Data Mining
(EECS 4412)

Decision Tree Learning

Parke Godfrey
EECS
Lassonde School of Engineering
York University

Thanks to

Professor Aijun An

for curation & use of these slides.

Outline

- Overview of classification
- Basic concepts in decision tree learning
  - Data representation in decision tree learning
  - What is a decision tree?
    - Decision tree representation
- How to learn a decision tree from data
  - Basic decision tree learning algorithm
  - How to select best attribute
  - Pruning decision tree
- Other issues involved in decision tree learning

Classification—A Two-Step Process

- Model construction (i.e., learning):
  - Learn a model from a *training set* (*a set of pre-classified training examples*) -- supervised learning
  - The model can be represented as *classification rules*, *decision trees*, *neural networks*, *mathematical formulae, etc.*
- Model usage (i.e., prediction or classification):
  - Classify future or unknown objects -- main purpose
  - Test the learned model on a *test set* (*another set of pre-classified examples*) to estimate accuracy of the model
    - The known class label of a test example is compared with the classification result from the model
    - Accuracy rate is the percentage of test examples that are correctly classified by the model
    - Test set is independent of training set, otherwise the testing result is not reliable

Classification Process (1): Model Construction

Training Data

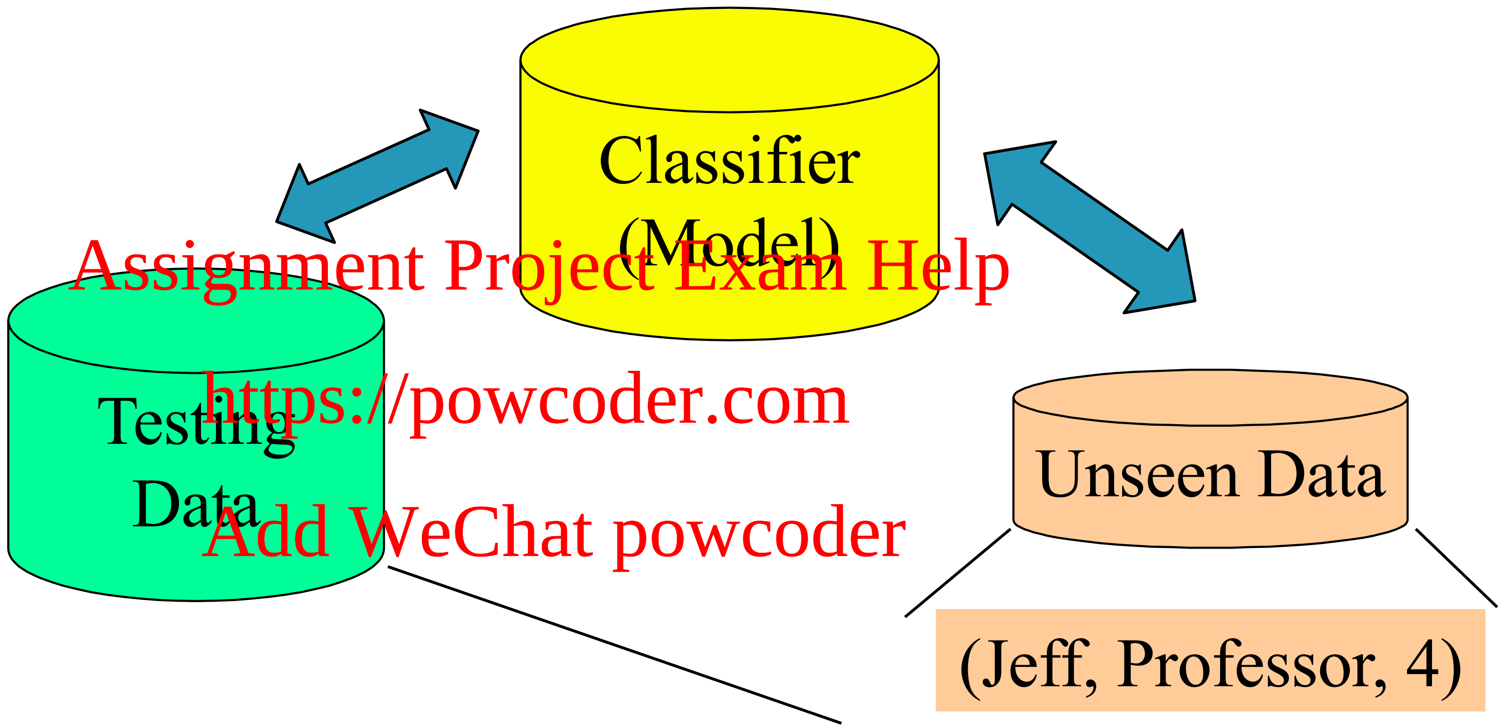
Classification Learning Algorithms

Classifier (Model)

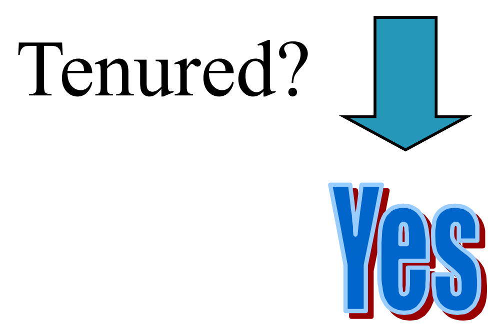
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = 'professor'
OR years > 6
THEN tenured = 'yes'

Classification Process (2): Use the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes



Classification Learning Techniques

- Decision tree learning
- Decision rule learning
- Bayesian classification
- Neural networks
- K-nearest neighbor method
- Support vector machines (SVM)
- Genetic algorithms
- etc.

Decision Tree Learning

- Objective of decision tree learning
  - Learn a decision tree from a set of training data
  - The decision tree can be used to classify new examples
- Decision tree learning algorithms
  - ID3 (Quinlan, 1986)
  - C4.5 (Quinlan, 1993)
  - CART (Breiman, Friedman, *et. al.* 1983)
  - CHAID (Kass, 1980)
  - QUEST(Loh and Shih, 1997)
  - etc.

