

Lecture Contents

- What is classification?
- K-nearest neighbour Classifier
- Bayesian Classification
- Decision Tree(ID3, C4.5)
- Evaluation of classifiers
- Summary

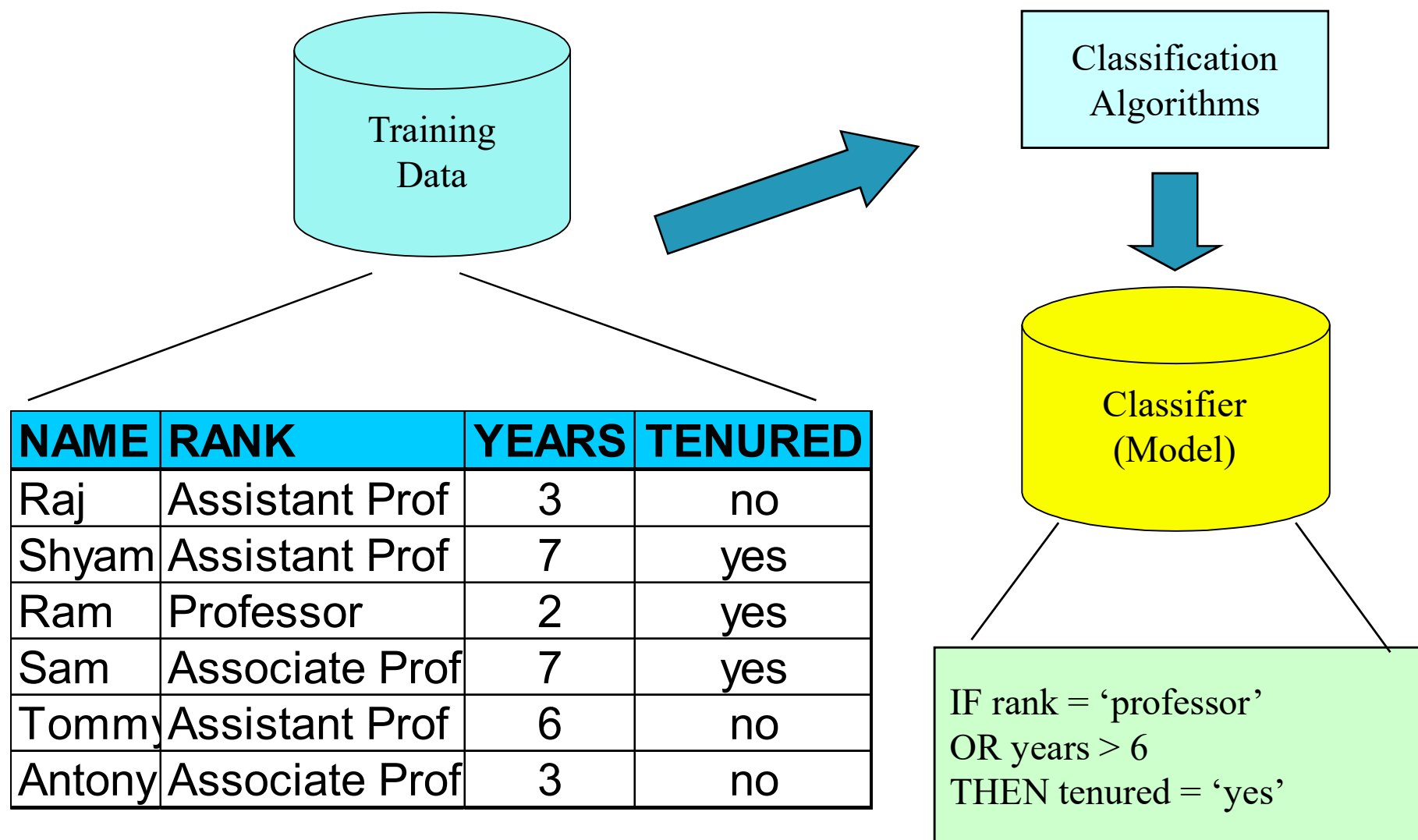
Classification: Definition

- Given a collection of records
 - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
- Classification is *supervised learning*: The training data are accompanied by labels indicating the class of the observations
- **Classification Vs Regression:**
 - Class. predicts categorical class labels while Reg. predicts continuous-valued functions
 - classifies data (constructs a model) based on the training set and the values (*class labels*) in a classifying attribute and uses it in classifying new data
- **Typical Applications**
 - Speech Recognition
 - target marketing
 - medical diagnosis
 - Pattern Recognition
 - Classification of students in a class

