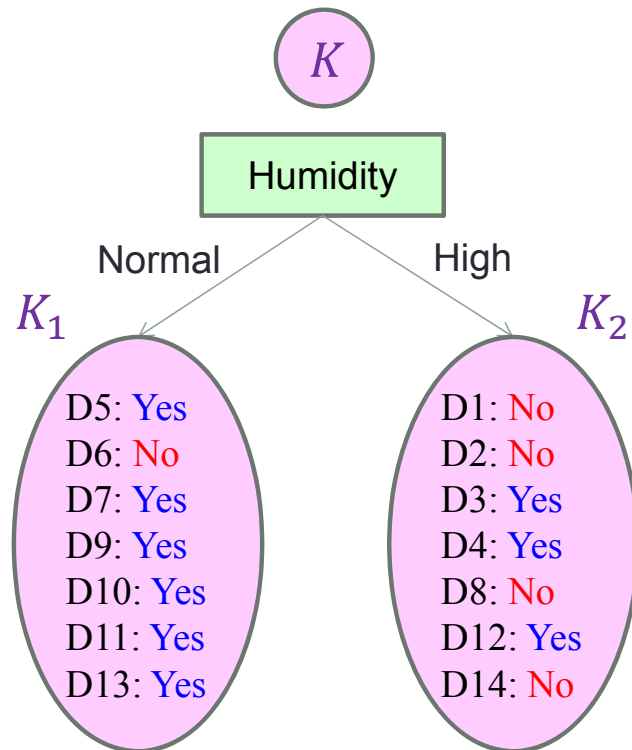
- Chose “Humidity” attribute for the root node



<i>Day</i>	<i>Humidity</i>	<i>Play?</i>
D1	High	No
D2	High	No
D3	High	Yes
D4	High	Yes
D5	Normal	Yes
D6	Normal	No
D7	Normal	Yes
D8	High	No
D9	Normal	Yes
D10	Normal	Yes
D11	Normal	Yes
D12	High	Yes
D13	Normal	Yes
D14	High	No

Play Tennis Example (2)

- Chose “Humidity” attribute for the root node



$$\begin{aligned} En(K_1) &= -\left(\frac{6}{7}\log_2\left(\frac{6}{7}\right) + \frac{1}{7}\log_2\left(\frac{1}{7}\right)\right) \\ &= 0.5917 \end{aligned}$$

$$\begin{aligned} En(K_2) &= -\left(\frac{3}{7}\log_2\left(\frac{3}{7}\right) + \frac{4}{7}\log_2\left(\frac{4}{7}\right)\right) \\ &= 0.9852 \end{aligned}$$

Iterative Dichotomiser (ID) 3

- A “greedy” algorithm for decision tree construction developed by Ross Quinlan in 1987
- ID3 follows the principle of Occam's razor in attempting to create the smallest decision tree possible
- Greedy means not backtracking
- [Personal website](#)



- Using information gain to select the "best attribute" for each node

Information Gain

- Information gain is defined as the information difference before and after a test with an attribute x :

$$Gain(K, x) = En(K) - En(K, x)$$

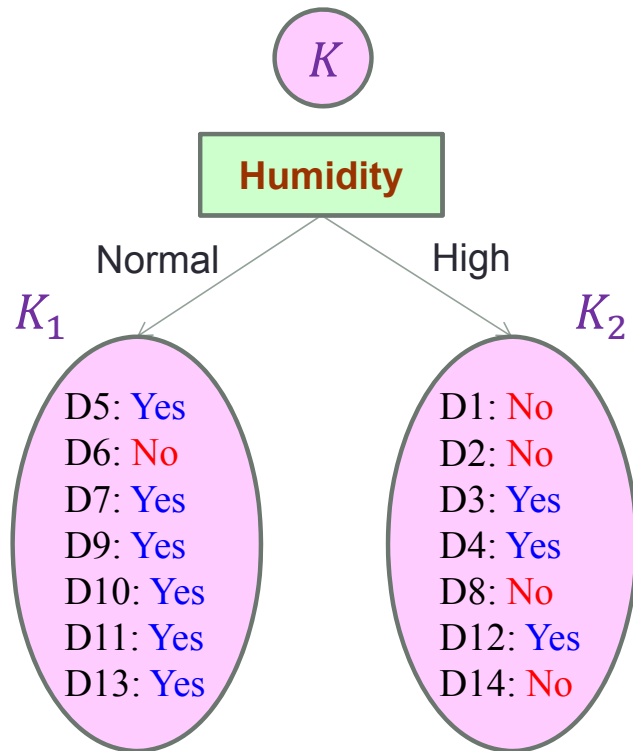
where

$$En(K, x) = \sum_{i=1}^r \frac{|K_i|}{|K|} \cdot En(K_i)$$

is the weighted sum of entropies over the subsets

- Note: instead of maximizing info gain we could just minimize information

Play Tennis Example (3)



$$En(K) = 0.9403$$

$$En(K_1) = 0.5917$$

$$En(K_2) = 0.9852$$

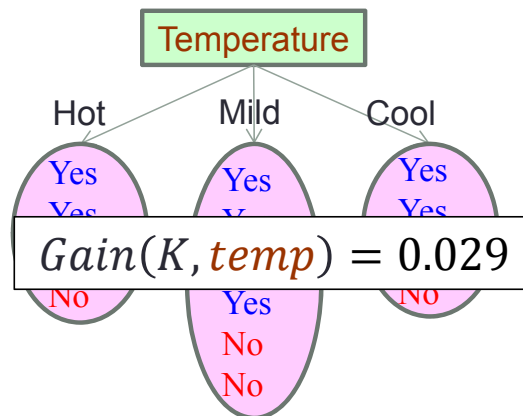
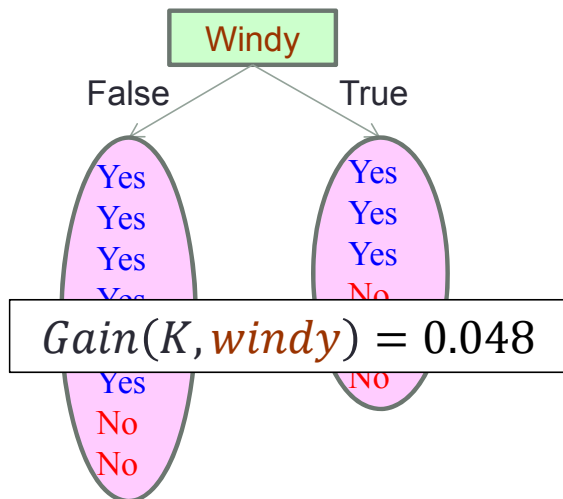
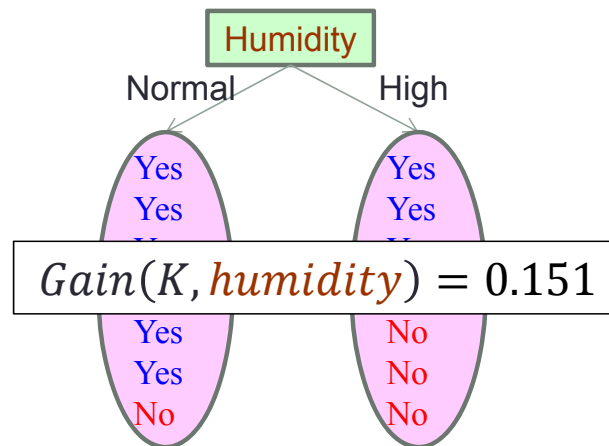
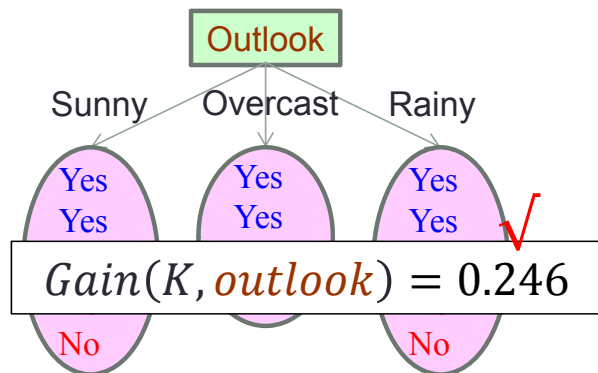
$$Gain(K, \text{humidity})$$

$$= En(K) - \frac{7}{14} En(K_1) - \frac{7}{14} En(K_2)$$

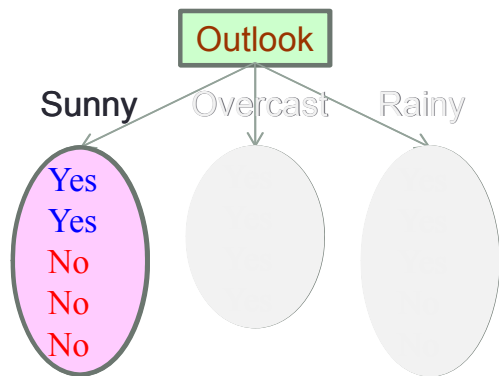
$$= 0.1518$$

Is information-gain always a positive value? Why or why not?

