MACHINE LEARNING

Decision Tree Learning

Dr G.Kalyani

Department of Information Technology

Velagapudi Ramakrishna Siddhartha Engineering College

Topics

- **Introduction to Tree Models and Decision Tree Representation**
- **Appropriate Problems for Decision Tree Learning**
- **Basic Decision Tree Algorithm**
- **Inductive Bias in Decision Tree Learning**
- **Issues in Decision Tree Learning**

Tree Models

- A tree model is a hierarchical structure of conditions, where leafs contain tree outcome.
- They represent recursive divide-and-conquer, Greedy strategies.
- Tree models are among the most popular models in machine learning, because they are easy to understand and interpret.

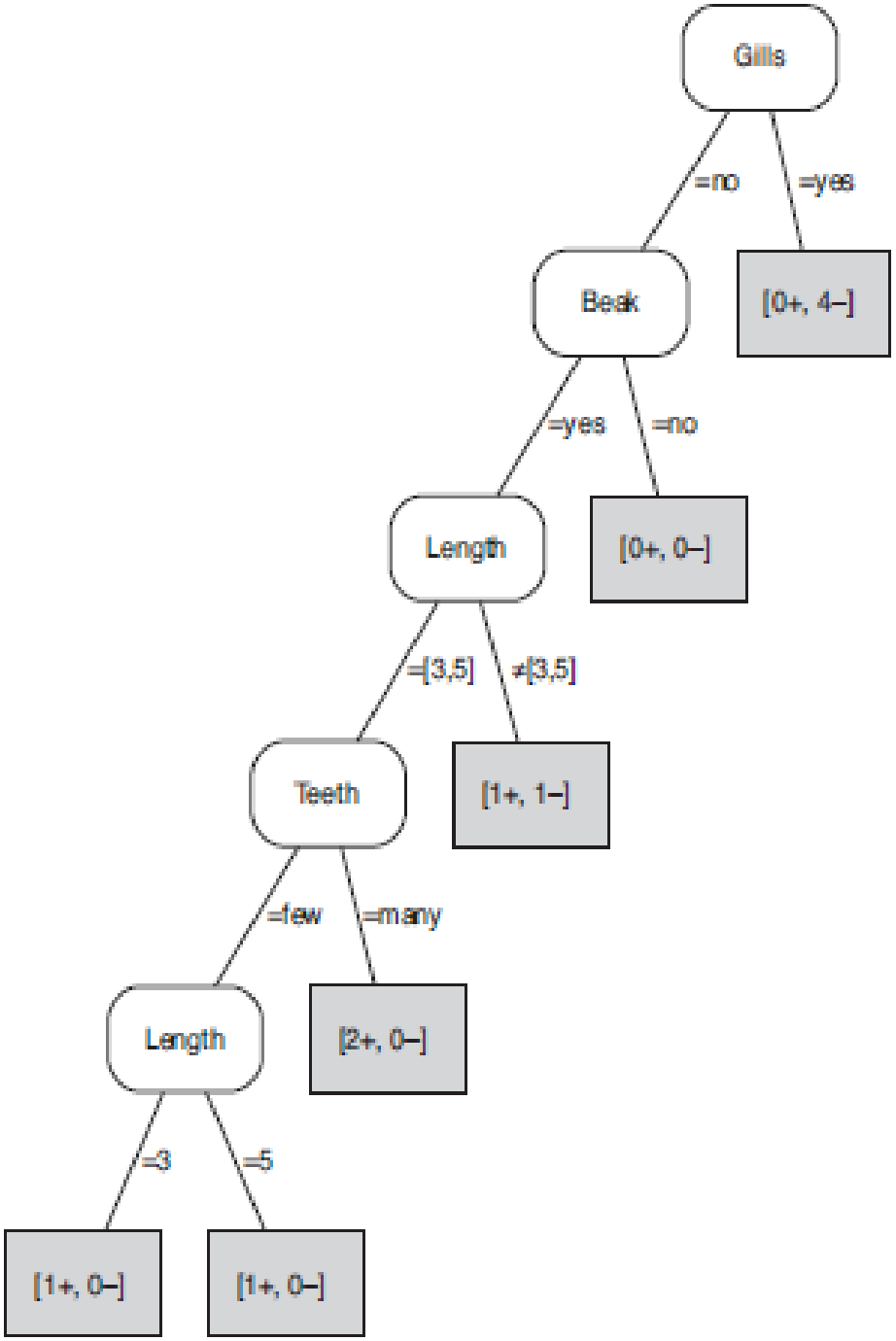
Dolphin Example