Representation of Training Examples

Condition attributes

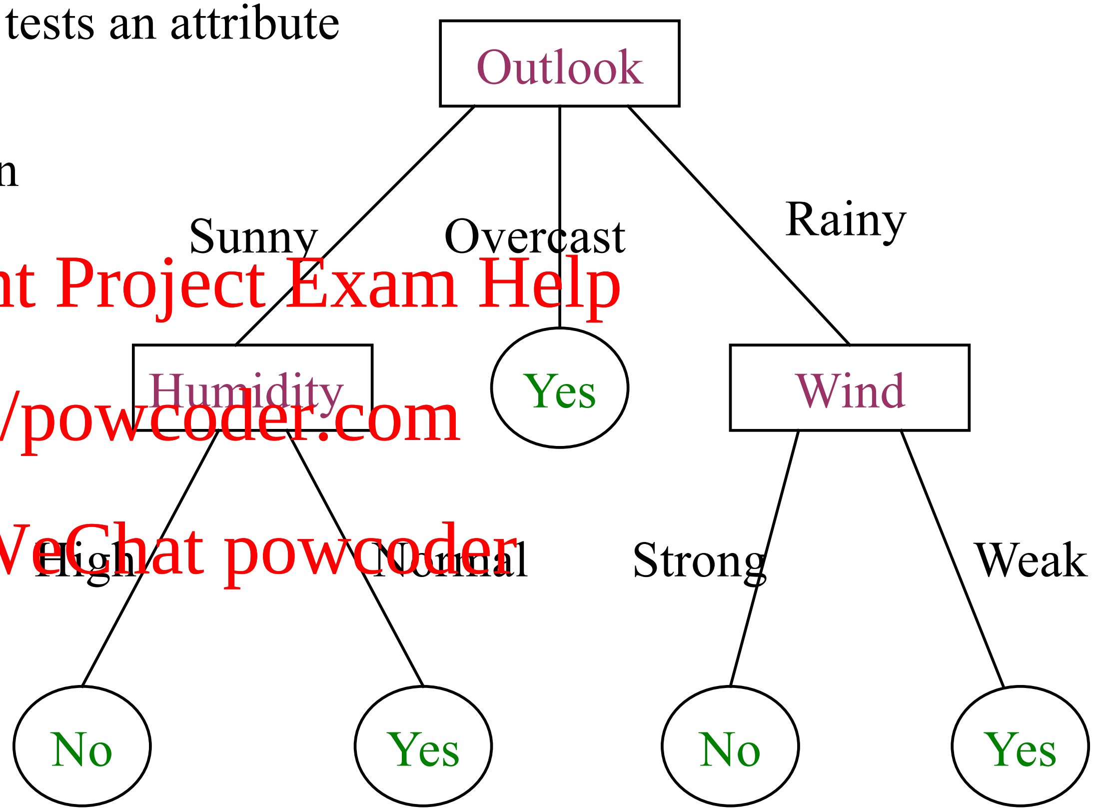
Class/Target/Decision attribute

Training examples or cases

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	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

Decision Tree Representation

- A decision tree: representation of classification knowledge
  - Each non-leaf (internal) node tests an attribute (Outlook, Humidity, Wind)
  - Each branch corresponds to an attribute value
  - Each leaf node assigns a classification



This tree classifies days according to whether or not they are suitable for playing tennis.

- Classification
  - A new case is classified by testing the case against the nodes from the root to a leaf node. The classification associated with the leaf is returned. For example,

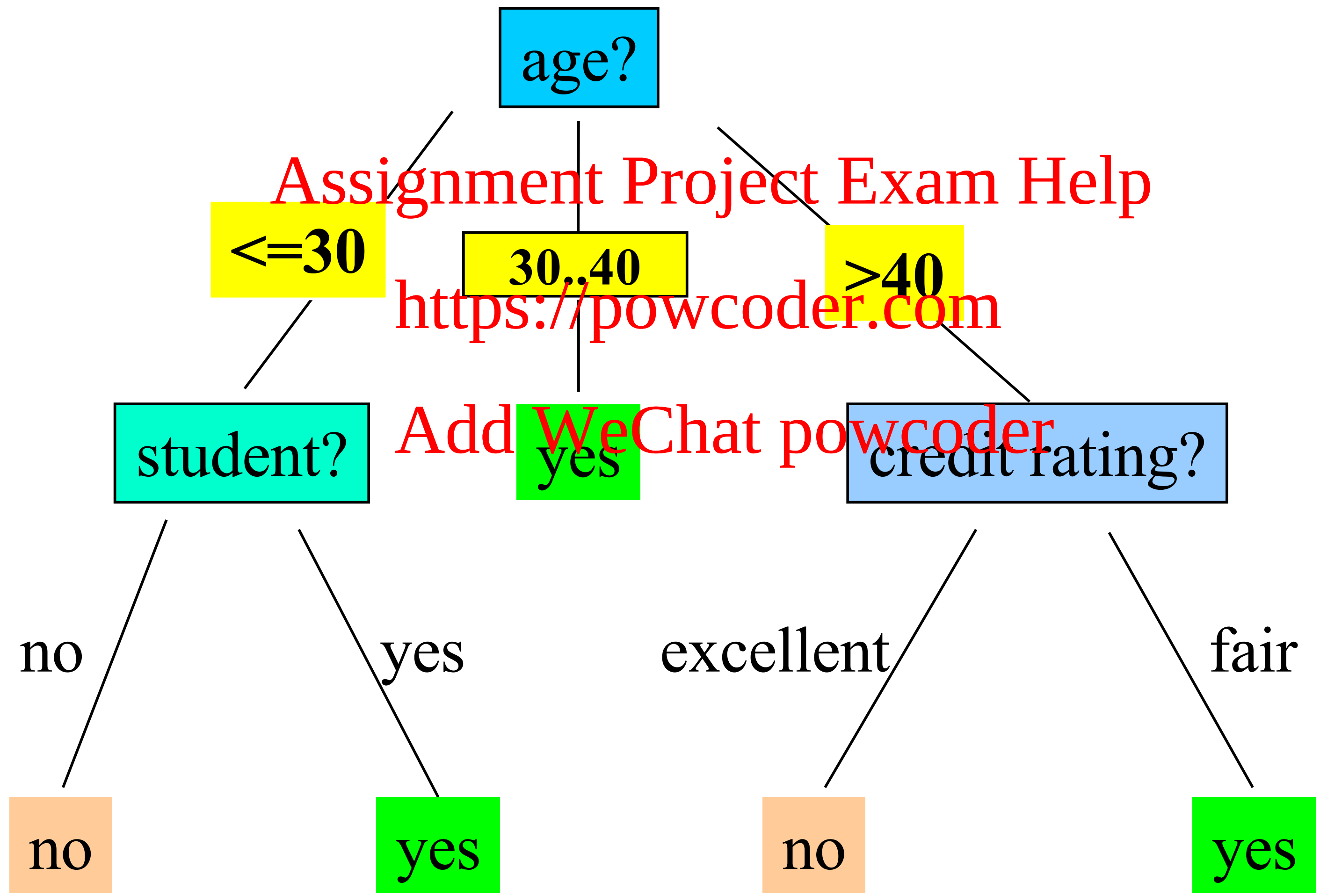
⟨Outlook = Sunny, Temperature = Mild, Humidity = high, Wind = Strong⟩ → No

Another Example of Training Dataset

- This follows an example from Quinlan's ID3

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
30…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

Output: A Decision Tree for "*buys_computer*"