Play Tennis Example (4)



“Recursively” Choose Attributes

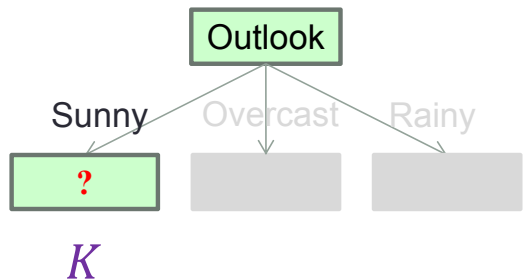


K

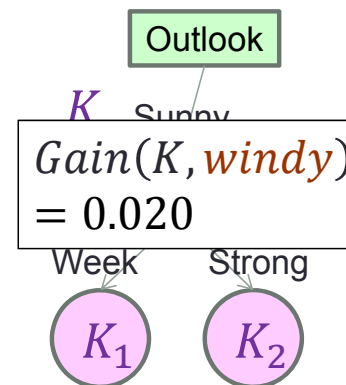
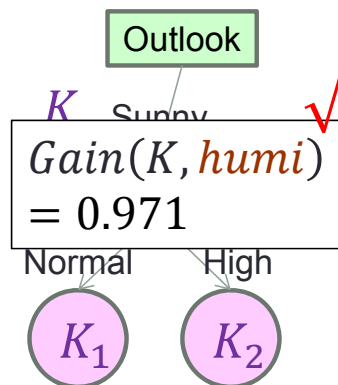
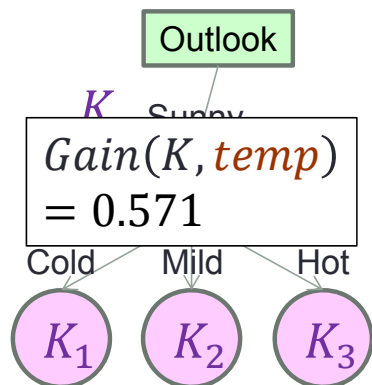
$$\begin{aligned}
 En(K) &= -\left(\frac{2}{5}\log_2\left(\frac{2}{5}\right)\right) \\
 &\quad + \frac{3}{5}\log_2\left(\frac{3}{5}\right) \\
 &= 0.9710
 \end{aligned}$$

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

Play Tennis Example (5)

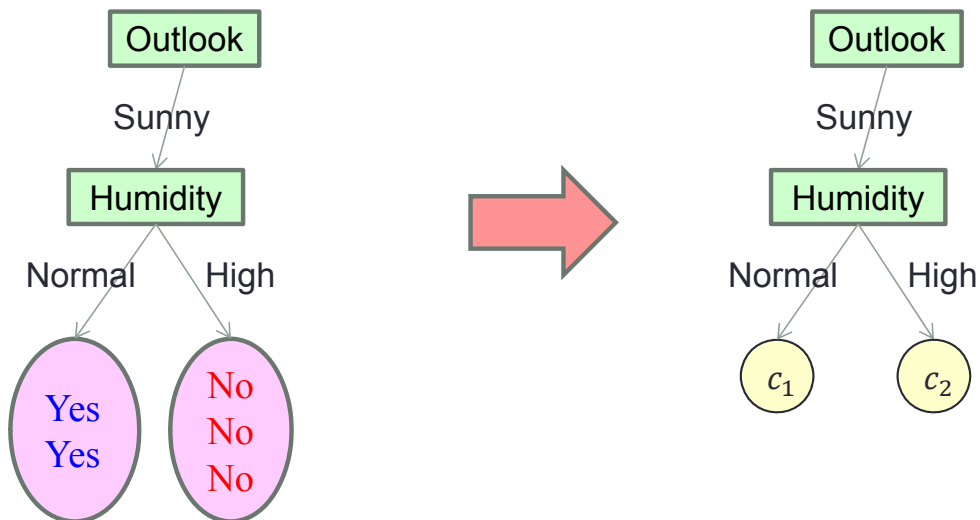


Day	Outlook	Temperature	Humidity	Windy	Play?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes



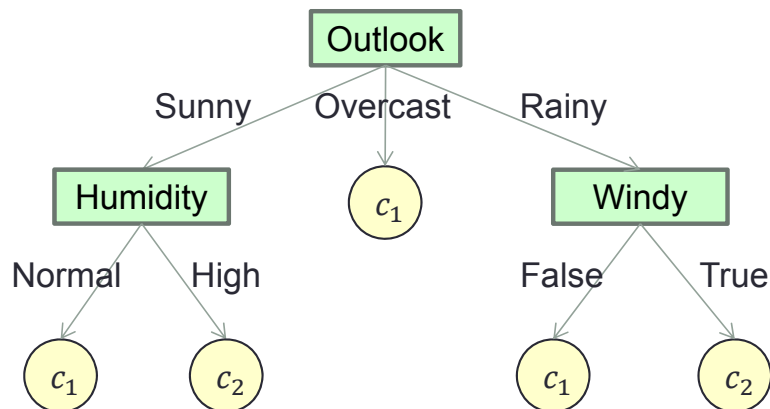
Play Tennis Example (6)

- Stop criteria: all instances have the same class
- Set the output to the end nodes to the class associated with the samples
- Let c_1 and c_2 represent yes and no, respectively

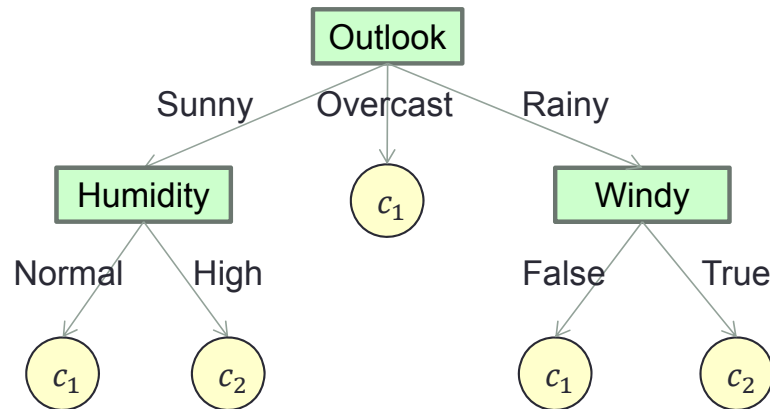


Play Tennis Example (7)

- Recursively separate the training samples until they cannot be split any further