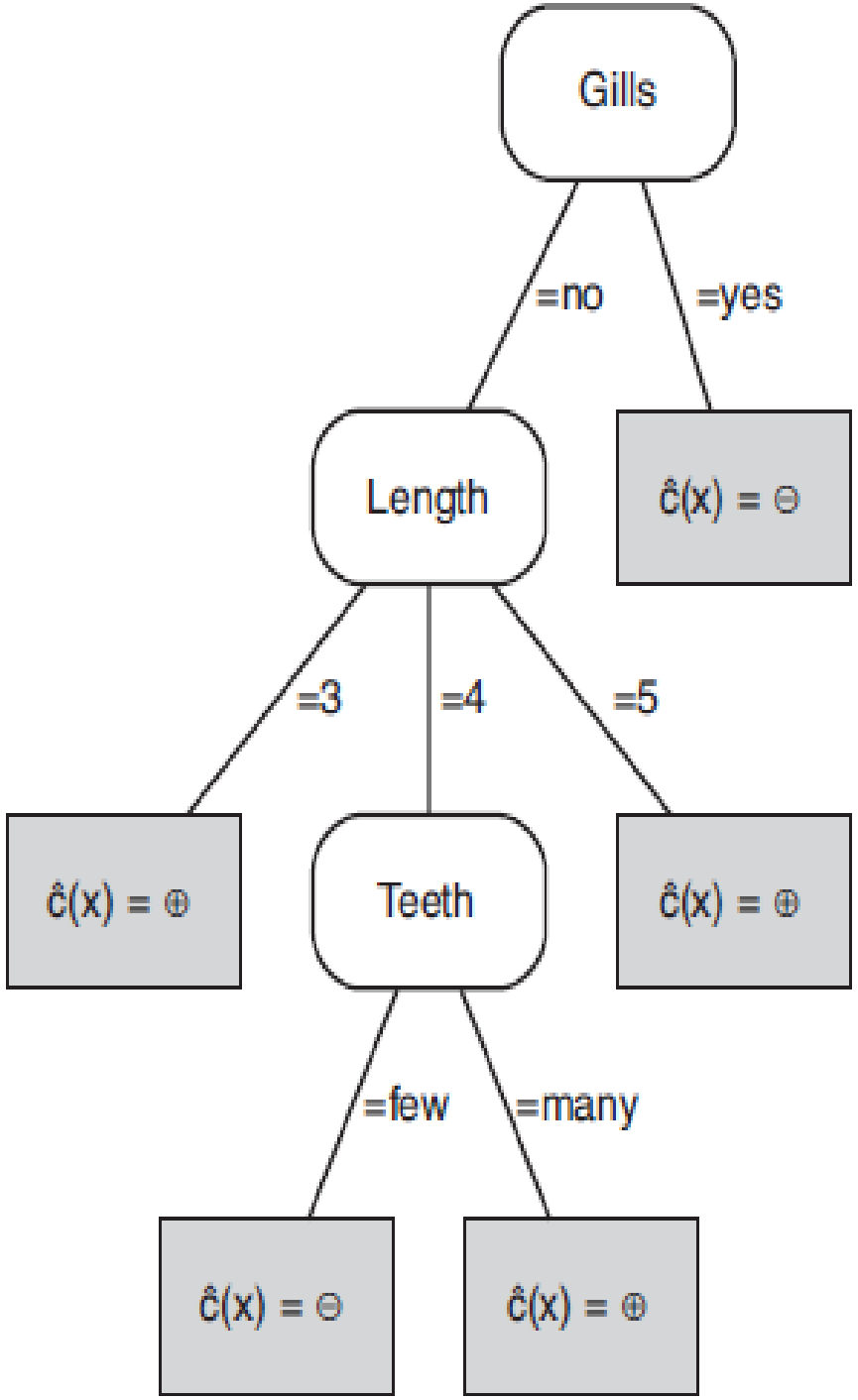
Suppose you come across a number of sea animals that you suspect belong to the same species. You observe their length in metres, whether they have gills, whether they have a prominent beak, and whether they have few or many teeth. Let the following be dolphins (positive class):

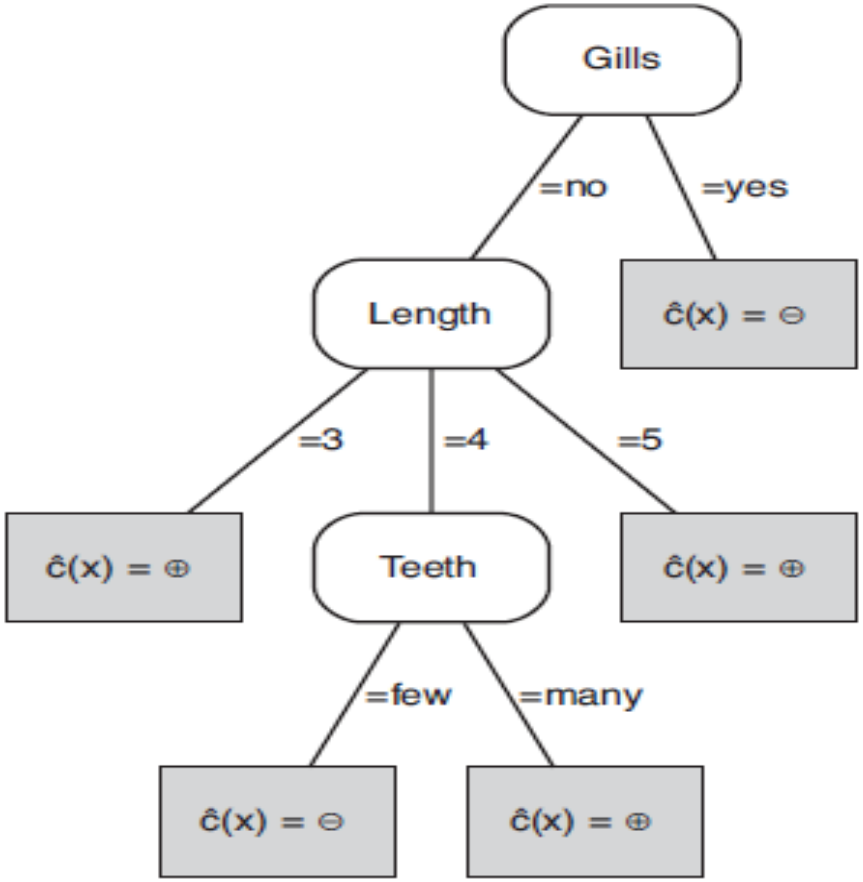
p1: $\text{Length} = 3 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$
p2: $\text{Length} = 4 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$
p3: $\text{Length} = 3 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{few}$
p4: $\text{Length} = 5 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$
p5: $\text{Length} = 5 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{few}$