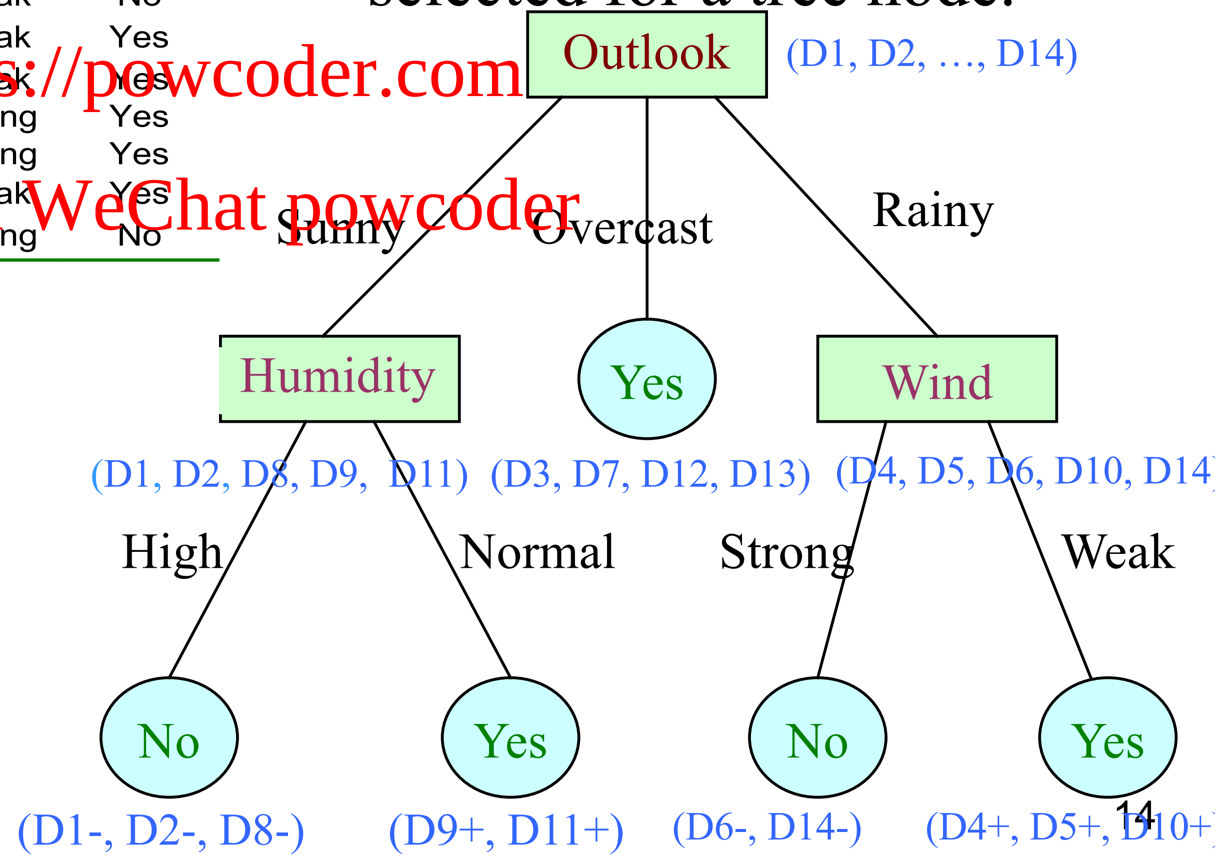
Outline

- Overview of classification
- Basic concepts in decision tree learning
  - Data representation in decision tree learning
  - What is a decision tree?
    - Decision tree representation
- <u>How to learn a decision tree from data</u>
  - Basic decision tree learning algorithm
  - How to select best attribute
  - Pruning decision tree
- Other issues involved in decision tree learning

Top-Down Induction of a Decision Tree

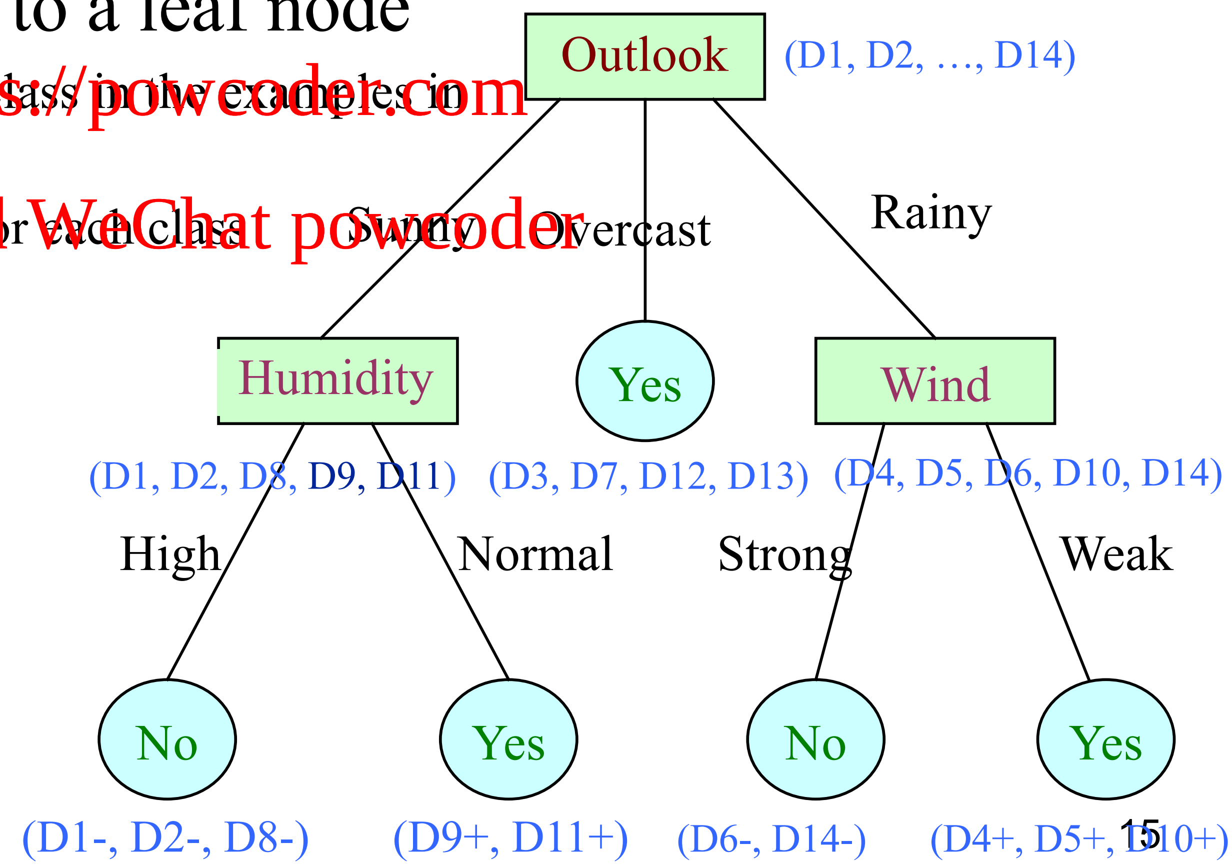
Day	Outlook	Tempe-rature	Humi-dity	Wind	PlayT-ennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	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

- Idea: divide and conquer
- Method: recursive partitioning of training data according to the attribute selected for a tree node.



Three Issues in Decision Tree Induction

- How to select an attribute for a node
- When to declare a node terminal
  - a naïve, but not robust method: when node is pure, stop growing.
- How to assign a class to a leaf node
  - Assign the most common class in the examples in the node to the node
  - Or output the probability for each class



How to Select Attribute

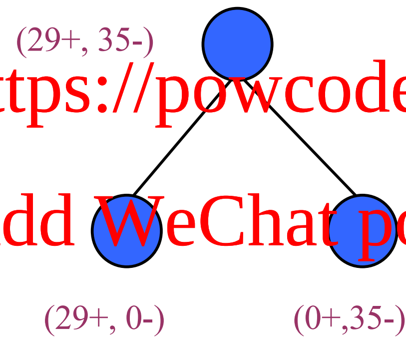
- Which attribute is the best attribute given a set of attributes and a set of examples?