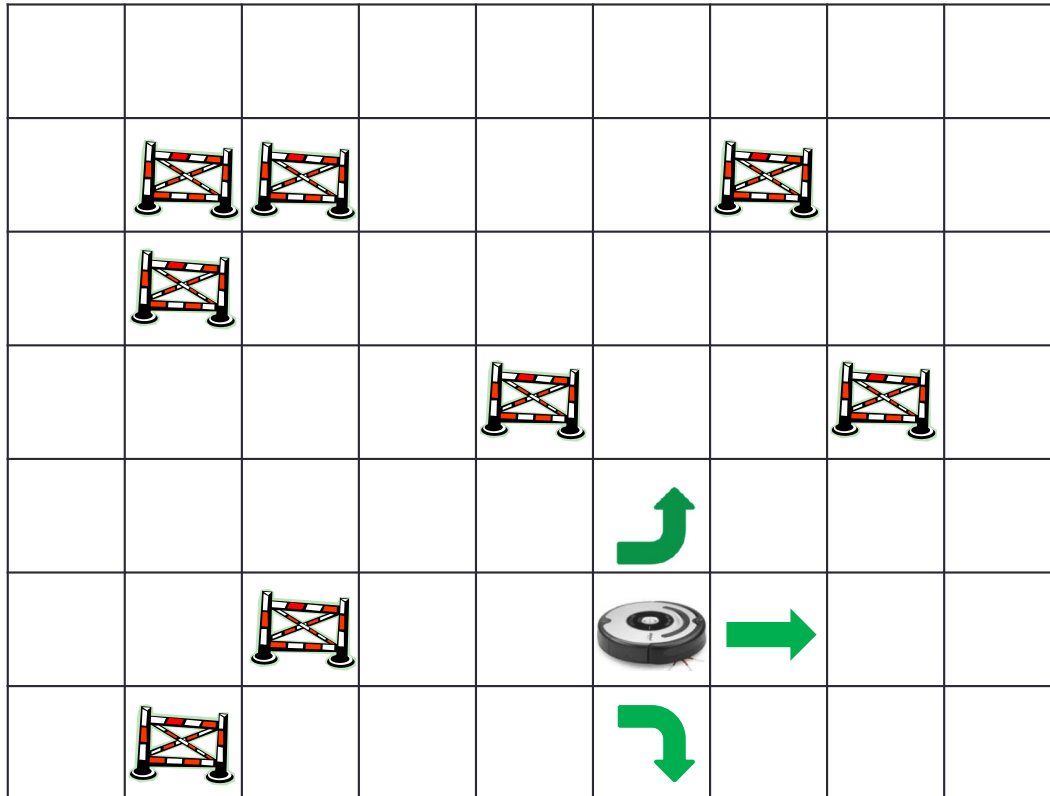


Convert A Decision Tree To Rules



- If (outlook = sunny) & (humidity = high) then **No**
- If (outlook = sunny) & (humidity = normal) then **Yes**
- If (outlook = overcast) then **Yes**
- If (outlook = rain) & (wind = strong) then **No**
- If (outlook = rain) & (wind = weak) then **Yes**

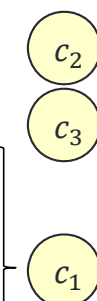
Example – Robot State Machine



Robot State Machine Example (1)

- Four sensors and three actions
- Objective is to decide which sensor to sense first

<i>State</i>	<i>Left sensor</i>	<i>Right sensor</i>	<i>Forward sensor</i>	<i>Back sensor</i>	<i>Action</i>
x1	Obstacle	Free	Obstacle	Free	Turn right
x2	Free	Free	Obstacle	Free	Turn left
x3	Free	Obstacle	Free	Free	Move forward
x4	Free	Obstacle	Free	Obstacle	Move forward
x5	Obstacle	Free	Free	Free	Move forward
x6	Free	Free	Free	Obstacle	Move forward



Robot State Machine Example (2)

$$En(K) = -\frac{1}{6}\log_2\left(\frac{1}{6}\right) - \frac{1}{6}\cdot\log_2\left(\frac{1}{6}\right) - \frac{4}{6}\cdot\log_2\left(\frac{4}{6}\right) = 1.25$$

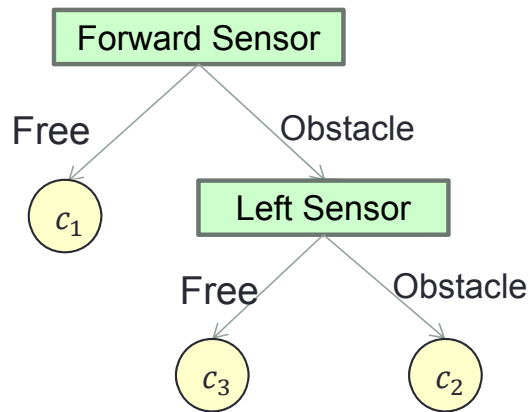
$$\begin{aligned} Gain(Left) &= En(K) - \frac{2}{6}En(Left = obstacle) - \frac{4}{6}En(Left = free) \\ &= 1.25 - \frac{2}{6}\cdot 1 - \frac{4}{6}\cdot 0.811 = 0.326 \end{aligned}$$

$$\begin{aligned} Gain(Right) &= En(K) - \frac{2}{6}En(Right = obstacle) - \frac{4}{6}En(Right = free) \\ &= 1.25 - \frac{2}{6}\cdot 0 - \frac{4}{6}\cdot 1.5 = 0.25 \end{aligned}$$

$$\begin{aligned} Gain(Forward) &= En(K) - \frac{2}{6}En(Forward = obstacle) - \frac{4}{6}En(Forward = free) \\ &= 1.25 - \frac{2}{6}\cdot 1 - \frac{4}{6}\cdot 0 = 0.917 \quad \checkmark \end{aligned}$$

$$\begin{aligned} Gain(Back) &= En(K) - \frac{2}{6}En(Back = obstacle) - \frac{4}{6}En(Back = free) \\ &= 1.25 - \frac{2}{6}\cdot 0 - \frac{4}{6}\cdot 1.5 = 0.25 \end{aligned}$$

Robot State Machine Example (3)



$$En(K) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$$

$$\begin{aligned}
 Gain(Left) &= En(K) - \frac{1}{2} \cdot En(Left = free) \\
 &\quad - \frac{1}{2} \cdot En(Left = obstacle) \\
 &= 1 - \frac{1}{2} \cdot 1 - \frac{1}{2} \cdot 0 - \frac{1}{2} \cdot 0 \\
 &= 1
 \end{aligned}$$

$$\begin{aligned}
 Gain(Right) &= En(K) - 1 \cdot En(Right = 0) \\
 &= 1 - 1 \\
 &= 0
 \end{aligned}$$

Some Issues with ID3

- What if there exists continuous-valued attributes?
- Is information gain the best evaluation index?
- More decision tree algorithms:
 - **C4.5** Quinlan (1993)
 - **C5.0** Quinlan
 - **Cubist** Quinlan
 - **CART** Breiman (1984)



Dealing with Continuous-valued Attributes

- Splitting training samples with a continuous-valued attribute potentially results in infinitely many subsets
- Suppose temperature in play tennis are recorded as

Temperature (°F)	48.3	40.1	60.2	90.4	80.2	71.9
<i>Play Tennis</i>	No	No	Yes	No	Yes	Yes

- One method is to use domain knowledge divide temperature into cool ($\leq 60^\circ\text{F}$), mild ($> 60^\circ\text{F}$ and $\leq 90^\circ\text{F}$), and hot ($> 90^\circ\text{F}$)
- Strategy of C4.5: find a threshold to split the training samples into two subsets

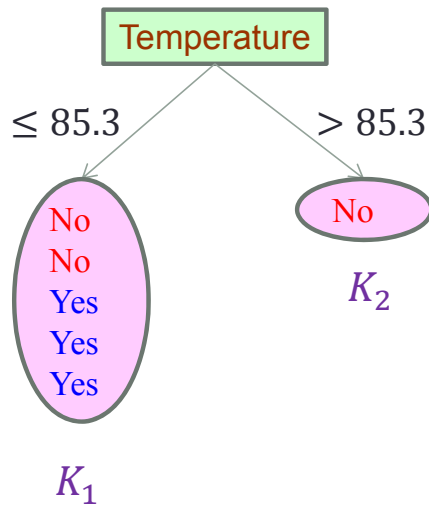
Dealing with Continuous-valued Attributes (Cont'd)

- What is the best threshold?
- Procedure for threshold search:
 1. Sort the attribute values in an ascending order
 2. The midpoint between each pair of adjacent values is considered as a split candidate points
 3. Calculate the weighted average entropies over the subsets that are split using the candidate points
 4. The point that gives the least entropy is selected as the threshold

Example – Numbered Temperature

Temperature (°F)	40.1	48.3	60.2	71.9	80.2	90.4
<i>Play Tennis</i>	No	No	Yes	Yes	Yes	No

\uparrow \uparrow \uparrow \uparrow \uparrow
 (44.2) (54.3) (66.1) (76.1) (85.3)



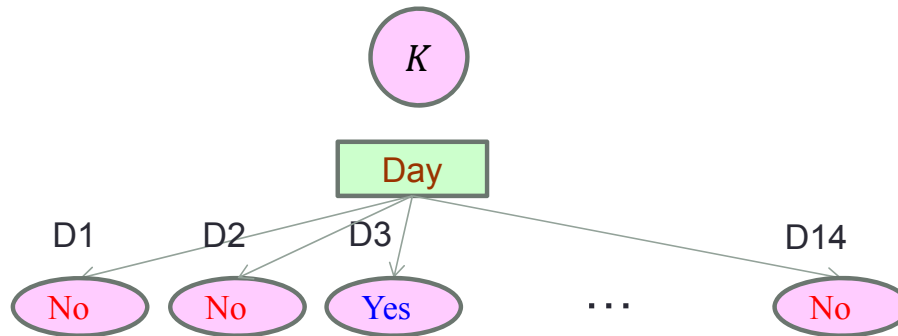
$$\begin{aligned}
 \bullet \text{ } En(K_1) &= -\left(\frac{3}{5}\log_2\left(\frac{3}{5}\right) + \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right) \\
 &= 0.971
 \end{aligned}$$

$$\bullet \text{ } En(K_2) = 0$$

$$\bullet \text{ } En(K, temp = 44.2) = \frac{5}{6} \cdot 0.971 = 0.809$$

Problem with Highly-branching Attributes

- Using attributes with a large number of values to split data is more likely to create “pure” subsets
- This may result in overfitting



$$En(K, \text{day}) = \sum_{i=1}^{14} \frac{1}{14} \cdot \log_2\left(\frac{1}{1}\right) = 0$$

Minimum entropy!!!