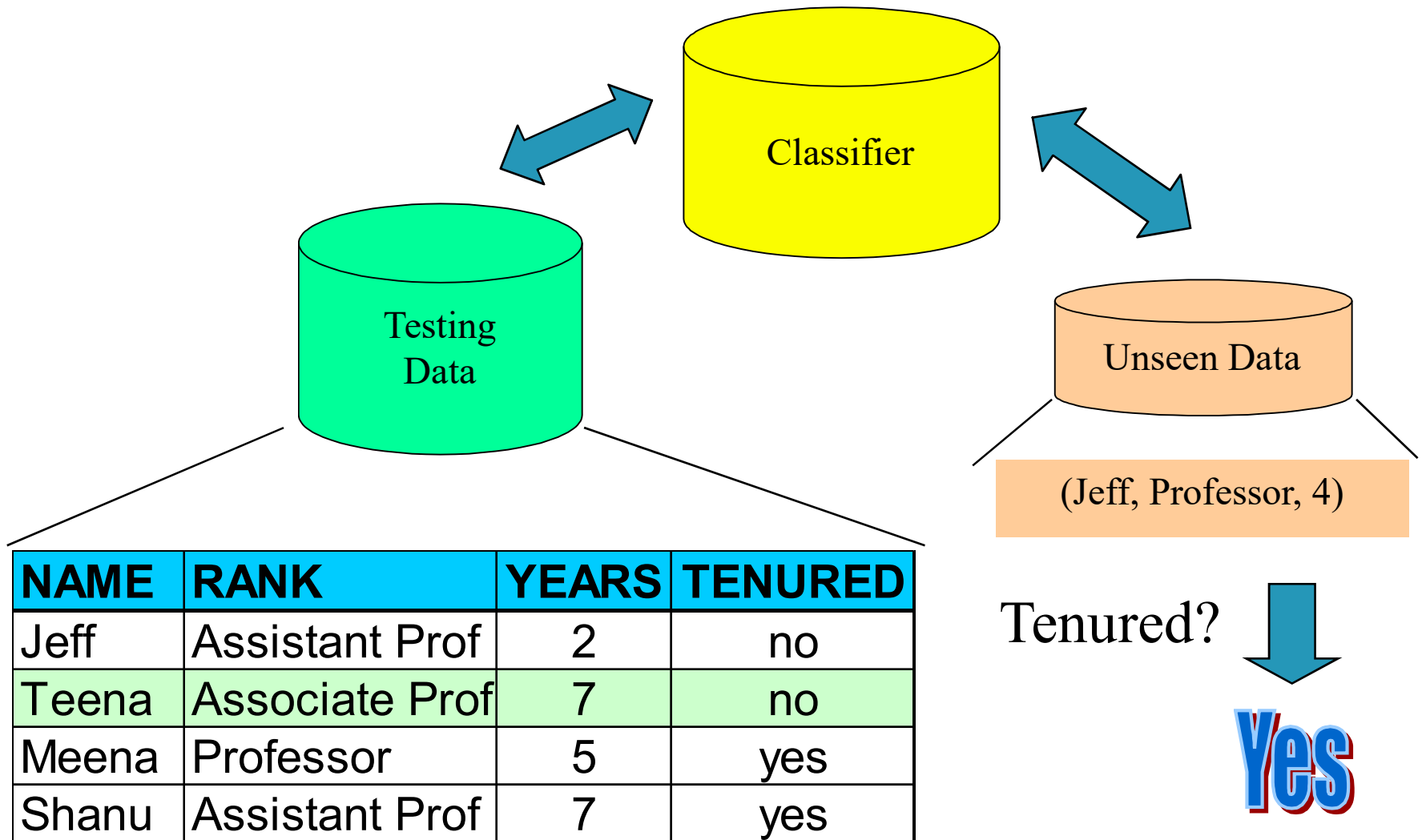
Classification—A Two-Step Process

- **Model construction:** describing a set of predetermined classes
 - Each record contains a set of *attributes*, one of the attributes is the *class*.
 - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
 - The set of tuples used for model construction: **training set**
 - The model is represented as classification rules, decision trees, or mathematical formulae
- **Model usage:** for classifying future or unknown objects
 - Estimate accuracy of the model
 - The known label of test sample is compared with the classified result from the model
 - **Accuracy rate** is the percentage of test set samples that are correctly classified by the model
 - **Test set** is independent of training set, otherwise over-fitting will occur

Classification Process: Model Construction



Classification Process : Use the Model in Prediction



Issues : Data Preparation regarding classification

- Preprocessing should be done to improve accuracy, efficiency and scalability of classification algorithms.
 - Preprocess data in order to reduce noise and handle missing values
 - Relevance analysis (feature selection)
 - Remove the irrelevant or redundant attributes
 - Data transformation
 - Generalize and/or normalize data

Comparison of class. methods

- Predictive Accuracy
- Speed
- Robustness
- Scalability
- Interpretability
- Goodness of rules
 - decision tree size
 - compactness of classification rules

Classification Using Distance

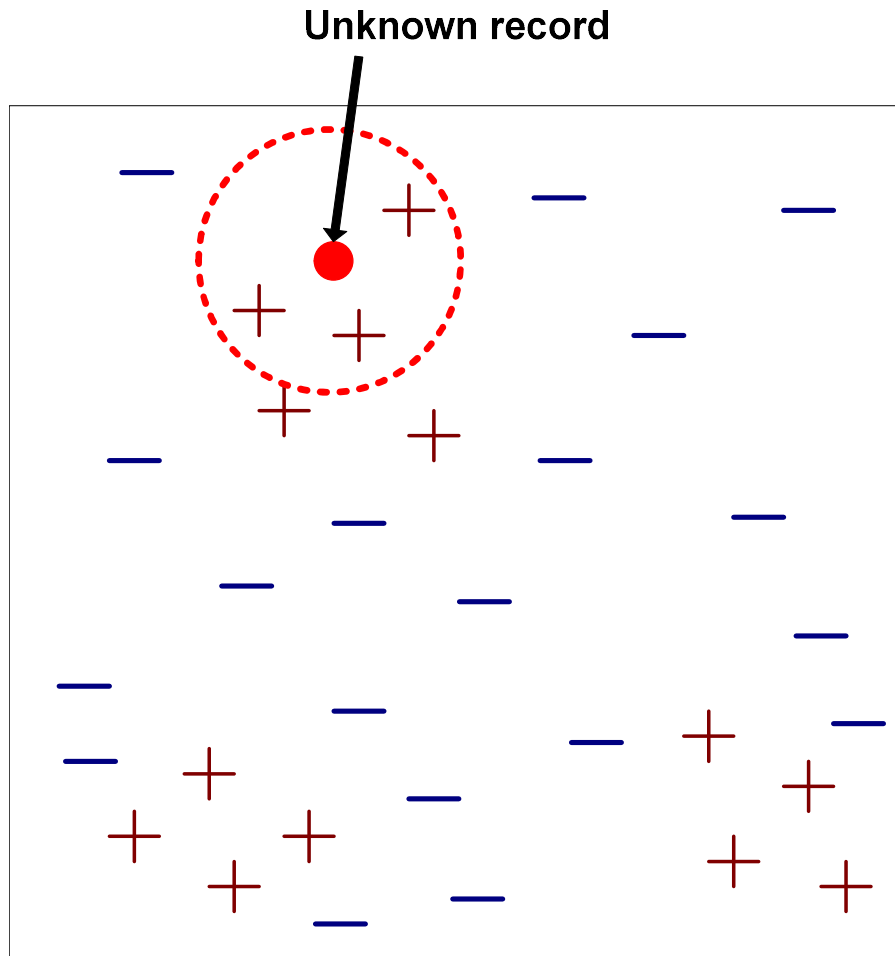
- Place items in class to which they are “closest”.
- Must determine distance between an item and a class.
- Classes represented by
 - ***Centroid***: Central value.
 - ***Medoid***: Representative point.
 - Individual points
- Algorithm: KNN

Nearest Neighbor Classifiers

Algorithm :