- Many selection criteria, including:
  - Information gain (Quinlan, 1983; used in ID3)
  - Gain ratio (Quinlan, 1986; used in C4.5)
  - Gini index (Breiman, 1984; used in CART)
  - Chi-square statistic (Kass,1980; used in CHAID. Mingers, 1989)
  - Binarization (Bratko & Kononenko, 86)
  - Normalized information gain (Lopez de Mantaras, 91)

Information Gain

- Objective:
  - Select an attribute so that the data in each of the descendant subsets are the "purest".

An ideal split:



- Based on the concept of *entropy*
  - *Entropy* is a measure, commonly used in information theory, that characterizes the impurity (uncertainty, chaos) of an arbitrary collection of examples.

Entropy

- Given a set S of examples and k classes (C_1, ..., C_k), the *entropy* of S with respect to the k classes is defined as:

$$Entropy(S) = -\sum_{i=1}^{k} P(C_i)\log_2(P(C_i))$$

where $P(C_i)$ is the probability of examples in S that belong to C_i.

- The bigger *Entropy(S)* is, the more impure S is.
- Examples:
  - If all examples in S belong to the same class (i.e., S is pure),

    $Entropy(S)=0$.
  - If half of the examples in S belong to class 1 and the other half belong to class 2, $Entropy(S)=1$.
  - Suppose 9 examples are in class 1 and 5 examples in class 2,

    $Entropy(S)= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$
  - If the examples are uniformly distributed in 3 classes,

    $Entropy(S)= - ((1/3)\log_2(1/3))\times 3=\log_2 3 =1.59$

Information Gain (*Cont'd*)

- An attribute-selection criterion:
  - Used to choose an attribute to split a data set
- Assume that attribute A has m values.
  - Using A, data set S is split into S_1, S_2, …, S_m.
- Information Gain

Gain(S, A) = expected reduction in entropy due to partitioning S on attribute A

$$Gain(S, A) = Entropy(S) - \sum_{i=1}^{m} \frac{|S_i|}{|S|} Entropy(S_i)$$

where $|S|$ is the number of examples in set S, and $|S_i|$ is the number of examples in S_i.

An illustrative example

- Training examples

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	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

Which attribute is the best for the root?

S: [9+, 5-]
Entropy=0.940

Humidity

High → S_{High}: [3+, 4-], Entropy=0.985

Normal → S_{Normal}: [6+, 1-], Entropy=0.592

Gain(S, Humidity)
=.940- (7/14).985- (7/14).592=0.151

S: [9+, 5-]
Entropy=0.940

Outlook

Sunny → S_{Sunny}: [2+, 3-], Entropy=0.97095

Overcast → $S_{Overcast}$: [4+, 0-], Entropy=0

Rainy → S_{Rain}: [3+, 2-], Entropy=0.97095

Gain(S, Outlook)
=.940- (5/14).97095- (4/14)0- (5/14).97095=0.2467

S: [9+, 5-]
E=0.940

Wind

Weak → S_{Weak}: [6+, 2-], Entropy=0.811

Strong → S_{Strong}: [3+, 3-], Entropy=1

Gain(S, Wind)
=.940- (8/14).811- (6/14)1.0=0.048

S: [9+, 5-]
E=0.940

Temperature

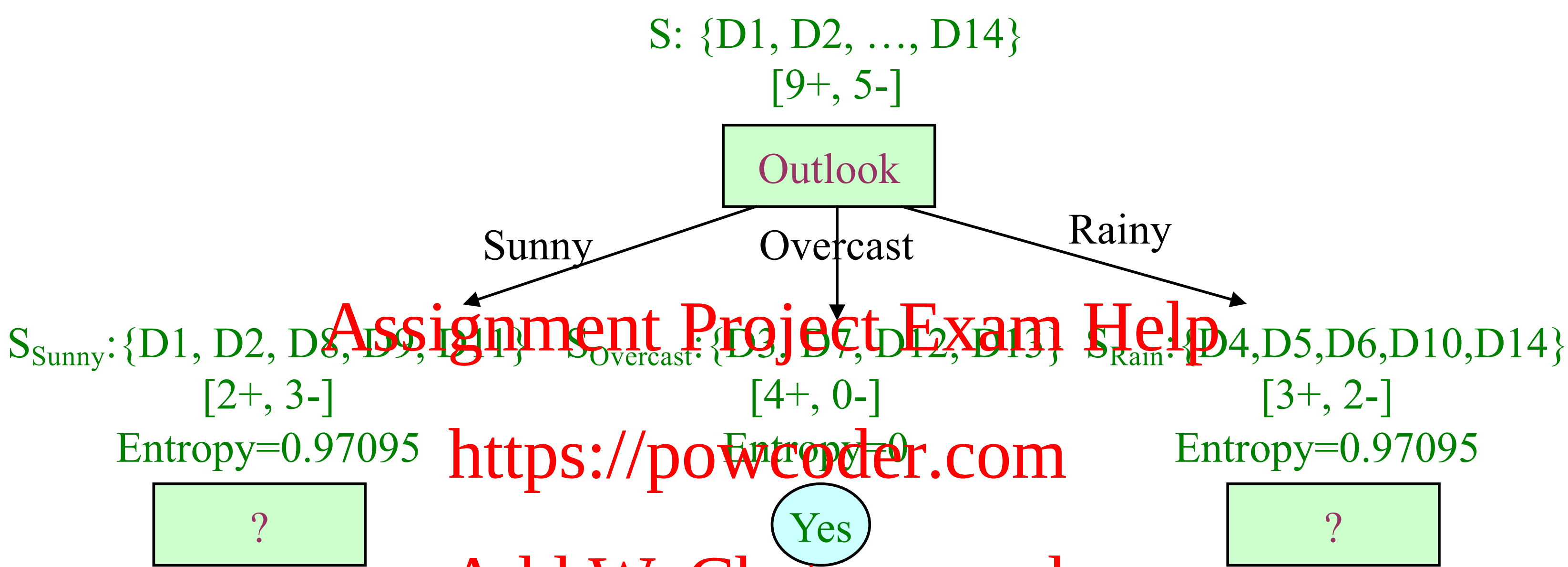
Hot → S_{Hot}: [2+, 2-], Entropy=1

Mild → S_{Mildt}: [4+, 2-], Entropy=0.91826

Cool → S_{Cool}: [3+, 1-], Entropy=0.811

Gain(S, Temperature)
=.940- (4/14)1- (6/14).91826- (4/14).811=0.029

An illustrative example (Cont'd.)