and the following be not dolphins (negative class):

n1: $\text{Length} = 5 \wedge \text{Gills} = \text{yes} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$
n2: $\text{Length} = 4 \wedge \text{Gills} = \text{yes} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many}$
n3: $\text{Length} = 5 \wedge \text{Gills} = \text{yes} \wedge \text{Beak} = \text{no} \wedge \text{Teeth} = \text{many}$
n4: $\text{Length} = 4 \wedge \text{Gills} = \text{yes} \wedge \text{Beak} = \text{no} \wedge \text{Teeth} = \text{many}$
n5: $\text{Length} = 4 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{few}$


Gills
=no
=yes
Beak
[0+, 4−]
=yes
=no
Length
[0+, 0−]
=[3,5]
≠[3,5]
Teeth
[1+, 1−]
=few
=many
Length
[2+, 0−]
=3
=5
[1+, 0−]
[1+, 0−]


Gills
=no
=yes
Length
ĉ(x) = ⊖
=3
=4
=5
ĉ(x) = ⊕
Teeth
ĉ(x) = ⊕
=few
=many
ĉ(x) = ⊖
ĉ(x) = ⊕



$$(\text{Gills} = \text{no} \wedge \text{Length} = 3) \vee (\text{Gills} = \text{no} \wedge \text{Length} = 4 \wedge \text{Teeth} = \text{many}) \vee (\text{Gills} = \text{no} \wedge \text{Length} = 5)$$

$$\text{Gills} = \text{no} \wedge [\text{Length} = 3 \vee (\text{Length} = 4 \wedge \text{Teeth} = \text{many}) \vee \text{Length} = 5]$$

Feature Tree

- Tree models are not limited to classification but can be employed to solve almost any machine learning task, including ranking and probability estimation, regression and clustering.

- The tree structure that is common to all those models is called as Feature Tree.

Definition of Feature Tree

- **A Feature Tree is a tree in which each internal node is labelled with a feature, and each edge from an internal node is labelled with a literal**

- The set of literals at a node is called a split.

- Each leaf of the tree represents a logical expression, which is the conjunction of literals encountered on the path from the root of the tree to the leaf.

Topics

- **Introduction to Tree Models and Decision Tree Representation**
- **Appropriate Problems for Decision Tree Learning**
- **Basic Decision Tree Algorithm**
- **Inductive Bias in Decision Tree Learning**
- **Issues in Decision Tree Learning**

Problem Characteristics for Decision Trees

- **Instances are represented by attribute-value pairs.**
  - Fixed set of attributes, and the attributes take a small number of disjoint possible values.
- **The target function has discrete output values.**
  - Decision tree learning is appropriate for a boolean classification, but it easily extends to learning functions with more than two possible output values.
- **Disjunctive descriptions may be required.**
  - decision trees naturally represent disjunctive expressions.
- **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.
- **The training data may contain missing attribute values.**
  - Decision tree methods can be used even when some training examples have unknown values.
- **Decision tree learning has been applied to problems such as learning to classify**
  - medical patients by their disease,
  - equipment malfunctions by their cause, and
  - loan applicants by their likelihood of defaulting on payments.

Topics

- **Introduction to Tree Models and Decision Tree Representation**
- **Appropriate Problems for Decision Tree Learning**
- **Basic Decision Tree Algorithm**
- **Inductive Bias in Decision Tree Learning**
- **Issues in Decision Tree Learning**

Decision Trees

- The core algorithm employs a top-down, greedy search through the space of possible decision trees.

- This approach is exemplified by the ID3 algorithm and its successor C4.5.

Decision Trees

- Basic algorithm, ID3, 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 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.

Decision Trees

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*

Decision Trees

- A good quantitative measure for evaluating attributes is, a statistical property, called ***information gain,*** that measures how well a given attribute separates the training examples according to their target classification.
- In order to define information gain precisely, we begin by defining a measure commonly used in information theory, called ***entropy,*** that characterizes the impurity 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

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

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

Decision Trees

- Given entropy as a measure of the impurity in a collection of training examples, define a measure for the effectiveness of an attribute in classifying the training data.
- The measure we will use, called ***information gain,*** is simply the expected reduction in entropy caused by partitioning the examples according to this attribute.