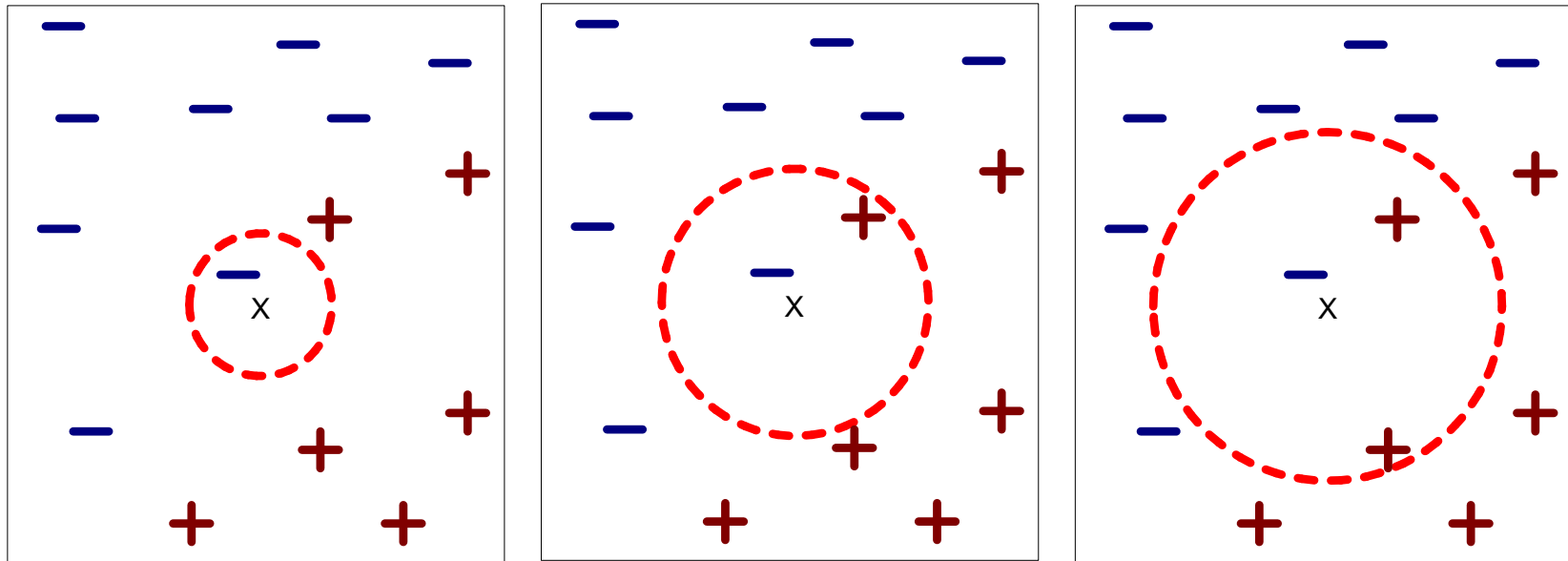
1. A positive integer k is specified, along with a new sample
2. We select the k entries in our database which are closest to the new sample
3. We find the most common classification of these entries
4. This is the classification we give to the new sample

Nearest-Neighbor Classifiers



- Requires three things
 - The set of stored records
 - **Distance Metric** to compute distance between records
 - The value of **k , the number of nearest neighbors** to retrieve
- To classify an unknown record:
 - **Compute distance** to other training records
 - Identify **k** nearest neighbors
 - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor



(a) 1-nearest neighbor

(b) 2-nearest neighbor

(c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

Nearest Neighbor Classification

- Compute distance between two points:
 - Euclidean distance

$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
 - take the majority vote of class labels among the k-nearest neighbors

What is the value of K?

Rule of thumb:

- $K \leq \text{square root}(\text{No of training Items})$
- K should not be multiple of no. of decision attribute (Class Attribute) .E.g.In binary classification k should not be multiple of 2.
- Commercial algo. use $k=10$

Discussion on the k -NN Algorithm

- It is good when no of samples are large.
- It is lazy learner because it does not build model explicitly.
- $O(q)$ for each tuple to be classified. (Here q is the size of the training set.)
- The k -NN algorithm for continuous-valued target functions
 - Calculate the mean values of the k nearest neighbors
- **Curse of dimensionality:** distance between neighbors could be dominated by irrelevant attributes/ all vectors are almost equidistant to the query vector .
 - To overcome it, axes stretch or elimination of the least relevant attributes.
 - Normalize the attributes

Bayesian Classifiers

- Probabilistic learning: Calculate explicit probabilities for hypothesis, among the most practical approaches to certain types of learning problems
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.
- Probabilistic prediction: Predict multiple hypotheses, weighted by their probabilities

Bayesian classification(3)

- The classification problem may be formalized using a-posteriori probabilities:
- $P(C|X)$ = prob. that the sample tuple $X = \langle x_1, \dots, x_n \rangle$ is of class C .
- E.g. $P(\text{class} = N \mid \text{outlook} = \text{sunny}, \text{temperature} = \text{mild}, \text{windy} = \text{true}, \text{humidity} = \text{high}, \dots)$
- Idea: assign to sample X the class label C such that $P(C|X)$ is maximal

Estimating a-posteriori probabilities

- Bayes theorem:

$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$

- $P(X)$ is constant for all classes
- $P(C)$ = relative freq of class C samples
- C such that $P(C|X)$ is maximum =
C such that $P(X|C) \cdot P(C)$ is maximum
- Problem: computing $P(X|C)$ is unfeasible!

Naïve Bayesian Classification

- If a given tuple $X = \langle x_1, \dots, x_n \rangle$ is having multiple attributes, computing $P(X|C)$ is infeasible! in Bayes theorem:

$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$

- Solution :
 - Naïve assumption: attribute independence

$$P(x_1, \dots, x_n | C) = P(x_1 | C) \cdot \dots \cdot P(x_n | C)$$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

What will the class of following sample:

X = <rain, hot, high, false>

What will the class of following sample: $X = \langle \text{rain, hot, high, false} \rangle$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

$P(C = \text{"P"} | X)$

$P(C = \text{"N"} | X)$

Play-tennis example: estimating $P(x_i | C)$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

$$P(p) = 9/14$$

$$P(n) = 5/14$$

outlook	
$P(\text{sunny} p) = 2/9$	$P(\text{sunny} n) = 3/5$
$P(\text{overcast} p) = 4/9$	$P(\text{overcast} n) = 0$
$P(\text{rain} p) = 3/9$	$P(\text{rain} n) = 2/5$
temperature	
$P(\text{hot} p) = 2/9$	$P(\text{hot} n) = 2/5$
$P(\text{mild} p) = 4/9$	$P(\text{mild} n) = 2/5$
$P(\text{cool} p) = 3/9$	$P(\text{cool} n) = 1/5$
humidity	
$P(\text{high} p) = 3/9$	$P(\text{high} n) = 4/5$
$P(\text{normal} p) = 6/9$	$P(\text{normal} n) = 2/5$
windy	
$P(\text{true} p) = 3/9$	$P(\text{true} n) = 3/5$
$P(\text{false} p) = 6/9$	$P(\text{false} n) = 2/5$

Play-tennis example: classifying X

- An unseen sample $X = \langle \text{rain, hot, high, false} \rangle$
- $P(X|p) \cdot P(p) =$
 $P(\text{rain}|p) \cdot P(\text{hot}|p) \cdot P(\text{high}|p) \cdot P(\text{false}|p) \cdot P(p) =$
 $3/9 \cdot 2/9 \cdot 3/9 \cdot 6/9 \cdot 9/14 = 0.010582$
- $P(X|n) \cdot P(n) =$
 $P(\text{rain}|n) \cdot P(\text{hot}|n) \cdot P(\text{high}|n) \cdot P(\text{false}|n) \cdot P(n) =$
 $2/5 \cdot 2/5 \cdot 4/5 \cdot 2/5 \cdot 5/14 = 0.018286$
- Sample X is classified in class n (don't play)

Summary of Naïve Bayes Classifier

- How to handle continuous attributes?

$$P(X_i = x_i | Y = y_j) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp \left(-\frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2} \right)$$

- Naïve Bayes Classifier very useful for text classification like Spam detection .