Day	Play Tennis
D1	No
D2	No
D3	Yes
D4	Yes
D5	Yes
D6	No
D7	Yes
D8	No
D9	Yes
D10	Yes
D11	Yes
D12	Yes
D13	Yes
D14	No

Split: the Intrinsic Information of Subsets

- The $Split(K, x)$ of the data set K using the attribute x

$$Split(K, x) = - \sum_{i=1}^r \frac{|K_i|}{|K|} \cdot \log_2 \left(\frac{|K_i|}{|K|} \right)$$

is a measure of number and size of split branches, i.e., intrinsic information

- It does NOT consider the sample classes in the subsets
- The more branches, the larger the $Split(K, x)$

- Example: $Split(K, \text{day}) = - \sum_{i=1}^{14} \frac{1}{14} \cdot \log_2 \left(\frac{1}{14} \right) = 3.807$

Gain Ratio for Attribute Selection (C4.5)

- Information gain is biased towards attributes with a large number of distinct values
- Gain ratio (GR) penalizes the bias by normalizing the gain ratio against $Split(K, x)$:

$$GR(K, x) = \frac{Gain(K, x)}{Split(K, x)}$$

Gain Ratios for the Play Tennis Example

Outlook		Temperature	
Entropy:	0.693	Entropy:	0.911
Gain: $0.940 - 0.693$	0.247	Gain: $0.940 - 0.911$	0.029
Split: $\text{En}([5,4,5])$	1.577	Split: $\text{En}([4,6,4])$	1.362
Gain ratio: $0.247/1.577$	0.156	Gain ratio: $0.029/1.362$	0.021

Humidity		Windy	
Entropy:	0.788	Entropy:	0.892
Gain: $0.940 - 0.788$	0.152	Gain: $0.940 - 0.892$	0.048
Split: $\text{En}([7,7])$	1.000	Split: $\text{En}([8,6])$	0.985
Gain ratio: $0.152/1$	0.152	Gain ratio: $0.048/0.985$	0.049

More on the Gain Ratio

- Gain ratio is expected to be large when data is evenly spread, and expected to be small when most samples belong to one branch
- Note that gain ratio may result in overcompensation, which means, it may choose an attribute just because its intrinsic information is very low
- One way to avoid over-compensation – only consider attributes with greater than average information gain

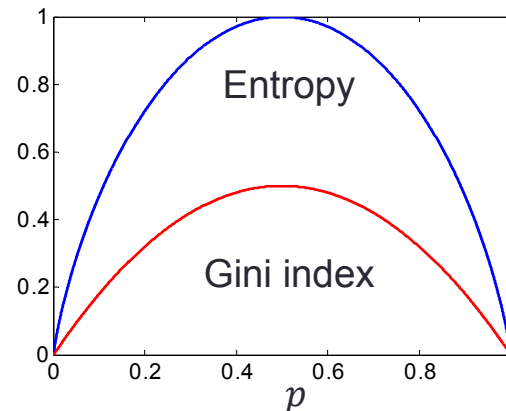
Gini Index (CART)

- The Gini index $Gini(K) \in \mathbb{R}$ of a data set K is:

$$Gini(K) = 1 - \sum_{i=1}^q p(K, c_i)^2$$

where $p(K, c_i)$ denotes proportion of samples in K that are associated with the class c_i , $i = 1 \dots q$, and $\sum p(K, c_i) = 1$

- Property:
 - Always positive
 - Maximum is $1/2$
 - Minimum is 0



Gini Gain

- Gini gain is defined as the Gini index difference before and after a test with an attribute x :

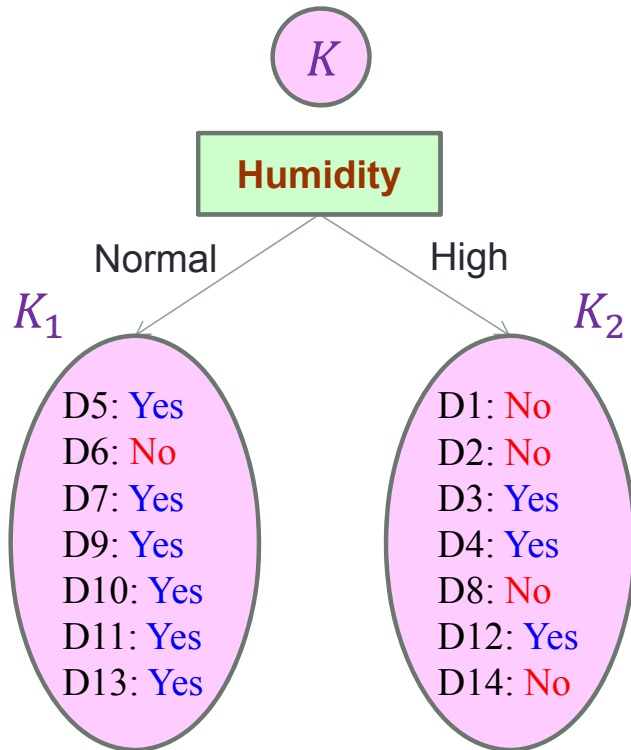
$$GiniGain(K, x) = Gini(K) - Gini(K, x)$$

where

$$Gini(K, x) = \sum_{i=1}^r \frac{|K_i|}{|K|} \cdot Gini(K_i)$$

is the weighted sum of Gini index over the subsets

Play Tennis Example (8)



$$Gini(K) = 1 - \left(\frac{5}{14}\right)^2 - \left(\frac{9}{14}\right)^2$$

$$= 0.459$$

$$Gini(K_1) = 1 - \left(\frac{6}{7}\right)^2 - \left(\frac{1}{7}\right)^2$$

$$= 0.245$$

$$Gini(K_2) = 1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2$$

$$= 0.490$$

$$GiniGain(K, \text{humidity})$$

$$= En(K) - \frac{7}{14}En(K_1) - \frac{7}{14}En(K_2)$$

$$= 0.092$$

Attribute Selection with Cost

- The availability of attributes may vary significantly in costs
- For example, medical diagnosis such as

Temperature
Pulse

Biopsy
Blood test

- In practice, it is desired to postpone acquisition of such high cost attribute values until they become necessary
- Attribute selection measures that penalize against cost
- Tan and Schlimmer (1990) Nunez (1988)

$$\frac{Gain^2(K, x)}{Cost(x)} \quad \frac{2^{Gain(K, x)} - 1}{(Cost(x) + 1)^w}, \quad w \in [0, 1]$$

Comparison of Attribute Selection Measures

- Information gain:

Biases towards multivalued attributes

- Gain ratio:

Tends to prefer unbalanced splits in which one partition is much smaller than the others

- Gini index:

Biases to multivalued attributes

Tends to favor tests that result in equal-sized partitions and purity in both partitions

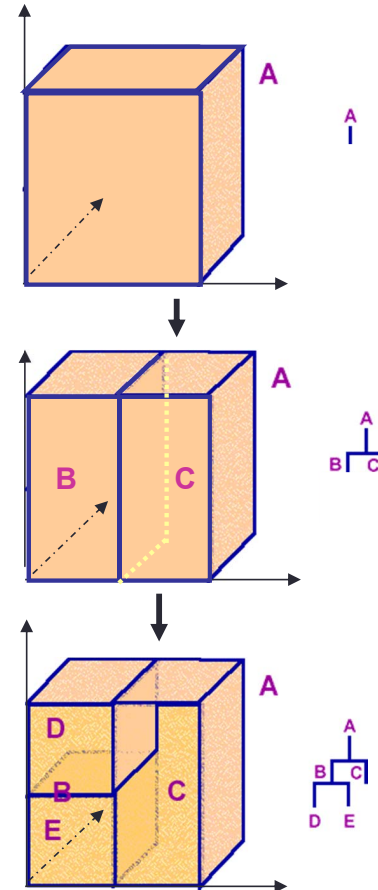
Other Attribute Selection Measures

- CHAID: a measure based on [\$\chi^2\$ test](#) for independence
- G-statistic: a close approximation to χ^2 distribution
- C-SEP
- MDL
- Which attribute selection measure is the best?