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

where Values(A) is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has value v

Decision Trees

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

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

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

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

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

$$Gain(A) = Info(D) - Info_A(D)$$

ID3 Decision Tree: Example

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

- Class P: buys_computer = “yes”
  - 9 tuples
- Class N: buys_computer = “no”
  - 5 tuples

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

$$E(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940$$

ID3 Decision Tree: Example

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

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

age	# T	p_i	n_i
<=30	5	2	3
31…40	4	4	0
>40	5	3	2

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

ID3 Decision Tree: Example

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

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

$$Gain(A) = Info(D) - Info_A(D)$$

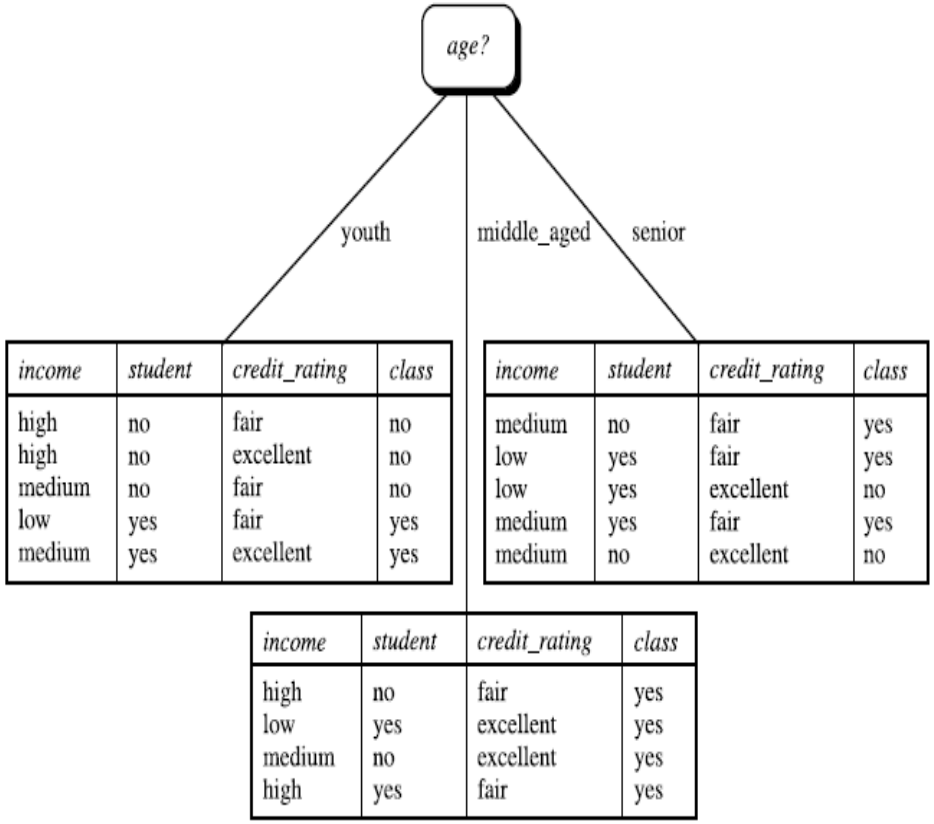
Gain (Age) = 0.940-0.694 = 0.246

Similarly, we can compute
Gain(income)= 0.029,
Gain(student)= 0.151,
and Gain(credit rating)= 0.048.