Summary of Naïve Bayes Classifier

- Independence hypothesis makes computation possible.

- Naïve assumption: attribute independence

$$P(x_1, \dots, x_k | C) = P(x_1 | C) \cdot \dots \cdot P(x_k | C)$$

- Independence hypothesis yields optimal classifiers when satisfied but is seldom satisfied in practice, as attributes (variables) are often correlated.
- Attempts to overcome this limitation:
 - Bayesian networks, that combine Bayesian reasoning with causal relationships between attributes

Lecture Contents

- What is classification?
- Bayesian Classification
- Summary of Naïve Bayes classifier
- Decision Tree(ID3, C4.5)
- Evaluation of classifiers
- Summary

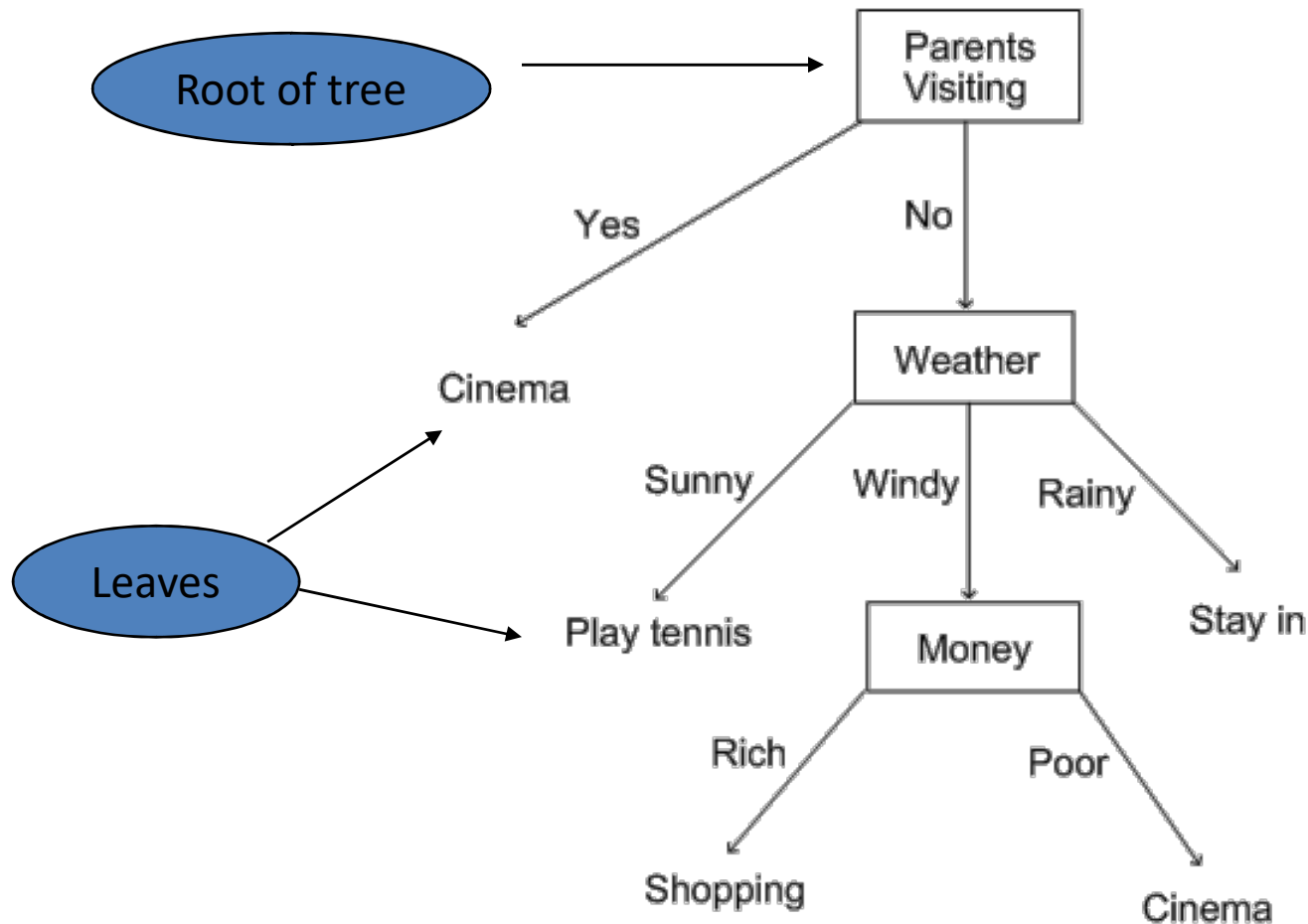
Classification by Decision Tree Induction

- Decision tree
 - A flow-chart-like tree structure
 - Internal node denotes a test on an attribute
 - Branch represents an outcome of the test
 - Leaf nodes represent class labels or class distribution
- Decision tree generation consists of two phases
 - Tree construction
 - At start, all the training examples are at the root
 - Partition examples recursively based on selected attributes
 - Tree pruning
 - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample
 - Test the attribute values of the sample against the decision tree