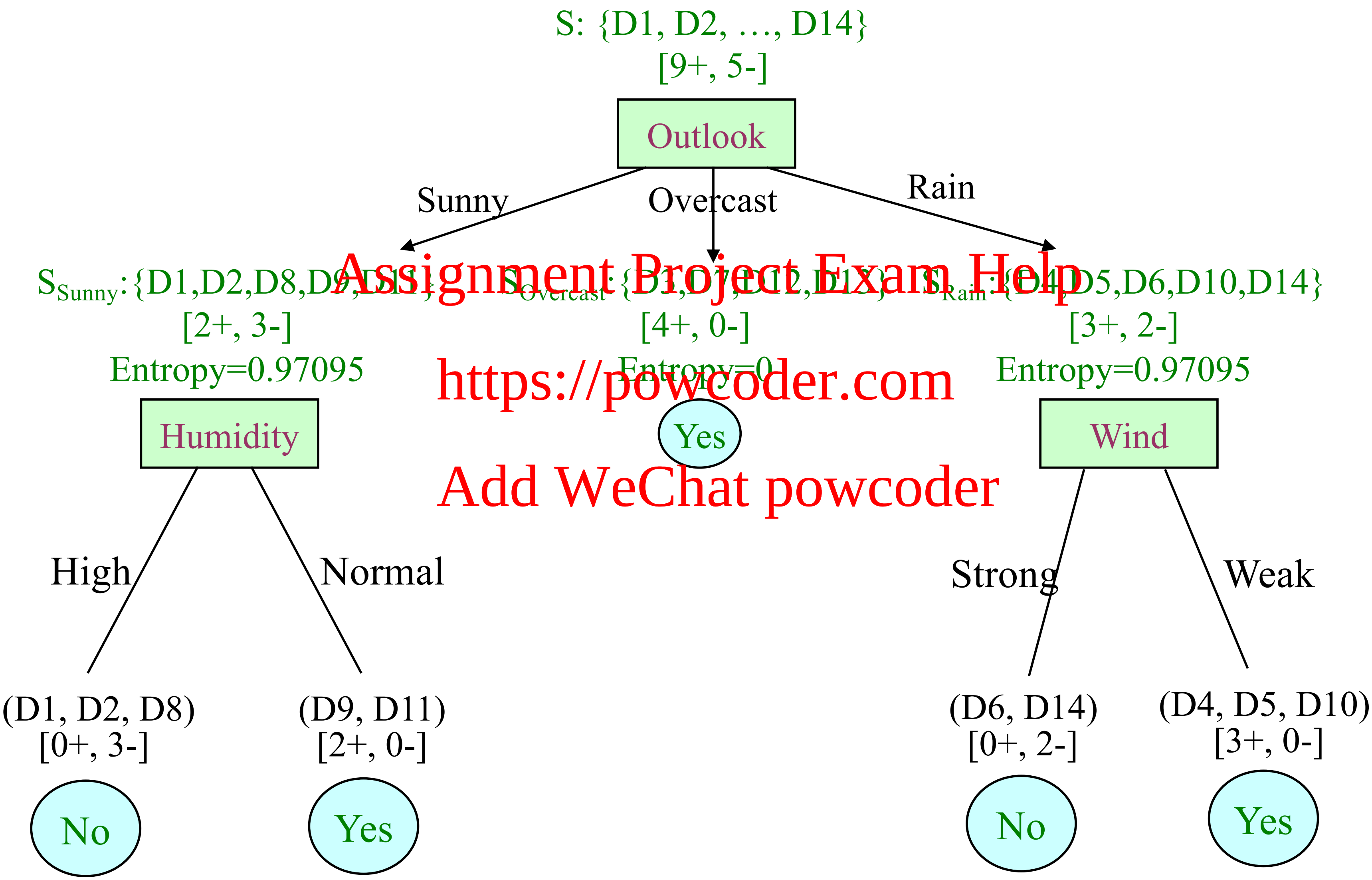


Which attribute should be tested here, Humidity, Temperature, or Wind?

Gain(S_{sunny}, Humidity) = .97095- (3/5)0.0- (2/5)0.0 = 0.97095
Gain(S_{sunny}, Temperature) = .97095- (2/5)0.0- (2/5)1.0- (1/5)0.0 = 0.57095
Gain(S_{sunny}, Wind) = .97095- (2/5)1.0- (3/5).918 = 0.02015

Therefore, Humidity is chosen as the next test attribute for the left branch.

An illustrative example (Cont'd.)



Basic Decision Tree Learning Algorithm

1. Select the "best" attribute *A* for the root node

2. Create new descendents of the node according to the values of *A*:

3. Sort training examples to the descendent nodes.

4. For each descendent node,

when to terminate the recursive process

- if the training examples associated with the node belong to the same class, the node is marked as a leaf node and labeled with the class
- else if there are no remaining attributes on which the examples can be further partitioned, the node is marked as a leaf node and labeled with the most common class among the training cases for classification;
- else if there is no example for the node, the node is marked as a leaf node and labeled with the majority class in its parent node.
- otherwise, recursively apply the process on the new node.

Outline

- Overview of classification
- Basic concept in decision tree learning
  - Data representation in decision tree learning
  - What is a decision tree?
- How to learn a decision tree from data
  - Basic decision tree learning algorithm
  - How to select best attribute
    - Information gain
    - Gain ratio
    - Gini index
  - Pruning decision tree
    - Pre-pruning
    - Post-pruning
- Other issues involved in decision tree learning

Bias in the Information Gain Measure

- Favor unfairly attributes with large numbers of distinct values at the expense of those with few.
  - E.g., attribute Date: poor predictor, but has the highest gain because it alone perfectly predicts the target attribute over the training data

Day	Outlook	Tempe-rature	Humi-dity	Wind	PlayT-ennis
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

Gain Ratio

- Proposed by Quinlan in 1986 (used in C4.5)
- Idea:
  - penalizes attributes with many distinct values by dividing information gain by attribute information (entropy of data with respect to the values of attribute):

$$SplitInformation(S, A) = -\sum_{v_i \in Values(A)} \frac{|S_{v_i}|}{|S|} \log_2 \frac{|S_{v_i}|}{|S|} = -\sum_{v_i \in Values(A)} P(v_i) \log_2 P(v_i)$$

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

Gini Index

- Gini diversity index (used by CART)
  - Another measure that measures the impurity of a data set.
  - S is a set of training examples associated with a node
  - Suppose there are n classes: C_i ($i = 1, \ldots, n$)
  - $P(C_i)$ is the probability of examples in S that belong to C_i.
  - The Gini impurity of S with respect to classes can be measured as:

$$i(S) = \sum_{j \neq i} P(C_j) P(C_i) = 1 - \sum_{j=1}^{n} (P(C_j))^2$$

- Similar to entropy
  - Minimized if classes for all examples are the same
  - Maximized if equal proportion of classes
- Selection of attribute using Gini index selects an attribute A that most reduces the impurity due to partitioning on A:

$$\Delta i(S, A) = i(S) - \sum_{v_i \in Values(A)} \frac{|S_{v_i}|}{|S|} i(S_{v_i})$$

Outline

- Overview of classification
- Basic concepts in decision tree learning
  - Data representation in decision tree learning
  - What is a decision tree?
    - Decision tree representation
- How to learn a decision tree from data
  - Basic decision tree learning algorithm
  - How to select best attribute
  - *Pruning decision tree*
- Other issues involved in decision tree learning

Basic Decision Tree Learning Algorithm (Review)

1. Select the "best" attribute *A* for the root node

2. Create new descendents of the node according to the values of *A*:

3. Sort training examples to the descendent nodes.

4. For each descendent node,

when to terminate the recursive process

- if the training examples associated with the node belong to the same class, the node is marked as a leaf node and labeled with the class
- else if there are no remaining attributes on which the examples can be further partitioned, the node is marked as a leaf node and labeled with the most common class among the training cases for classification;
- else if there is no example for the node, the node is marked as a leaf node and labeled with the majority class in its parent node.
- otherwise, recursively apply the process on the new node.

Overfitting Problem

- When to declare a node terminal in the basic algorithm:
  - grow each branch of the tree just deeply enough to classify the training examples as perfectly as possible.
- This strategy leads to producing deep branches that cover very few examples.
- This kind of trees *overfits* the training data in the following two situations:
  - there is noise in the data → tree fits the noise.
  - the number of training examples is too small to produce a representative sample of the true target function → tree is too specific to classify future examples well.