Ans: none is significantly superior than others

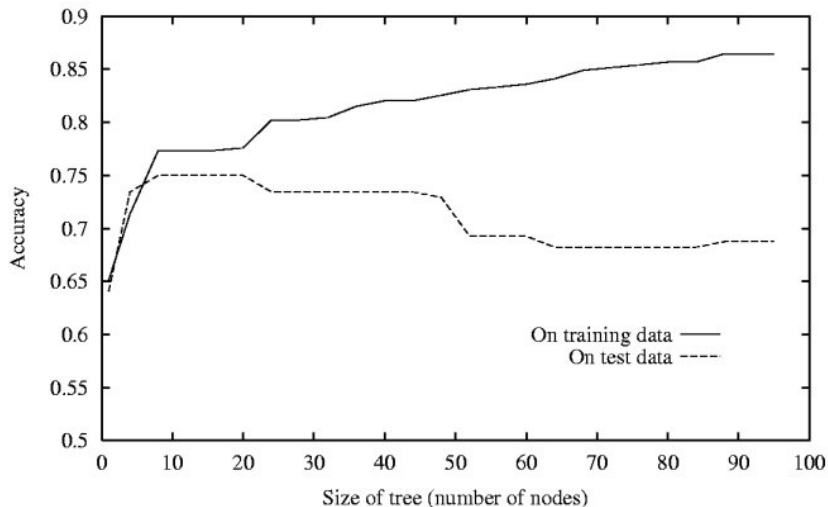
Geometrical View of Decision Tree Growing

- Assumed binary tree
- Border line between two neighboring regions is parallel to axes because test condition involves a single attribute at a time
- Each node draws a boundary that can be geometrically interpreted as a hyperplane perpendicular to the axis



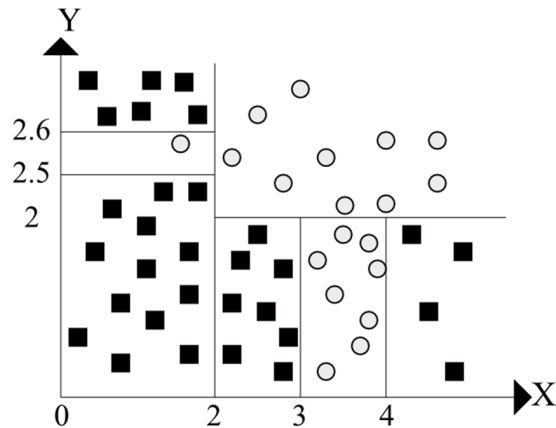
Decision Tree Overfitting

- Decision trees can be learned to perfectly fit the data
- Overfitting can occur with noisy training examples, or with small numbers of (coincidental or accidental) examples
- Overfitted tree provides poor accuracy for unseen data

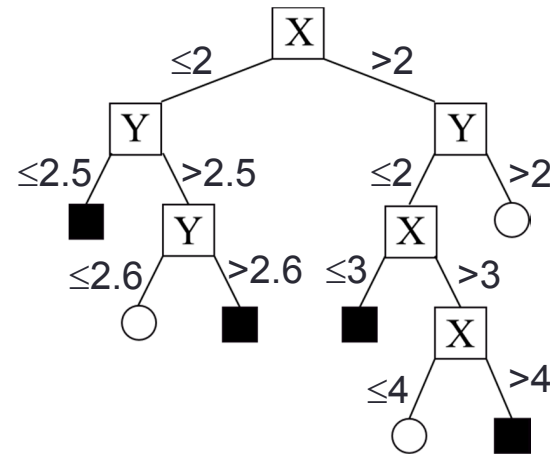


Tom Mitchell.
Machine Learning

Geometrically Explanation



(A) A partition of the data space



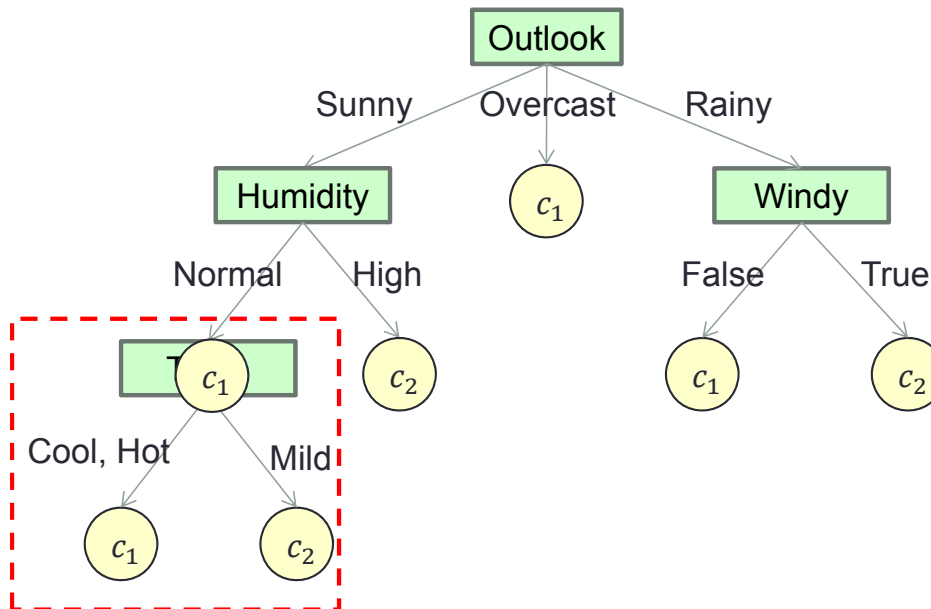
(B) The decision tree

Play Tennis Example (9)

<i>Day</i>	<i>Outlook</i>	<i>Temperature</i>	<i>Humidity</i>	<i>Windy</i>	<i>Play Tennis</i>
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
D15	Sunny	Mild	Normal	Strong	No

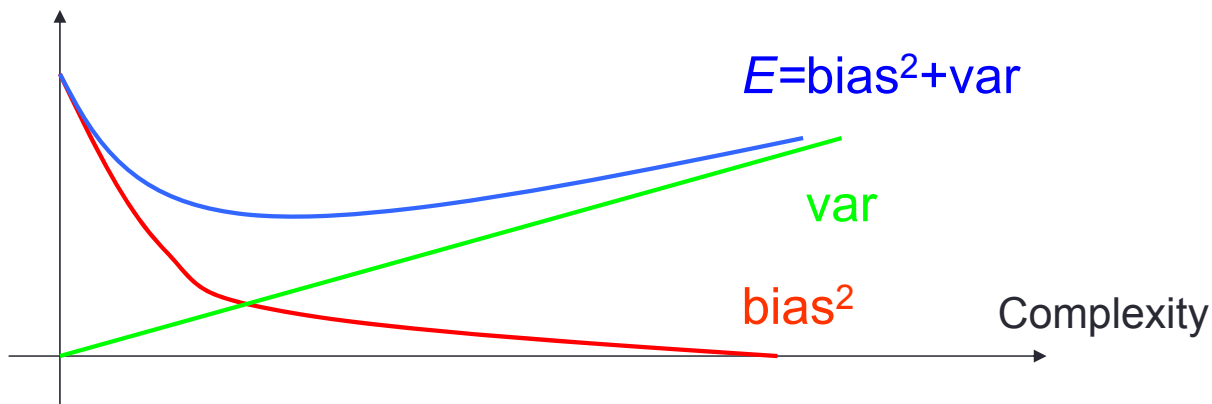
Play Tennis Example (9)

- An overfitted tree



Complexity of the Tree

- Usually, the bias is a decreasing function of the complexity, while variance is an increasing function of the complexity



How to Avoid Overfitting

- Pruning – a technique to reduce the number of attributes used in a tree
- Two types of pruning:
 - Pre-pruning (forward pruning): Stop growing a tree before it becomes a fully-grown tree
 - Post-pruning (backward pruning): Completely grow the tree, then remove branches that may not improve prediction accuracy for unseen samples