Example

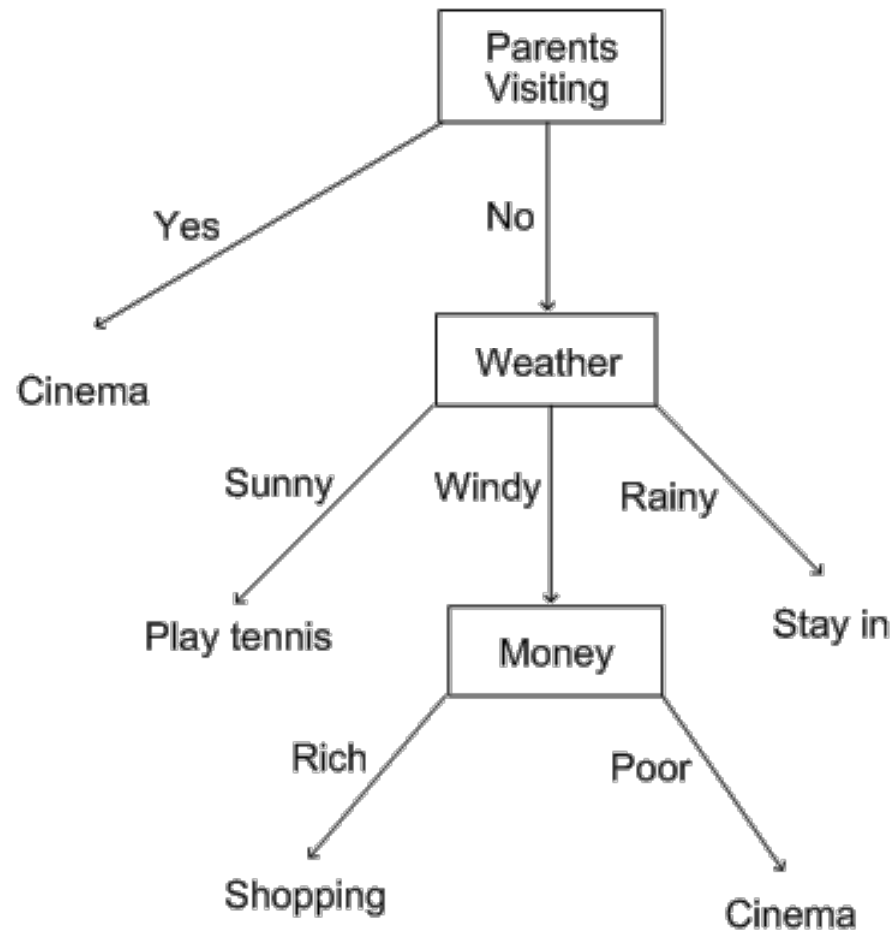
Weekend (Example)	Weather	Parents	Money	Decision (Category)
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay in
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

Written as a Decision Tree



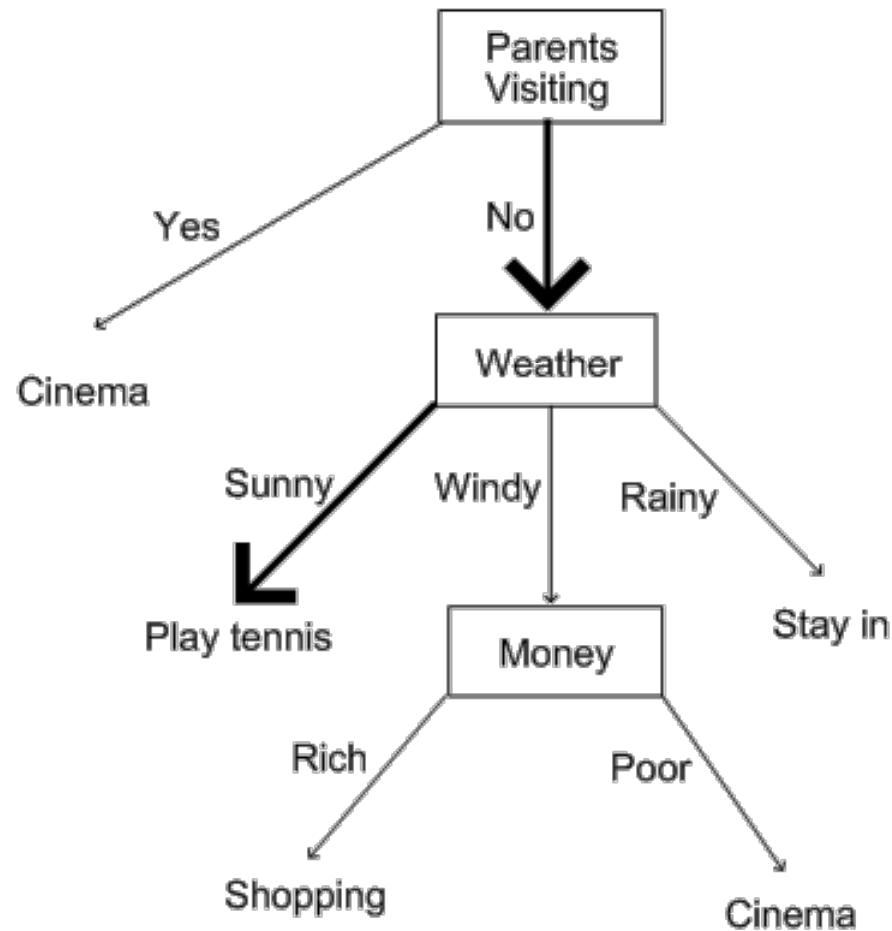
Using the Decision Tree

(No parents on a Sunny Day)



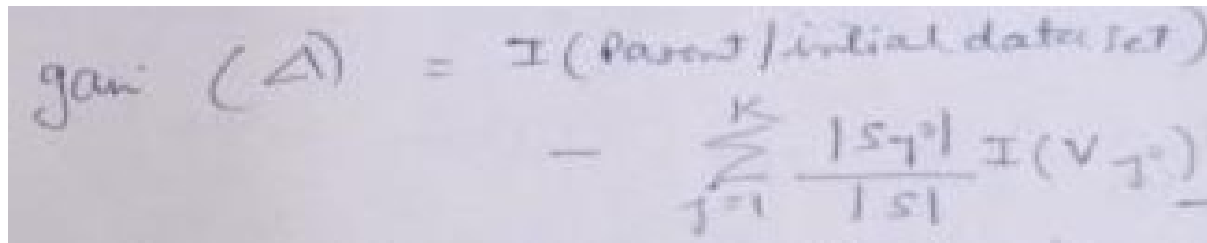
Using the Decision Tree

(No parents on a Sunny Day)



Measures for Selecting Best Split

- The measures are based on degree of impurity of the child node. The smaller the degree of impurity, the more skewed class distribution.
 - Distribution (0,1) has zero impurity.
 - Distribution(.5,.5) has the highest impurity.
- **Different impurity measures**
 - Entropy
 - Gini
 - Classification Error
- To choose best split :
- **Gain = Degree of parent node(before split) - degree of child node(After split) is computed . And highest is selected.**


$$\text{gain}(A) = I(\text{parent}/\text{initial data set}) - \sum_{j=1}^K \frac{|S_j|}{|S|} I(V_j)$$

•Where, k is no of attribute values for a given attribute.