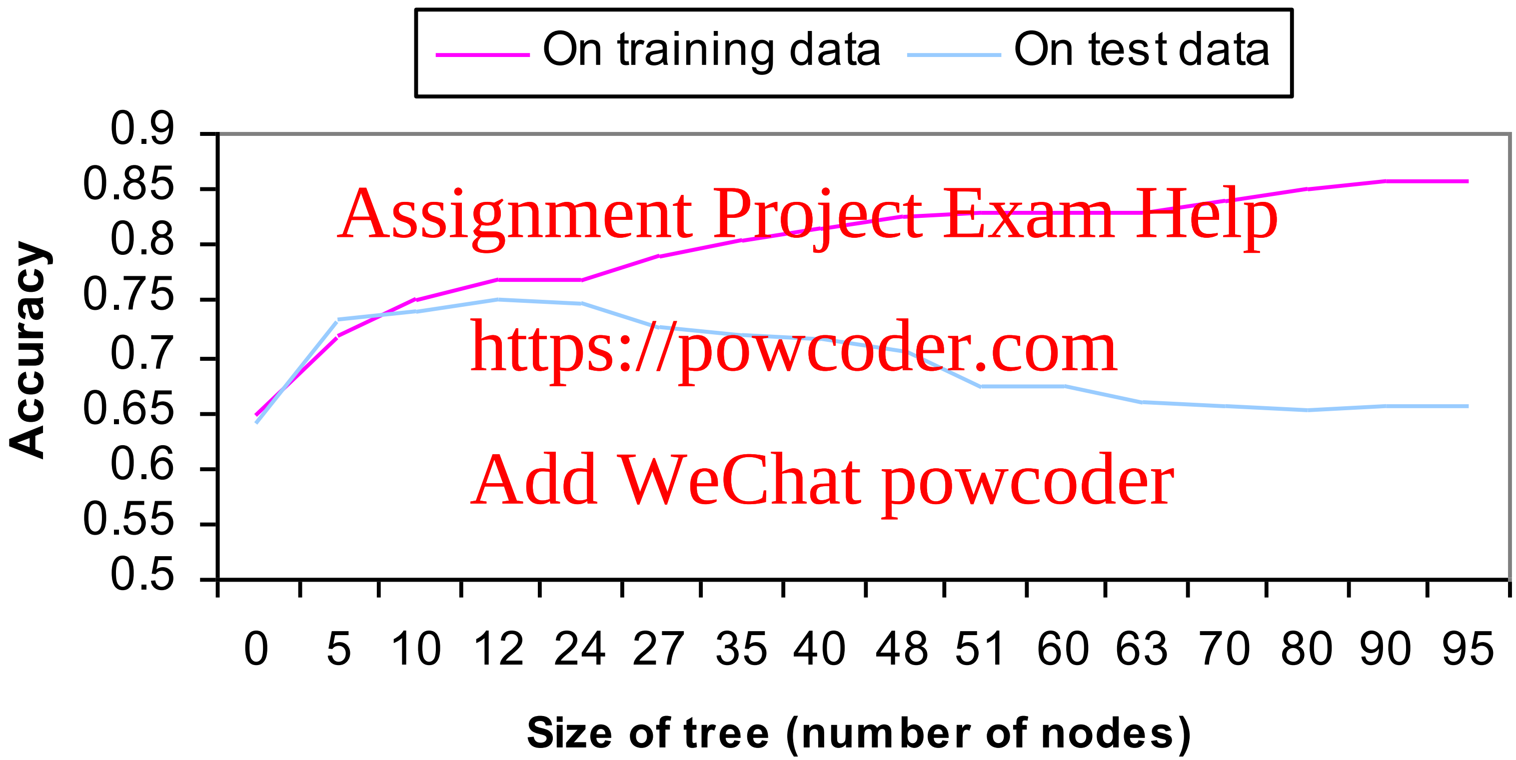
Overfitting Problem (*Cont'd*)

- A definition of *overfitting*
  - Consider the error of a model (e.g., a tree) h over
    - training data: $error_{train}(h)$
    - entire distribution D of data: $error_D(h)$
  - Model h overfits training data if there is an alternative model h' such that

    $error_{train}(h) < error_{train}(h')$ and

    $error_D(h) > error_D(h')$

Overfitting in Decision Tree Learning (An Example Diagram)



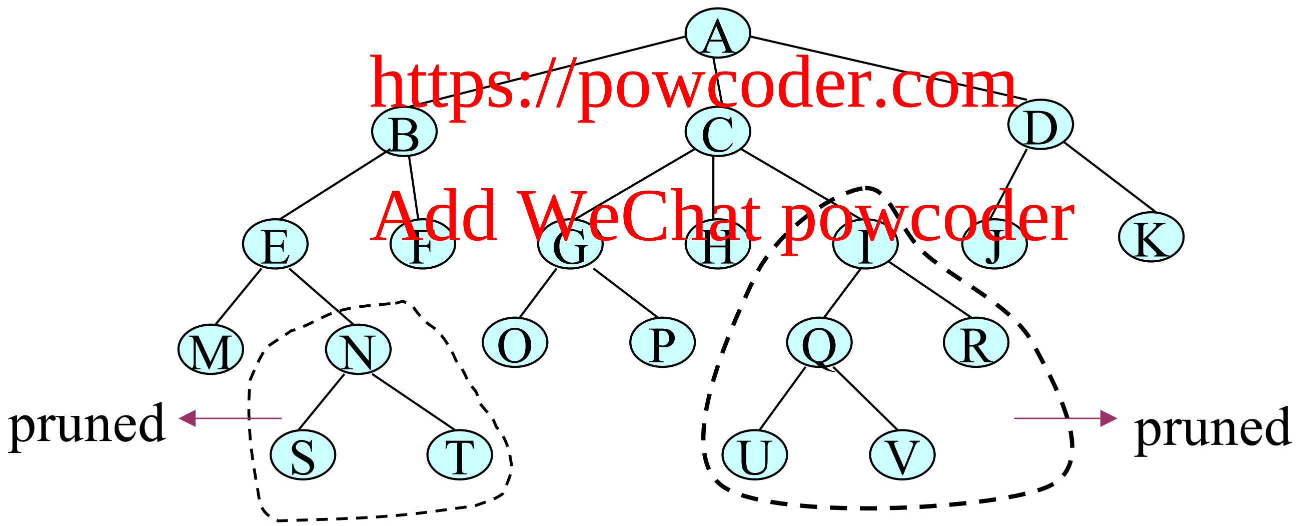
Accuracy = 1 – error rate

Preventing Overfitting

- **Pre-pruning**: stop growing the tree when data split is not statistically significant
  - For example,
    - set a threshold $\alpha > 0$ and declare a node terminal if percentage of examples in the most common class $> \alpha$, or
    - set a threshold $\beta > 0$ and declare a node terminal if highest information gain $< \beta$
  - Problems:
    - Hard to set the threshold value
    - The splitting is either stopped too soon or continued too far depending on the threshold
  - Using more complicated stopping rule does not help

Preventing Overfitting (Cont'd)

- **Post-pruning**: grow full tree, allow it to overfit the data, and then remove some subtrees