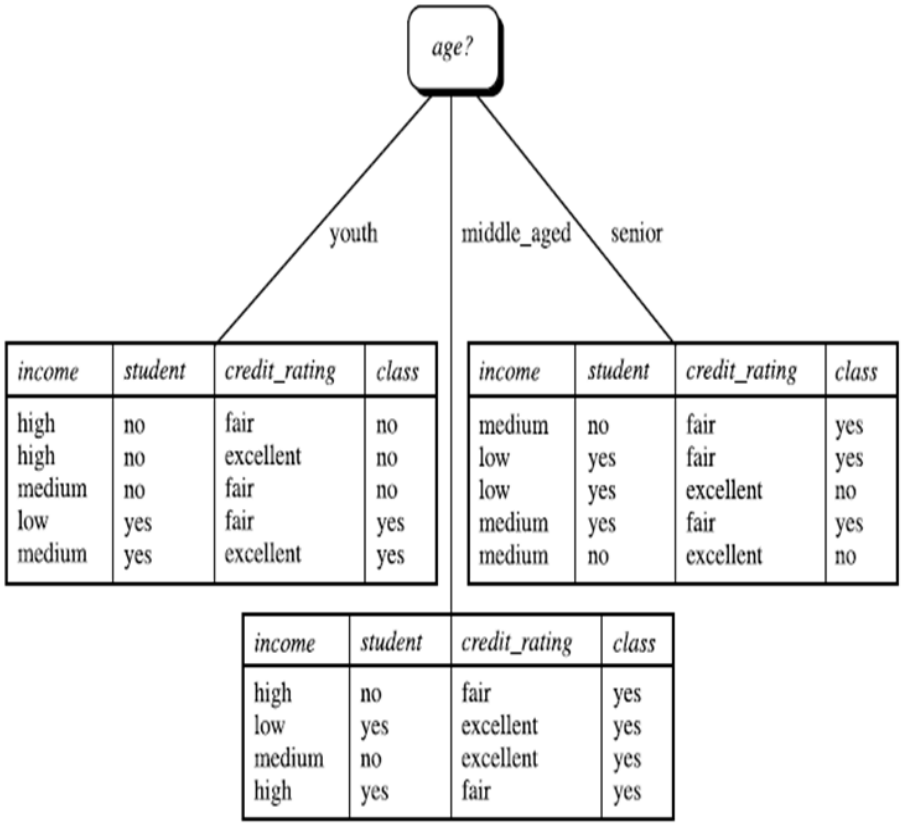
ID3 Decision Tree: Example

- Because *age* has the highest information gain
- among the attributes, it is selected as the splitting attribute.
- Node *N* is labeled with *age*, and branches are grown for each of the attribute's values.

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



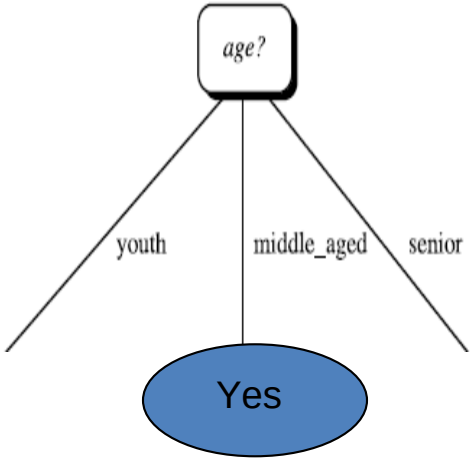
ID3 Decision Tree: Example



income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	*student*	*credit_rating*	*class*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	*student*	*credit_rating*	*class*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes



ID3 Decision Tree: Example

For age=youth:

income	student	credit_rating	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

- Class P: buys_computer = "yes"
  - 2 tuples
- Class N: buys_computer = "no"
  - 3 tuples

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

$$Info(D) = I(2,3) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}$$

= -0.4*-1.322-0.6* -0.737

= 0.971

$Info_{income}(D) = \frac{1}{5}Info(1) + \frac{2}{5}Info(1,1) + \frac{2}{5}Info(1,1)$

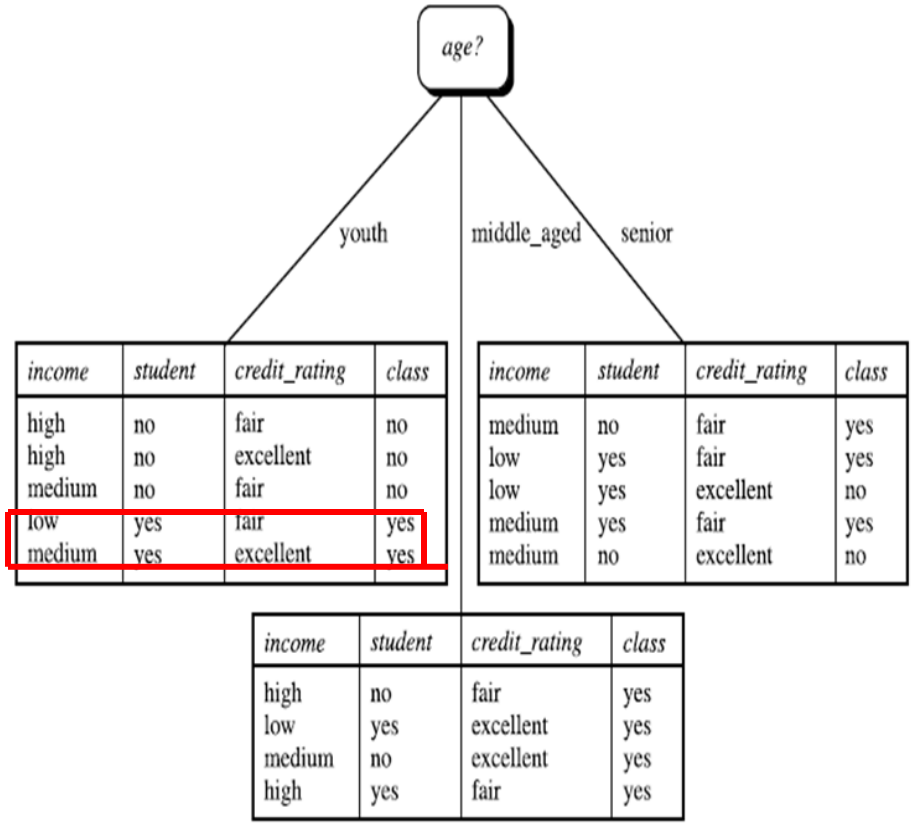
= 0.2*0 + 2*0.4*[$-\frac{1}{2} * log0.5 - \frac{1}{2} * log0.5$]

=0+0.8* 1 = 0.8

Gain $_{income}$(D) = 0.971-0.8 =0.171

Similarly Calculate Gain $_{student}$(D) and Gain $_{credit\text{-}rating}$ (D)