• I: Entropy, Gini, Classification Error

Iterative Dichotomiser (ID3) Algorithm - Overview

- The major question in decision tree learning
 - Which nodes to put in which positions
 - Including the root node and the leaf nodes
- ID3 uses a measure called Information Gain
 - Based on a notion of entropy
 - “Impurity in the data”
 - Used to choose which node to put in next
- Node with the highest information gain is chosen
 - When there are no choices, a leaf node is put on

Entropy – General Idea

- From Tom Mitchell's book:
 - “In order to define information gain precisely, we begin by defining a measure commonly used in information theory, called entropy that characterizes the (im)purity of an arbitrary collection of examples”
- Want a notion of impurity in data

Entropy - Formulae

- Given a set of examples, S
- For examples in a binary categorization
 - Where p_+ is the proportion of positives
 - And p_- is the proportion of negatives

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

- For examples in categorizations c_1 to c_n
 - Where p_n is the proportion of examples in c_n

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

Entropy - Explanation

- Each category adds to the whole measure
- When p_i is near to 1
 - (Nearly) all the examples are in this category
 - So it should score low for its bit of the entropy
 - $\log_2(p_i)$ gets closer and closer to 0
 - And this part *dominates* the overall calculation
 - So the overall calculation comes to nearly 0 (which is good)
- When p_i is near to 0
 - (Very) few examples are in this category
 - So it should score low for its bit of the entropy
 - $\log_2(p_i)$ gets larger (more negative), but does not dominate
 - Hence overall calculation comes to nearly 0 (which is good)

Information Gain

- Calculate $\text{Gain}(S, A)$
 - Estimates the reduction in entropy we get if we know the value of attribute A for the examples in S

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

An Example Calculation of Information Gain

- Suppose we have a set of examples
 - $S = \{s_1, s_2, s_3, s_4\}$
 - In a binary categorization
 - With one positive example and three negative examples
 - The positive example is s_1
- And Attribute A
 - Which takes values v_1, v_2, v_3
- S_1 takes value v_2 for A, S_2 takes value v_2 for A
 S_3 takes value v_3 for A, S_4 takes value v_1 for A

Sample No	Attribute(A)	Class
S1	v2	p
S2	v2	n
S3	v3	n
S4	v1	n

First Calculate Entropy(S)

Sample No	Attribute(A)	Class
S1	v2	p
S2	v2	n
S3	v3	n
S4	v1	n

- Recall that

$$\text{Entropy}(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$$

- From binary categorisation, we know that

$$p_+ = 1/4 \text{ and } p_- = 3/4$$

- Hence

$$\begin{aligned} \text{Entropy}(S) &= -(1/4)\log_2(1/4) - (3/4)\log_2(3/4) \\ &= 0.811 \end{aligned}$$

- Note for users of old calculators:
 - May need to use the fact that $\log_2(x) = \ln(x)/\ln(2)$
- And also note that, by convention:
 - $0 \cdot \log_2(0)$ is taken to be 0

Calculate IG for each Value of A

Sample No	Attribute(A)	Class
S1	v2	p
S2	v2	n
S3	v3	n
S4	v1	n

- Remember that

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

- And that $S_v = \{\text{set of example with value } v \text{ for } A\}$
 - So, $S_{v1} = \{s_4\}$, $S_{v2} = \{s_1, s_2\}$, $S_{v3} = \{s_3\}$
- Now, $(|S_{v1}|/|S|) * Entropy(S_{v1})$

$$= (1/4) * (-(0/1) * \log_2(0/1) - (1/1) * \log_2(1/1))$$

$$= (1/4) * (0 - (1) * \log_2(1)) = (1/4)(0 - 0) = 0$$
- Similarly, $(|S_{v2}|/|S|) * Entropy(S_{v2}) = 0.5$ and $(|S_{v3}|/|S|) * Entropy(S_{v3}) = 0$

Final Calculation

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

- So, we add up the three calculations and take them from the overall entropy of S:
- Final answer for information gain:
 - $Gain(S, A) = 0.811 - (0 + 1/2 + 0) = 0.311$

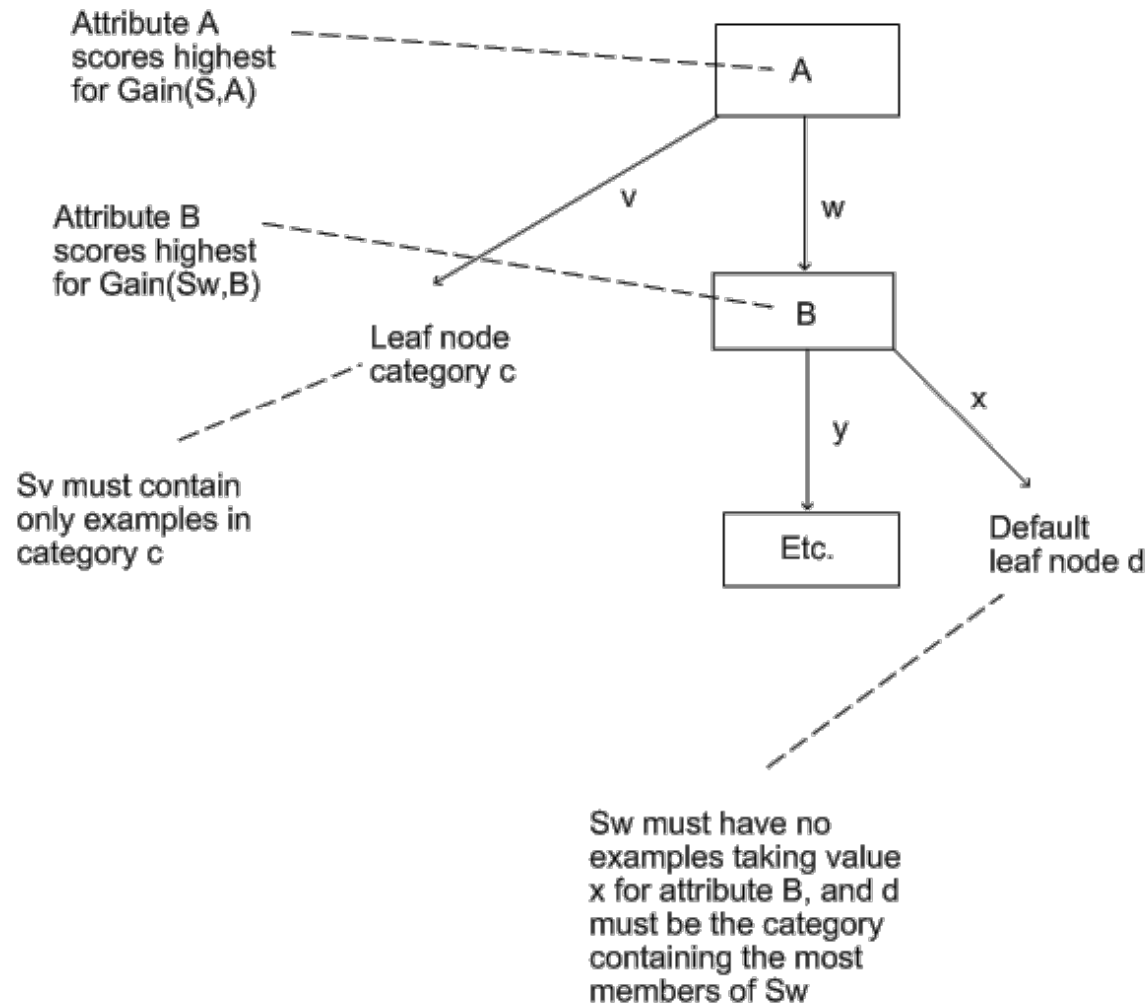
The ID3 Algorithm

- Given a set of examples, S
 - Described by a set of attributes A_i
 - Categorised into categories c_j
- 1. Choose the root node to be attribute A
 - Such that A scores highest for information gain
 - Relative to S , i.e., $\text{gain}(S, A)$ is the highest over all attributes
- 2. For each value v that A can take
 - Draw a branch and label each with corresponding v
 - Then see the options in the next slide!