  - More successful in practice
  - Criterion is needed to determine what to prune

Post-pruning Functions

- Post-pruning functions are needed to determine which part(s) of the tree should be pruned
- Several post-pruning functions, including:

  - Reduced-error (Quinlan, 87)

  - Error-complexity (CART, 84)

  - Pessimistic error (Quinlan, 86)
  - Minimum description length (MDL) (SLIQ by Mehta *et al.* 1996)
  - Minimum error (Niblett & Bratko, 86)
  - Critical value (Mingers, 87)
- There is no single best pruning algorithm

Reduced-Error Pruning

- Procedure
  - Split training data into *growing* and *pruning* sets
  - Generate an overfitted decision tree using the *growing* set
  - Post-pruning: Do until further pruning is harmful:
    - Consider each of the internal non-root nodes in the tree to be candidates for pruning
      - Prune a node by removing subtree rooted at this node, making it a leaf node, and assigning it the most common classification of the training examples affiliated with this node
      - Evaluate impact of pruning this node on the *pruning* set by
        - calculating the classification error rate of the pruned tree on the *pruning* set and comparing it with the error rate of the unpruned tree.
    - Greedily remove the one whose removal most reduces the error on *pruning* set.
- Aim to produce smallest version of most accurate subtree
- What if data is limited?

But may be sub-optimal.

Error-Complexity Pruning

- Similar procedure to the reduced-error pruning
  - Split training data into *growing* and *pruning* sets
  - Generate the decision tree with the *growing* set
  - Post-pruning: Do until further pruning is harmful:

    - Evaluate impact of pruning each subtree on the *pruning* set
    - Greedily remove the one whose removal minimizes the following expression on the *pruning* set:

$$R_\alpha(T) = E(T) + \alpha\, L(T)$$

where $E(T)$ is the classification error of tree T, $L(T)$ is the number of leaf nodes in T and α is the complexity cost per leaf node.

    - $R_\alpha(T)$ is a linear combination of the classification error rate of the tree and its complexity.

- Similar to reduced-error pruning, not suitable if the size of training data is small.

Pessimistic-Error Pruning

- This method does not require a separate pruning set.
- Suppose a subtree T_s contains L leaves and J of training examples associated with T_s are misclassified
- If we replace T_s with a leaf which misclassifies E of the associated training examples, the pruned tree will be accepted if

$$E+1/2 < J+L/2$$

- Advantage
  - fast: no need to separate growing and pruning sets.
  - Tree building makes use of all the training data.

Outline

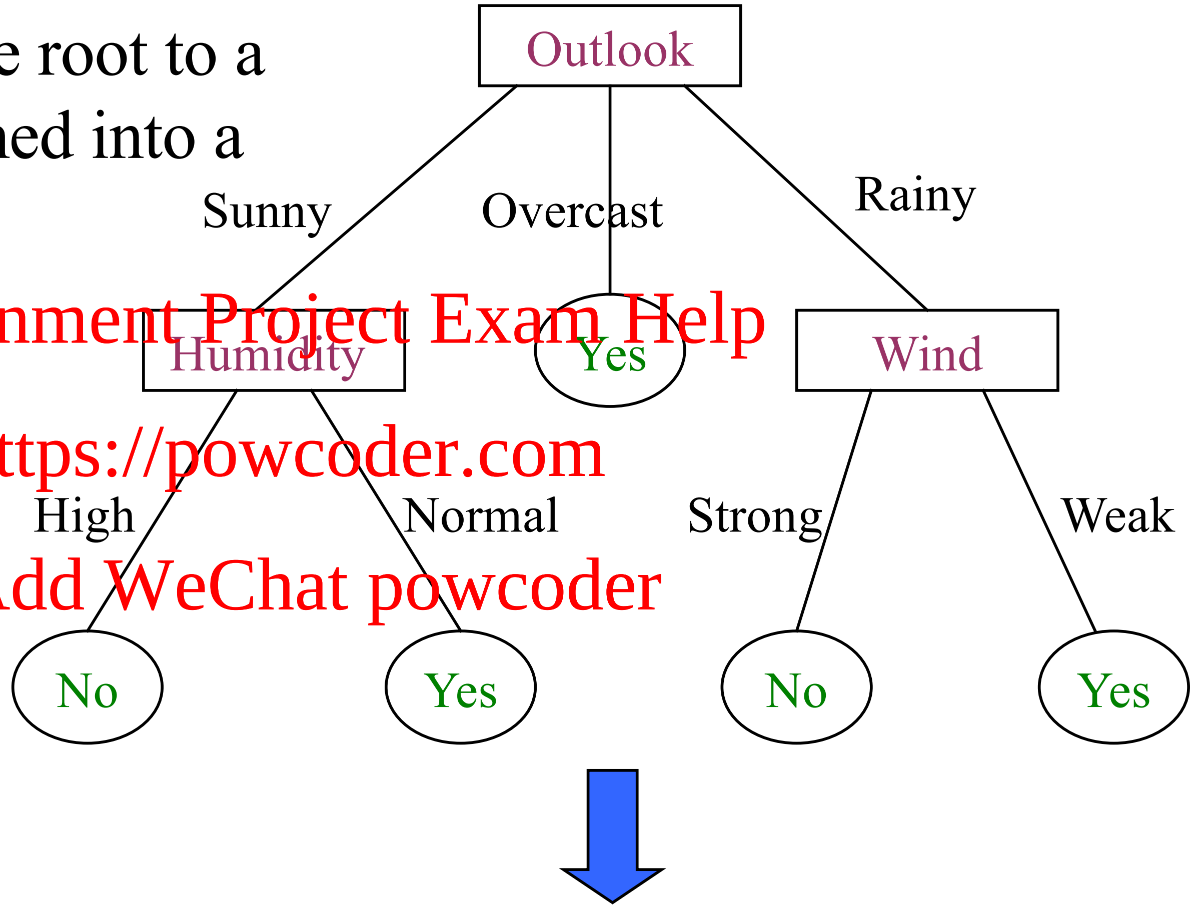
- Overview of classification
- Basic concepts in decision tree learning
  - Data representation in decision tree learning
  - What is a decision tree?
    - Decision tree representation
- How to learn a decision tree from data
  - Basic decision tree learning algorithm
  - How to select best attribute
  - Pruning decision tree
- *Other issues involved in decision tree learning*

Some Other Issues in Decision Tree Learning

- Convert decision trees to a set of rules
- How to deal with continuous attributes
- How to scale up decision tree learning
- Look-ahead approaches
  - Search for best sequence of individual tests.
- Multiple attributes per test
  - These approaches tend to have greatly increased complexities (i.e., larger search spaces).

Convert Decision Trees to a Set of Rules

- Each branch from the root to a leaf can be transformed into a if-then rule.



- If (Outlook is Sunny and Humidity is High), then class is No.
- If (Outlook is Sunny and Humidity is Normal), then class is Yes.
- If (Outlook is Overcast), then class is Yes.
- ……

How to deal with continuous attributes

Two ways:

- Discretization before learning decision tree
  - Converts a continuous attribute into a discrete attribute by partitioning the range of the continuous attribute into intervals.
  - Interval labels can then be used to replace actual data values.
- Dynamic discretization
  - Dynamically split the value range into two sub-ranges during the tree learning process

Dynamic Discretization

- Dynamically split the value range into two sub-ranges and each descendent node corresponds to a sub-range.
  - Choose to split at the middle value between two examples which are in different categories

Temperature	PlayTennis?
40	No
48	No
60	Yes
70	Yes
80	Yes
90	No

The possible split points are $\frac{48+60}{2} = 54$ and $\frac{80+90}{2} = 85$

  - Evaluate the binary splitting points using the splitting criterion used for selecting attribute. For example, choose the splitting point that leads to the best information gain.