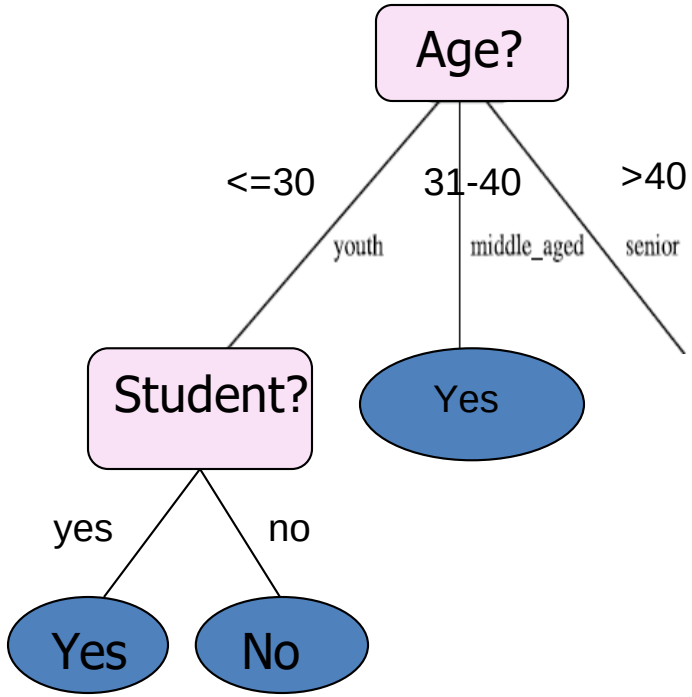
ID3 Decision Tree: Example



income	student	credit_rating	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

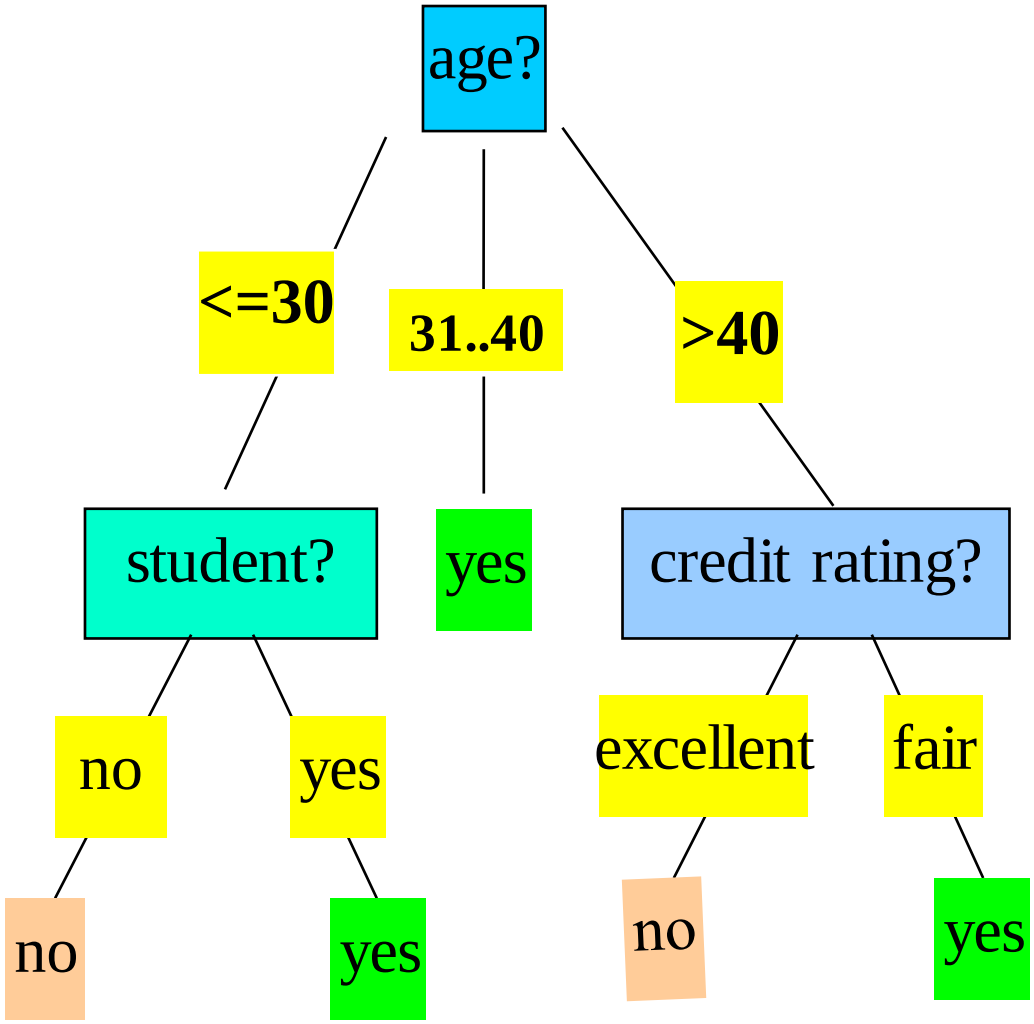
income	student	credit_rating	class
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	student	credit_rating	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes



ID3 Decision Tree: Example

- Training data set: Buys_computer
- Resulting tree:



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

ID3 Decision Tree: Example for Practice

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*Play Tennis*
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

Training examples for the target concept *PlayTennis*

Practice Example for Decision Tree

S.NO	LENGTH	GILLS	BEAK	TEETH	CLASS LABEL
1	5	NO	YES	MANY	D
2	4	YES	YES	MANY	ND
3	4	YES	YES	MANY	ND
4	5	NO	YES	FEW	D
5	4	NO	YES	MANY	D
6	5	YES	YES	MANY	ND
7	3	NO	YES	FEW	D
8	5	NO	NO	MANY	ND
9	4	YES	NO	FEW	ND
10	3	NO	YES	MANY	D

Example for practice

Patient ID	Age	Sex	BP	Cholesterol	Drug
p1	Young	F	High	Normal	Drug A
p2	Young	F	High	High	Drug A
p3	Middle-age	F	Hiigh	Normal	Drug B
p4	Senior	F	Normal	Normal	Drug B
p5	Senior	M	Low	Normal	Drug B
p6	Senior	M	Low	High	Drug A
p7	Middle-age	M	Low	High	Drug B
p8	Young	F	Normal	Normal	Drug A
p9	Young	M	Low	Normal	Drug B
p10	Senior	M	Normal	Normal	Drug B
p11	Young	M	Normal	High	Drug B
p12	Middle-age	F	Normal	High	Drug B
p13	Middle-age	M	High	Normal	Drug B
p14	Senior	F	Normal	High	Drug A
p15	Middle-age	F	Low	Normal	?

ID3- A Decision Tree Algorithm

ID3(*Examples, TargetAttribute, Attributes*)

- 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 *TargetAttribute* in *Examples*
- Otherwise Begin
  - A ← the attribute from *Attributes* that best classifies *Examples*
  - The decision attribute for *Root* ←A
  - For each possible value, vi, of A,
    - Add a new tree branch below *Root*, corresponding to the test A = vi
    - Let $Examples_{vi}$ be the subset of *Examples* that have value vi for A
    - If $Examples_{vi}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *TargetAttribute* in *Examples*
      - Else below this new branch add the subtree
        ID3($Examples_{vi}$, *TargetAttribute*, *Attributes* – {A})
- End
- Return *Root*

Topics

- **Introduction to Tree Models and Decision Tree Representation**
- **Appropriate Problems for Decision Tree Learning**
- **Basic Decision Tree Algorithm**
- **Inductive Bias in Decision Tree Learning**
- **Issues in Decision Tree Learning**

Definition of Inductive Bias

- The **inductive bias** (also known as **learning bias**) of a learning algorithm is the set of assumptions that the learner uses to predict outputs of given inputs that it has not encountered.

Inductive Bias of ID3 Decision Tree Alg

- **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.

Preference and Restriction Bias

- A ***preference bias*** is an inductive bias where some hypothesis are preferred over others.

- A ***restriction bias*** is an inductive bias where the set of hypothesis considered is restricted to a smaller set.

Preference and Restriction Bias

- Occam's razor
  - Prefer the simplest hypothesis that fits the data
  - Argument in favor
    - Fewer short hypotheses than long hypotheses
  - Argument opposed
    - There are many ways to define small sets of hypotheses
    - What's so special about small sets based on size of hypothesis?

Topics

- **Introduction to Tree Models and Decision Tree Representation**
- **Appropriate Problems for Decision Tree Learning**
- **Basic Decision Tree Algorithm**
- **Inductive Bias in Decision Tree Learning**
- **Issues in Decision Tree Learning**

Issues in Decision Tree Learning

- **Avoiding Overfitting the Data**
- **Incorporating Continuous-Valued Attributes**
- **Alternative Measures for Selecting Attributes**
- **Handling Training Examples with Missing Attribute Values**
- **Handling Attributes with Differing Costs**