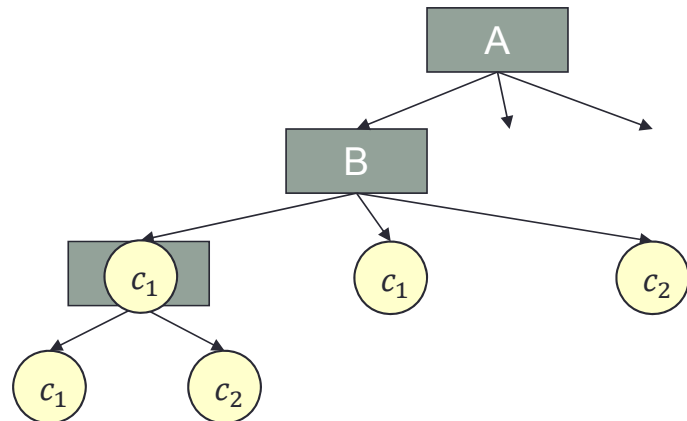
Pre-pruning

- Typical stopping conditions for a node:
 - All instances belong to the same class
 - All the attribute values are the same
- Pre-pruning stopping conditions:
 - Number of instances is less than some user-specified threshold
 - Expanding the current node does not improve impurity measures (e.g., Gini or information gain)
 - Class distribution of instances are independent of the available features (e.g., using χ^2 test)

Post-pruning

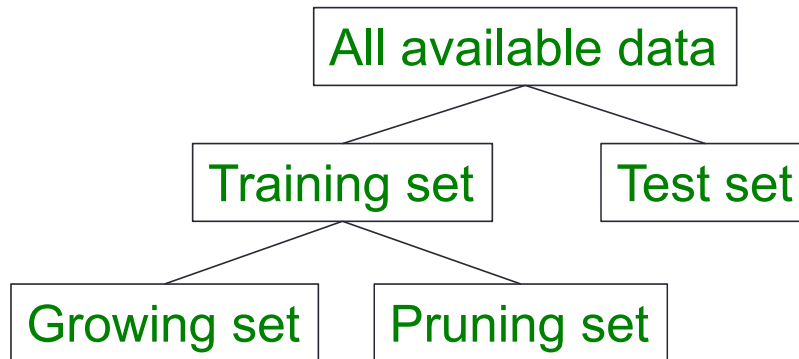
- Trim the nodes of the fully-grown decision tree in a bottom-up fashion
- Replace a sub-tree by a leaf node if generalization error improves after trimming
- Class label of leaf node is determined from majority class of instances in the sub-tree

How does one know whether pruning a sub-tree improve the prediction accuracy?



Estimating Accuracy of Sub-trees

- If one has enough data, partitioning data into:



Typical choice:
Training set: 70%
Test set 30%
Growing set 70%
Pruning set 30%

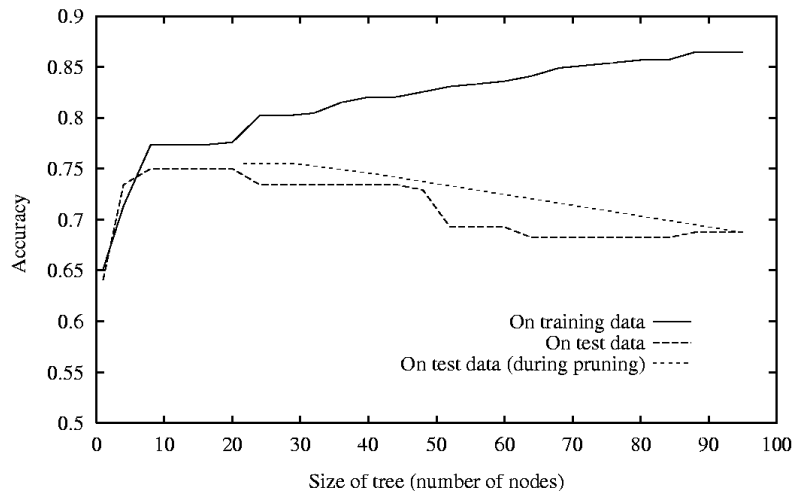
- In case of shortage of data, try estimate accuracy directly from learning data

Reduced-error Pruning

- Procedure ([Quinlan, 1986](#)):
 1. Split the data into training and validation set
 2. Build a decision tree using the training set, allowing overfitting to occur
 3. For each node in tree:
 - Consider effect of removing sub-tree rooted at a final node, making it a final node, and assigning it majority classification
 - Check the performance of pruned tree over validation set
 4. Replace the sub-tree with a node if pruning improves the accuracy over validation set
 5. Repeat steps 3 and 4 until further pruning is harmful

Reduced-error Pruning (Cont'd)

- Impact of reduced-error pruning:

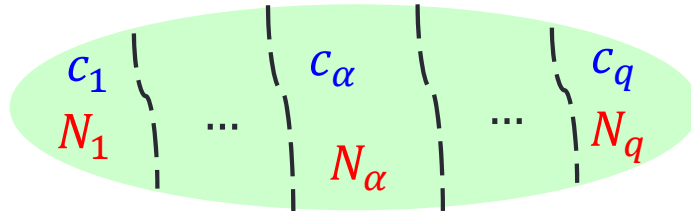


Tom Mitchell.
Machine Learning

- What if data is limited?
- Try estimate accuracy directly from learning data

Minimum-error Pruning (Niblett and Bratko)

- Assume a data set K of with c_i classes, $i = 1 \dots q$
- Each class is associated with N_i samples, i.e., $\sum N_i = |K|$
- The majority samples are associated with class c_α

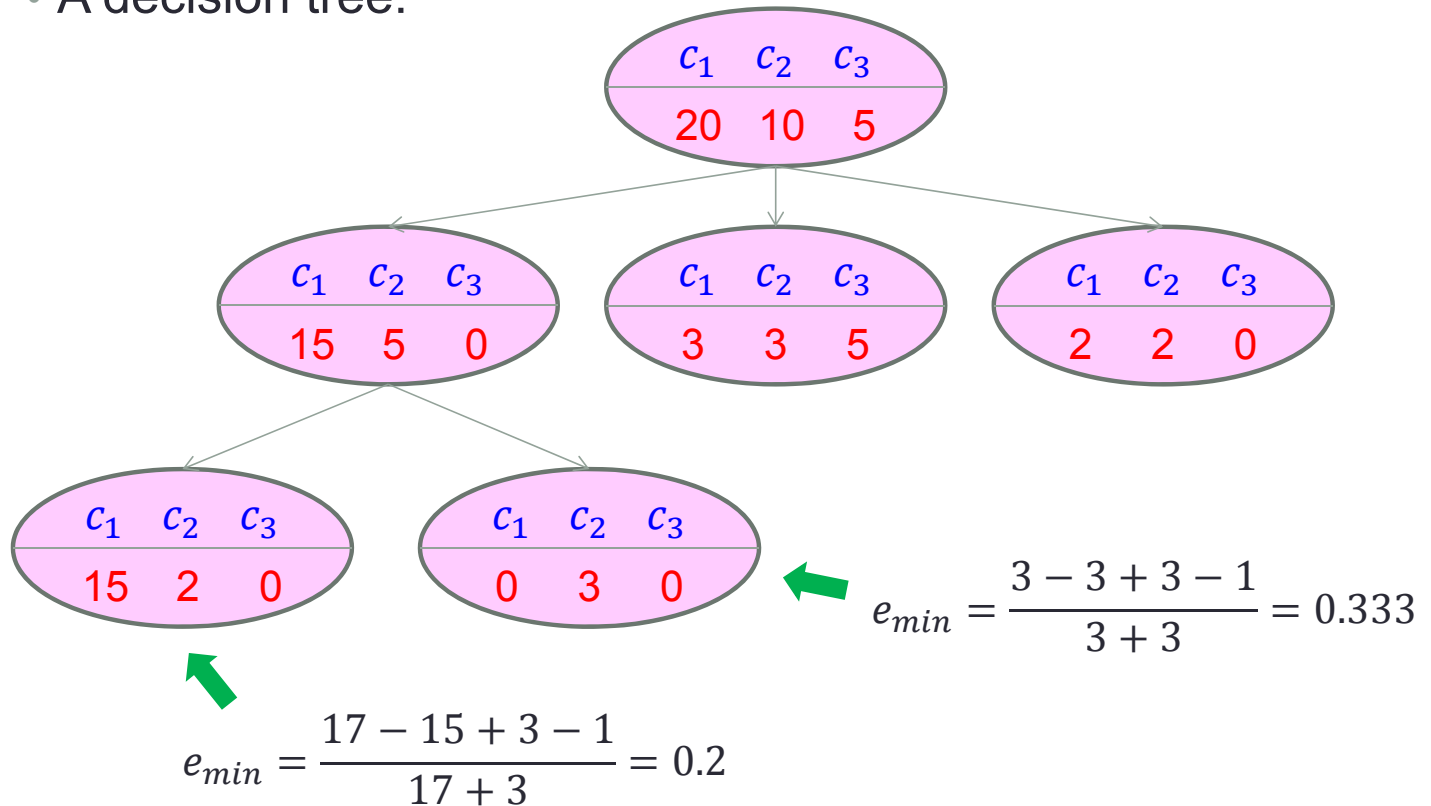


- The minimum error rate e_{min} for classifying “unseen” sets of data as c_α is

$$e_{min}(K) = \frac{|K| - N_\alpha + q - 1}{|K| + q}$$


Example

- A decision tree:



Minimum-error Pruning (Cont'd)

- Suppose a sub-tree contains $K_1 \dots K_r$ leaf nodes
- The sub-tree is replaced by a leaf node if the weighted average minimum error rate before pruning is larger than the minimum error rate after pruning, i.e.,

$$e_{min}(K) \leq \sum_{i=1}^r \frac{|K_i|}{|K|} e_{min}(K_i)$$


After
Pruning

Before
Pruning

Example

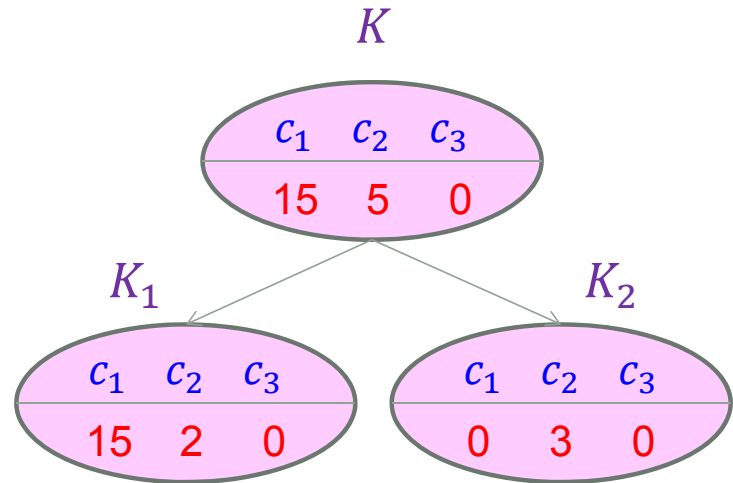
- After pruning

$$e_{min}(K) = \frac{20 - 15 + 3 - 1}{20 + 3} = 0.304$$

- Before pruning

$$\frac{17}{20} e_{min}(K_1) + \frac{3}{20} e_{min}(K_2) = 0.220$$

- Pruning or not?



Pessimistic Pruning (Quinlan)

- For a leaf node with data set K_i , the pessimistic error rate $e_{pes}(K_i)$ using the continuity correction for the binomial distribution is:

$$e_{pes}(K_i) = \frac{|K_i| - N_{i\alpha} + 1/2}{|K_i|}$$

- The pessimistic error rate for a sub-tree with r leaf nodes and a set of samples K , where $K = \sum_{i=1}^r K_i$, is:

$$e_{pes}(K) = \frac{\sum_{i=1}^r (|K_i| - N_{i\alpha}) + r/2}{|K|}$$

Pessimistic Pruning (Quinlan)

- The standard error $StdErr_{pes}(K)$ for the number of the pessimistic error of a sub-tree with sample set K is:

$$StdErr_{pes}(K) = \sqrt{\frac{e_{pes}(K) \cdot |K| \cdot (|K| - e_{pes}(K) \cdot |K|)}{|K|}}$$

- Prune the sub-tree if the pessimistic error of the sub-tree is smaller than the sum of pessimistic error over leaf nodes plus the standard error:

$$\underbrace{e_{pes}(K) \cdot |K|}_{\text{After Pruning}} < \underbrace{\sum_{i=1}^r e_{pes}(K_i) \cdot |K_i| + StdErr_{pes}(K)}_{\text{Before Pruning}}$$

Example

- Before pruning

$$e_{pes}(K_1) = \frac{17 - 15 + 1/2}{17} = 0.147$$

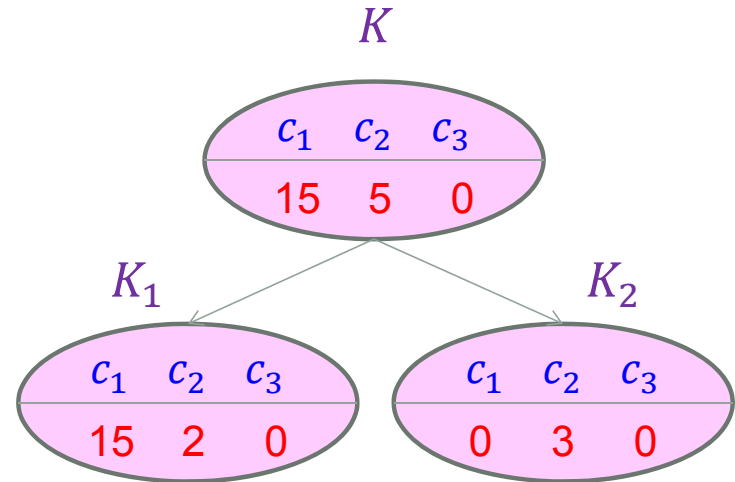
$$e_{pes}(K_2) = \frac{3 - 3 + 1/2}{3} = 0.166$$

- After pruning

$$e_{pes}(K) = \frac{20 - 15 + 1/2}{20} = 0.275$$

- Standard error

$$\begin{aligned} & StdErr_{pes}(K) \\ &= \sqrt{e_{pes}(K) \cdot |K| \cdot (1 - e_{pes}(K))} \\ &= 1.997 \end{aligned}$$



- After pruning

$$0.275 \cdot 20 = 5.5$$

- Before pruning

$$\begin{aligned} & 0.147 \cdot 17 + 0.166 \cdot 3 + 1.997 \\ &= 5.991 \end{aligned}$$

- Pruning or not?

Cost-based Pruning

- In many practical applications, decision tree prediction errors are associated with different costs
- Common errors:
 - False negative error
 - False positive error
- Let $\varepsilon(c_i, c_j) \in \mathbb{R}$ denote the misclassification cost (or penalty) of falsely predicting a sample as the class c_i when in fact it belongs to the class c_j , where $\varepsilon(c_i, c_j) > 0$ if $i \neq j$ and $\varepsilon(c_i, c_j) = 0$ otherwise

Cost-based Pruning (Cont'd)

- At a final node associated with a training sample subset K , the cost of using c_i as the output calibration action is

$$\text{Cost}(K, c_i) = |K| \cdot \sum_{j=1}^q \varepsilon(c_i, c_j) \tilde{p}(K, c_j)$$

where $\tilde{p}(K, c_j)$ is the Laplace correction of training sample proportion associated with the class c_j

- The Laplace correction sample proportion is defined as:

$$\tilde{p}(K, c_j) = \frac{p(K, c_j) \cdot |K| + 1}{|K| + q}$$

Cost-based Pruning (Cont'd)

- The decision tree output c is selected as the calibration action that gives the minimum cost for this final node associated with sample set K , i.e.,

$$c = \min_{c_i} \text{Cost}(K, c_i)$$

- The sub-tree is replaced with a leaf node if the pruning reduces the cost, i.e.,

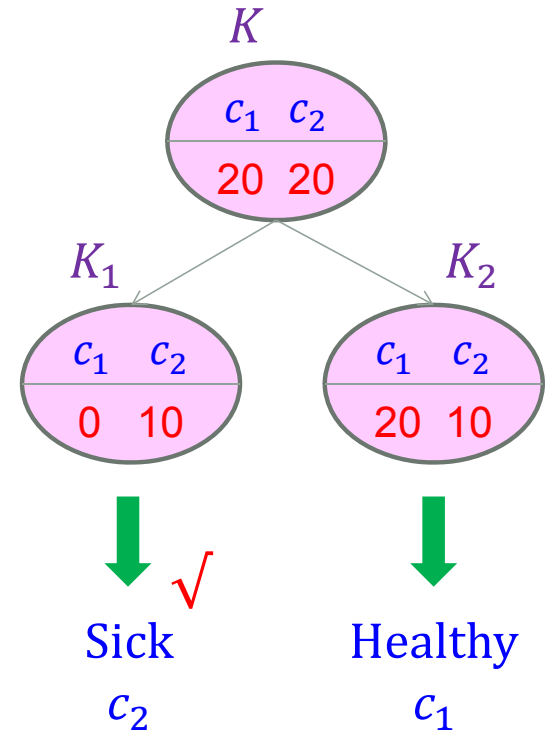
$$\min_{c_i} \text{Cost}\left(\sum_{j=1}^r K_j, c_i\right) < \sum_{j=1}^r \min_{c_i} \text{Cost}(K_j, c_i)$$