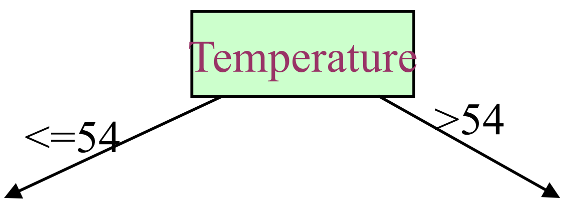
Dynamic Discretization

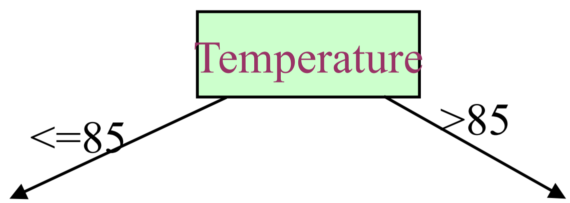
Example:

Temperature	PlayTennis?
40	No
48	No
60	Yes
70	Yes
80	Yes
90	No

The possible split points are $\frac{48+60}{2} = 54$ and $\frac{80+90}{2} = 85$

Possible splits:





Choose the one that gives the better information gain to compete with other attributes

Scalable Decision Tree Learning Methods

- Scalability: deal with millions of examples and hundreds of attributes with reasonable speed
- Most algorithms assume data can fit in memory.
- Data mining research contributes to the scalability issue, especially for decision trees.
- Successful examples
  - SLIQ (Mehta *et al.*, 1996)
  - SPRINT (Shafer *et al.*, 1996)
  - RainForest (Gehrke, *et al.*, 1998)

Some Other Issues in Decision Tree Learning

- Convert decision trees to a set of rules
- How to deal with continuous attributes
- How to scale up decision tree learning
- *Look-ahead approaches*
  - *Search for best sequence of individual tests.*
- *Multiple attributes per test/node*
  - *These approaches tend to have greatly increased complexities (i.e., larger search spaces).*

Decision Trees: Strengths

- Comprehensibility: Small trees are highly interpretable and intuitive for humans
- Fast classification
- Relatively fast induction
- Mature technology

Decision Trees: Weaknesses

- Trees can become incomprehensible when their size grows.
  - Rules converted from a tree
    - are *mutually exclusive,*

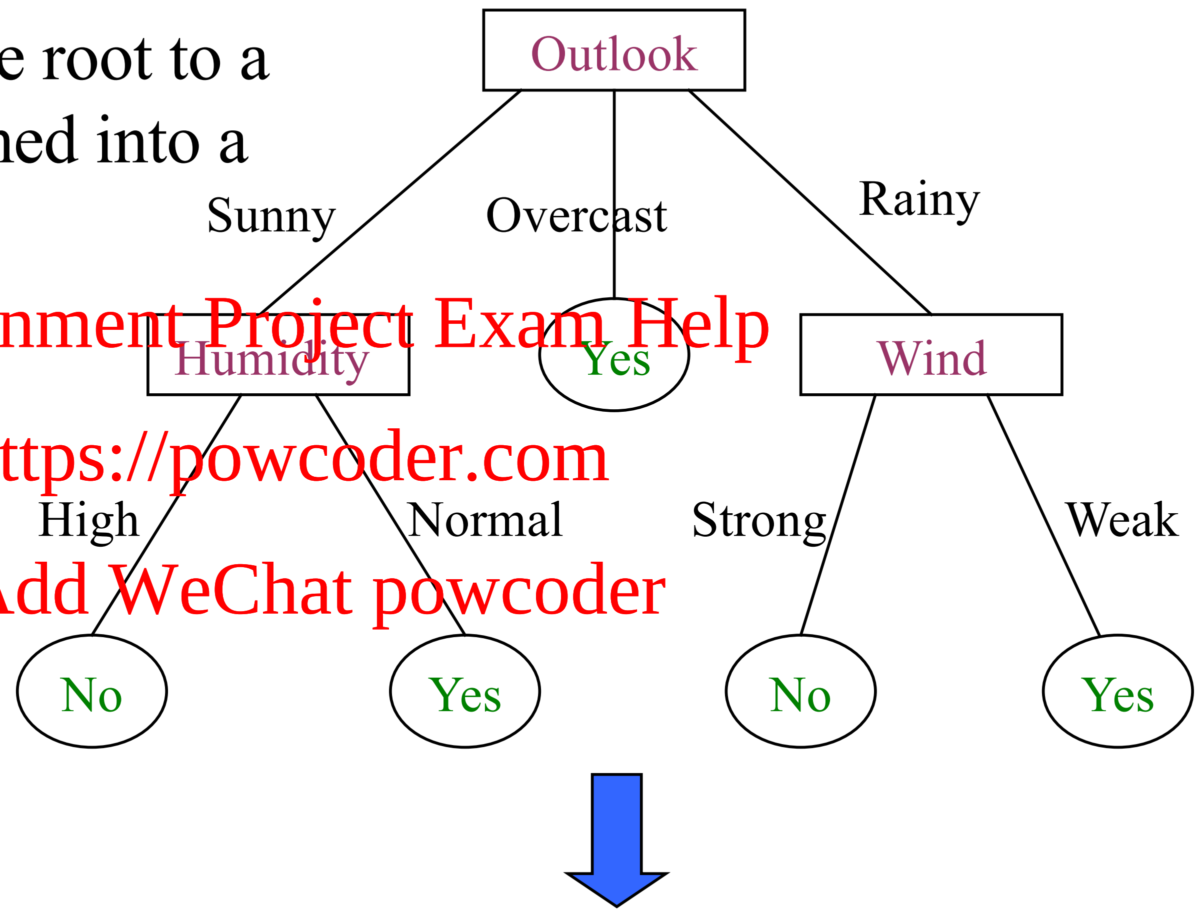
    - share at least one attribute (the root)

  - Thus, the size of the tree can grow much larger than the logic needed for overlapping rules.

  - As an example, in a successful application of ID3 to a chess end game (Mitchie, 86), the tree representation could not be understood at all even by the chess experts.
- When using only one attribute at each internal node, trees are limited to axis-parallel partitions of the instance space

Convert Decision Trees to a Set of Rules

- Each branch from the root to a leaf can be transformed into a if-then rule.



- If (Outlook is Sunny and Humidity is High), then class is No.
- If (Outlook is Sunny and Humidity is Normal), then class is Yes.
- If (Outlook is Overcast), then class is Yes.
- ……

Summary of Decision Tree Learning

- Decision tree represents classification knowledge
- Decision tree learning is a top-down recursive partitioning process. It is a two phase process if post-pruning is used:
  - Tree building
    - An attribute selection criterion (also called splitting criterion) is used: information gain, gain ratio, gini index, etc.
  - Post-pruning tree
    - Reduced-error (Quinlan, 87)
    - Error-complexity (CART, 84)
    - Pessimistic error (Quinlan, 86)
- Some other issues

Next Class

- Data Preprocessing (Chapter 3)