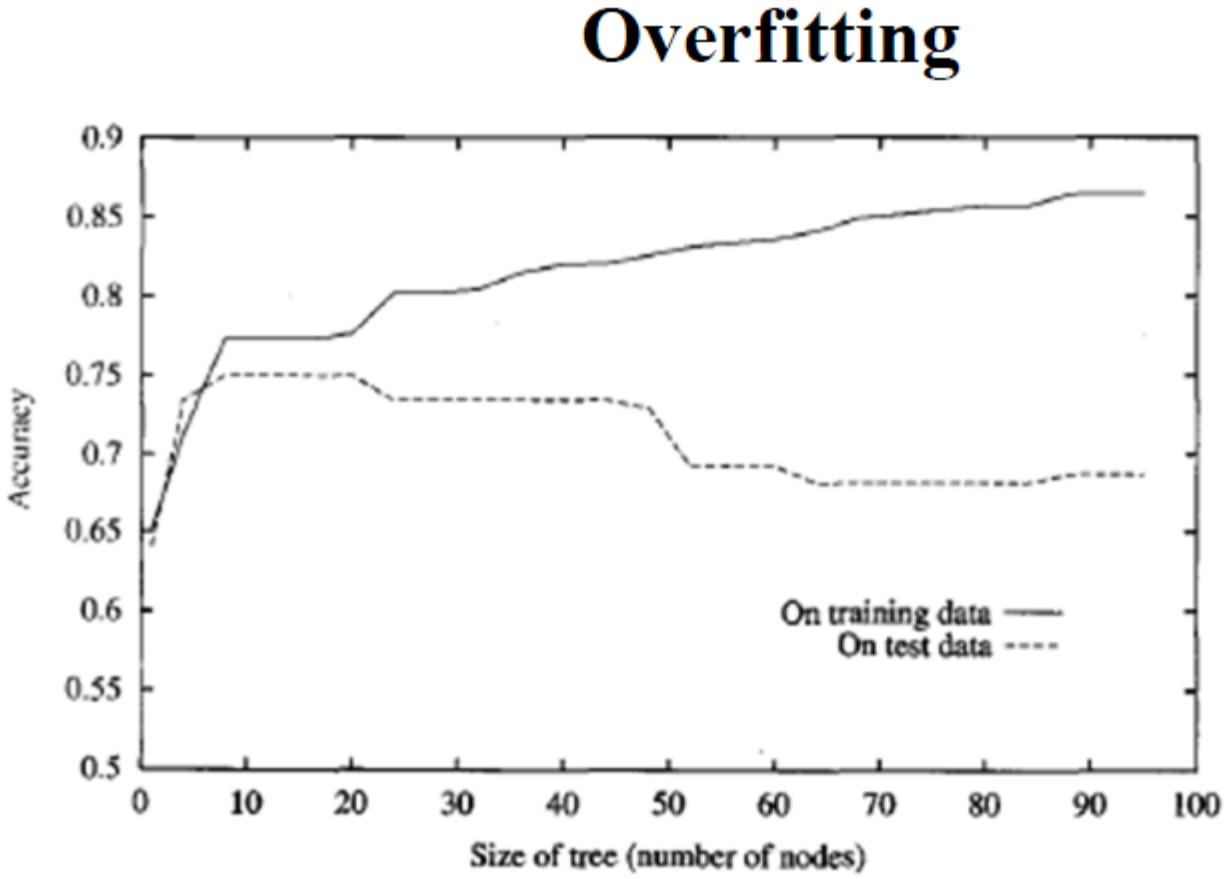
Issue-1: Overfitting the Data

- **Decision tree algorithms grows each branch of the tree just deeply enough to perfectly classify the training examples.**
- It can lead to difficulties when there is
  - noise in the data, or
  - number of training examples is too small
- In either of these cases, this simple algorithm can produce trees that ***overfit*** the training examples.

- **Definition of Overfitting**

> 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.

Overfitting in Decision Trees



- 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.

Avoiding the Overfitting

- **Approaches to avoid overfitting** in decision tree learning 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.**
- **Criterion used to determine the correct final tree size:**
  - **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.
  - 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.

Reduced Error Pruning

- 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.**

RULE POST-PRUNING

- 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.

- **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.

- **Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.**

- **Sort the pruned rules by their estimated accuracy**, and consider them in this sequence when classifying subsequent instances.

Issue-2: Incorporating Continuous-Valued Attributes

- **ID3 is restricted to attributes that take on a discrete set of values.**
- This 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 A, that is true if A < c and false otherwise.**
- The only question is **how to select the best value for the threshold c.**

Issue-2: Incorporating Continuous-Valued Attributes

- We would like to **pick a threshold, *c*, that produces the greatest information gain.**
- By **sorting the examples according to the continuous attribute A, then identifying adjacent examples that differ in their target classification.**
- We can **generate a set of candidate thresholds midway between the corresponding values of *A*.**
- These **candidate thresholds can then be evaluated by computing the information gain associated with each**.

- **Example:**

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

- In the current example, there are two candidate thresholds, corresponding to the values of Temperature at which the value of PlayTennis changes: (48 + 60)/2, and (80 + 90)/2.
- The information gain can then be computed for each of the candidate attributes, Temperature>54 and Temperature>85.
- The best can be selected (Temperature>54).

Issue-3: Alternative Measures for Selecting Attributes

- A natural bias in the **information gain measure is, it favors attributes with many values over those with few values.**
- One **alternative measure** that has been used successfully is the **gain ratio.**

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

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

Issue-4: Handling Training Examples with Missing Attribute Values

- 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, **assign it the most common value among examples at node *n* that have the classification *c(x).***

- A second, more complex procedure is to assign a probability to each of the possible values of *A* rather than simply assigning the most common value to *A(x).*

- These probabilities can be estimated again based on the observed frequencies of the various values for *A* among the examples at node *n.*

Issue-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.
- In such tasks, **prefer decision trees that use low-cost attributes** where possible, relying on high-cost attributes only when needed to produce reliable classifications.

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

Topics

- **Introduction to Tree Models and Decision Tree Representation**
- **Appropriate Problems for Decision Tree Learning**
- **Basic Decision Tree Algorithm**
- **Inductive Bias in Decision Tree Learning**
- **Issues in Decision Tree Learning**