Example

- Suppose a test can distinguish health c_1 from sick c_2
- Assume the cost ratio between false healthy and false sick is 10
- For K_1

$$Cost(K_1, c_1) = 10 \cdot 10 \cdot \frac{10 + 1}{10 + 2} = 91.67$$

$$Cost(K_1, c_2) = 10 \cdot 1 \cdot \frac{1}{10 + 2} = 0.833$$



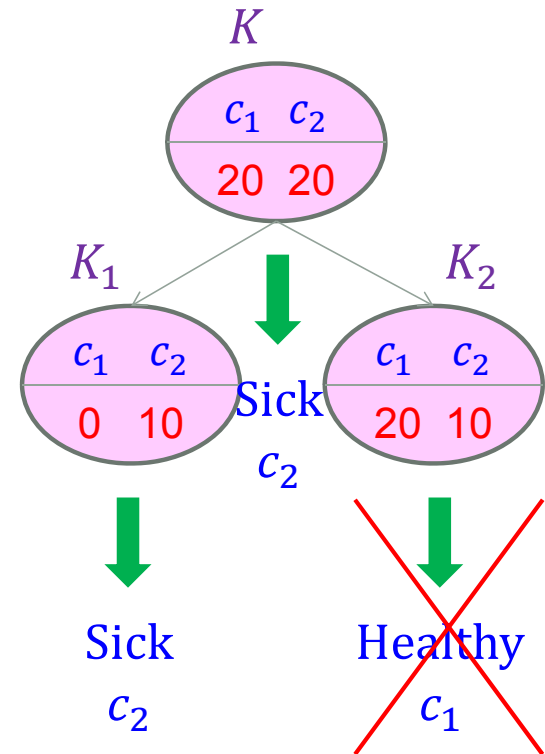
Example (Cont'd)

- For K_2

$$\text{Cost}(K_2, c_1) = 30 \cdot 10 \cdot \frac{10 + 1}{30 + 2} = 103.1$$

$$\text{Cost}(K_2, c_2) = 30 \cdot 1 \cdot \frac{20 + 1}{30 + 2} = 19.688$$

- Prune the sub-tree will reduce the overall cost, and the output for the pruned set K is sick c_2



Comparison of Pruning Methods

- Further reading: [J. Mingers. An Empirical Comparison of Pruning Methods for Decision Tree Induction](#)
- Other pruning methods mentioned in the paper:
 - Error-complexity pruning (Breiman et al.)
 - Critical value pruning (Mingers)
- Conclusion – methods (critical value, error-complexity, and reduced-error) that uses a test data set are more accurate but are slower

Unknown Attribute Values in DT Induction

- In real-world applications, it is often to encounter cases (training or test), some of whose attribute values are not known
- Unknown attribute values cause three problems:
 - Affects how impurity measures are computed
 - Affects how to distribute instance with missing value to child nodes
 - Affects how a test instance with missing value is classified

Dealing with Unknown Values for An Attribute

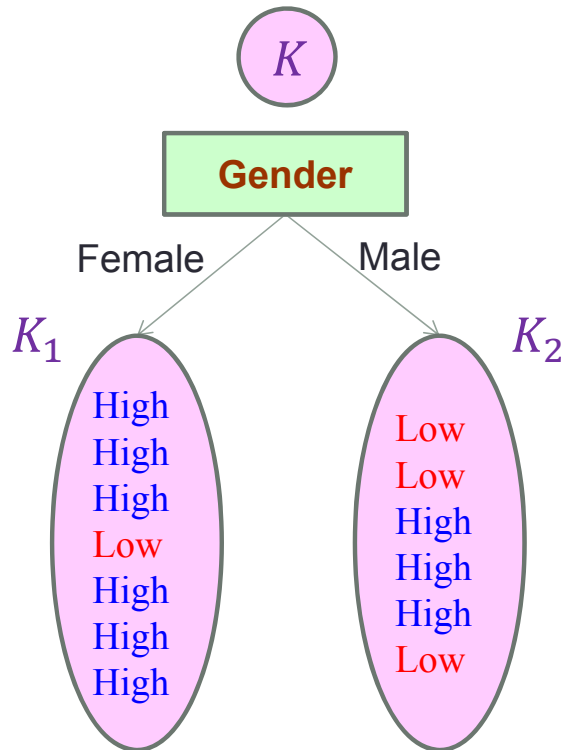
- Simple approaches:
 - Ignore cases in the training set with unknown attribute values
 - Use the most common value among examples at a node
- More complex, probabilistic approach (used in C4.5):
 - Assign a probability to each of the possible values for x based on the observed frequencies on x , and then propagate examples down the tree with these probabilities
- Other:
 - Include the training case in all subsets

Example

1. Remove data
2. Use "Female"

No.	Attributes				Class (Loyalty)
	<i>Location</i>	<i>Age</i>	<i>Marriage status</i>	<i>Gender</i>	
1	Urban	Below 21	Married	?	Low
2	Urban	Below 21	Married	Male	Low
3	Suburban	Below 21	Married	Female	High
4	Rural	Between 21 and 30	Married	Female	High
5	Rural	Above 30	Single	Female	High
6	Rural	Above 30	Single	Male	Low
7	Suburban	Above 30	Single	Male	High
8	Urban	Between 21 and 30	Married	Female	Low
9	Urban	Above 30	Single	Female	High
10	Rural	Between 21 and 30	Single	Female	High
11	Urban	Between 21 and 30	Single	Male	High
12	Suburban	Between 21 and 30	Married	Male	High
13	Suburban	Below 21	Single	Female	High
14	Rural	Between 21 and 30	Married	Male	Low

Example (Cont'd)



$$En(K) = - \left(\frac{9}{13} \log_2 \frac{9}{13} + \frac{4}{13} \log_2 \frac{4}{13} \right) = 0.8905$$

$$En(K_1) = - \left(\frac{6}{7} \log_2 \left(\frac{6}{7} \right) + \frac{1}{7} \log_2 \left(\frac{1}{7} \right) \right) = 0.5917$$

$$En(K_2) = - \left(\frac{3}{6} \log_2 \frac{3}{6} + \frac{3}{6} \log_2 \frac{3}{6} \right) = 1$$

Example (Cont'd)

- Adjust the information gain to reflect the case of unknown attribute values:

$$Gain(K, x) = \frac{\# \text{ of known cases}}{\# \text{ of total cases}} (En(K) - En(K, x))$$

- Information gain:

$$\begin{aligned} Gain(K, \textit{gender}) &= \frac{13}{14} (0.8905 - \left(\frac{7}{13} \cdot 0.5917 + \frac{6}{13} \cdot 1 \right)) \\ &= 0.103 \end{aligned}$$

Strength and Weakness of Decision Tree

- Strengths
 - Computationally inexpensive
 - Relatively faster learning speed
 - Convertible to easy to understand classification rules
 - Comparable classification accuracy with other methods
- Weaknesses
 - "Univariate" splits/partitioning using only one attribute at a time so limits types of possible trees
 - Pruning algorithms can be expensive since many potential sub-trees must be formed and compared

Incremental Learning

- Update the decision tree with each additional training example
- Good for adaptive system when environment undergoes changes
- May be with higher computational cost
- Some incremental decision tree methods:
 - Extension CART algorithm (Crawford, 1989)
 - ID3' (Schlimmer and Fisher, 1986)
 - ID6MDL (Kroon et. al., 2007)

Appropriate Problems for Decision Tree

- From Tom Mitchell's book:
 - Background concepts describe examples in terms of attribute-value pairs, values are always finite in number
 - Concept (target function) to be learned has discrete values
 - Disjunctive descriptions might be required in the answer

Summary

- A smaller tree is usually a better tree
- Information gain biased towards attributes with a large number of values
- Gain ratio takes number and size of branches into account when choosing an attribute
- Accuracy evaluation methods include k-fold cross-validation, bagging, and boosting
- No single method has been found to be superior over all others for all data sets

Decision Tree Software

- [C++ source code for C4.5](#)
- [CART supported by Matlab](#)
- [Weka\(supports C4.5\)](#)

Further Reading

- [J. Han and M. Kamber. *Data mining: concepts and techniques*](#)
- Ross Quinlan. *C4.5: Programs for Machine Learning*

Reference

- Ross Quinlan. *C4.5: Programs for Machine Learning*
- Tom Mitchell. *Machine Learning*