The ID3 Algorithm

- For each branch you've just drawn (for value v)
 - If S_v only contains examples in category c
 - Then put that category as a leaf node in the tree
 - If S_v is empty
 - Then find the default category (which contains the most examples from S)
 - Put this default category as a leaf node in the tree
 - Otherwise
 - Remove A from attributes which can be put into nodes
 - Replace S with S_v
 - Find new attribute A scoring best for $\text{Gain}(S, A)$
 - Start again at part 2
- Make sure you replace S with S_v

Explanatory Diagram



Example 1

Sample training data to determine whether an animal lays eggs.

Independent/Condition attributes					Dependent/ Decision attributes
Animal	Warm-blooded	Feathers	Fur	Swims	Lays Eggs
Ostrich	Yes	Yes	No	No	Yes
Crocodile	No	No	No	Yes	Yes
Raven	Yes	Yes	No	No	Yes
Albatross	Yes	Yes	No	No	Yes
Dolphin	Yes	No	No	Yes	No
Koala	Yes	No	Yes	No	No

Approach

Independent/Condition attributes					Dependent/ Decision attributes
Animal	Warm-blooded	Feathers	Fur	Swims	Lays Eggs
Ostrich	Yes	Yes	No	No	Yes
Crocodile	No	No	No	Yes	Yes
Raven	Yes	Yes	No	No	Yes
Albatross	Yes	Yes	No	No	Yes
Dolphin	Yes	No	No	Yes	No
Koala	Yes	No	Yes	No	No

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

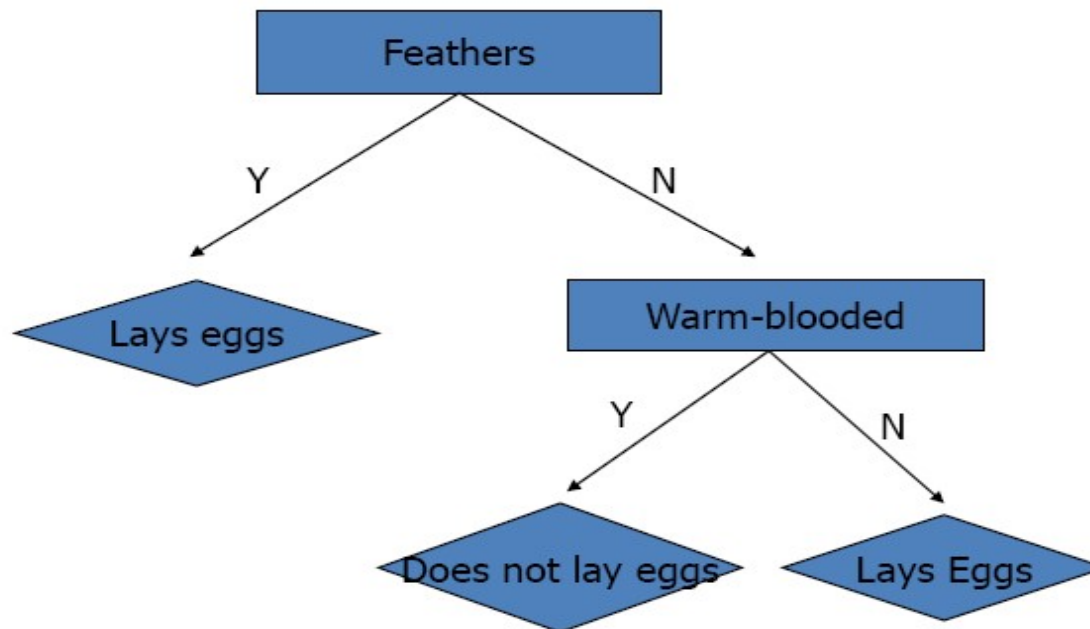
$$Entropy(S) = \sum_{i=1}^n -p_i \log_2(p_i)$$

Entropy =

$$-(4/6)\log_2(4/6) - (2/6)\log_2(2/6) \\ = 0.91829$$

Solution

The final decision tree will be:



Avoid Overfitting in Classification

- Doing better on the training set may actually lead to doing worse on a test set. This phenomenon is called overfitting.
- The generated tree may overfit the training data
 - Too many branches, some may reflect anomalies due to noise or outliers
 - Result is in poor accuracy for unseen samples
- Two approaches to avoid overfitting
 - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
 - Difficult to choose an appropriate threshold
 - Postpruning: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
 - Use a set of data different from the training data to decide which is the “best pruned tree”

Problems with ID3

- DT algo. are slow for large and noisy datasets
- Handling only categorical attributes
- Overfitting due to noise or irrelevant attributes
- No Backtracking

C4.5 Algorithm

- Allow for continuous-valued attributes
 - Split data into ranges .
- Handle missing attribute values
 - Missing data is ignored at the time of tree building. The classification of a record having missing values can be predicted based on what is known about the attributes values for other record
- Pruning trees after creation(Post Pruning)
 - C4.5 goes back through the tree once it's been created and attempts to remove branches that do not help by replacing them with leaf nodes (allows backtracking)
 - Subtree replacement & Subtree raising
- Splitting
 - For splitting purposes, it uses the largest **GainRatio** that ensures the average information gain.
- Rules
 - Classification is done via trees or Rules

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2 \left(\frac{|D_j|}{|D|} \right)$$

– **GainRatio(A) = Gain(A)/SplitInfo(A)**

- The attribute with the maximum gain ratio is selected as the splitting attribute

Example

<i>RID</i>	<i>age</i>	<i>income</i>	<i>student</i>	<i>credit_rating</i>	<i>Class: buys_computer</i>
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$SplitInfo_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2 \left(\frac{|D_j|}{|D|} \right)$$

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2 \left(\frac{4}{14} \right) - \frac{6}{14} \times \log_2 \left(\frac{6}{14} \right) - \frac{4}{14} \times \log_2 \left(\frac{4}{14} \right) = 1.557$$

$$GainRatio(A) = Gain(A)/SplitInfo(A)$$

$$gain_ratio(income) = 0.029/1.557 = 0.019$$

Continuous /Numeric Attributes

- Split on temperature attribute:

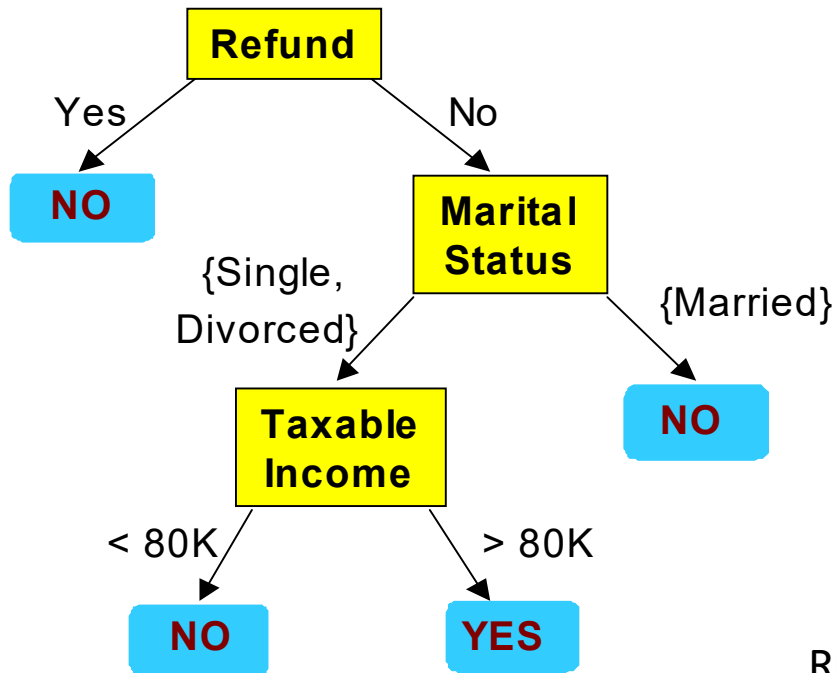
64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

- E.g. temperature < 71.5 : yes/4, no/2
temperature ≥ 71.5 : yes/5, no/3

- Calculate Information Gain for different split and choose the one which gives maximum gain.

- Place split points halfway between values

From Decision Trees To Rules



Classification Rules

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single, Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Single, Divorced}, Taxable Income>80K) ==> Yes

(Refund=No, Marital Status={Married}) ==> No

Rules are mutually exclusive and exhaustive

Rule set contains as much information as the tree

Other Measures of Best Split for Decision Tree based Classifier

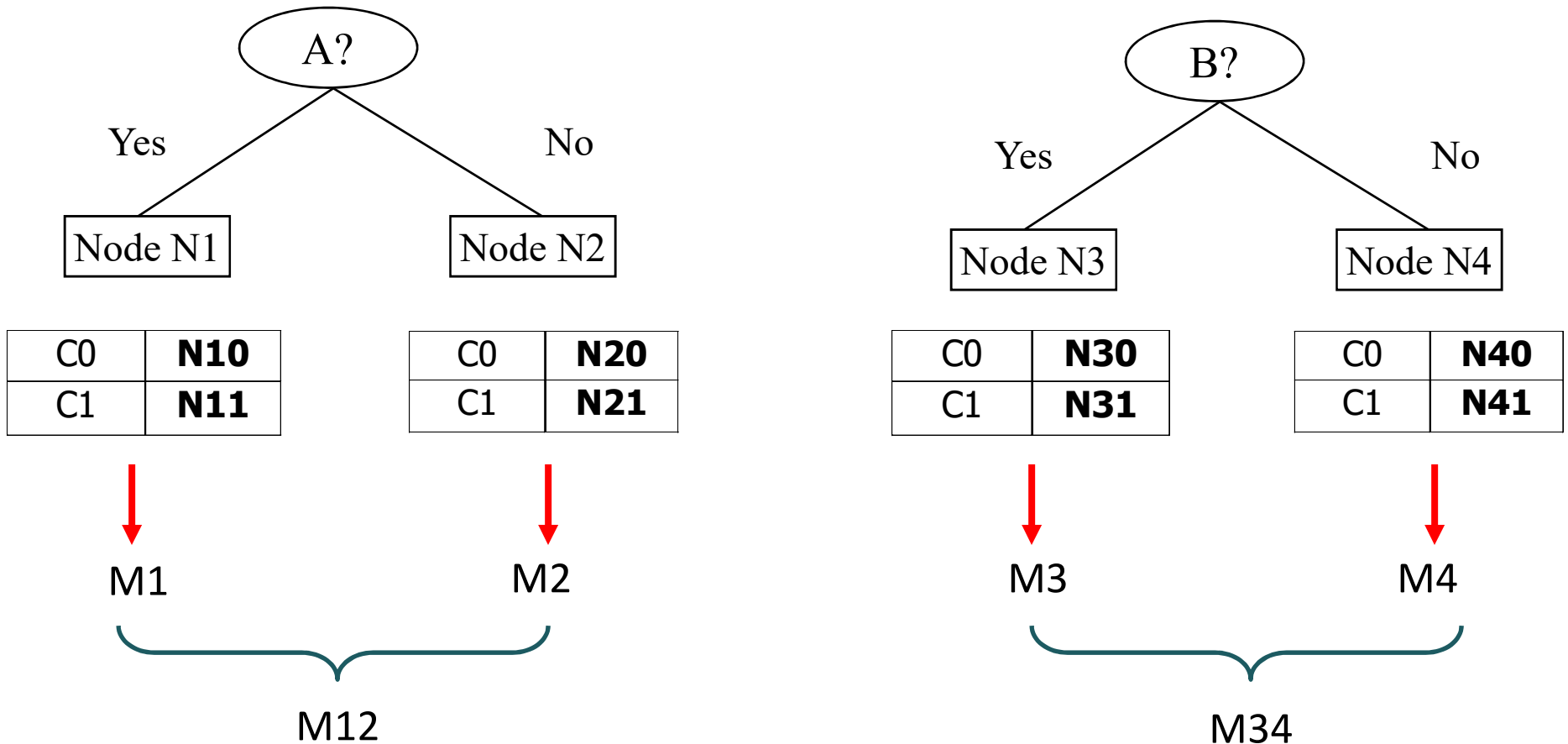
- Gini Index
- classification error

How to Find the Best Split

Before Splitting:

C0	N00
C1	N01

→ M0



$$\text{Gain} = M0 - M12 \text{ vs } M0 - M34$$

Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

(NOTE: $p(j | t)$ is the relative frequency of class j at node t).

- Maximum $(1 - 1/n_c)$ when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information

Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$Gini = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$Gini = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$Gini = 1 - (2/6)^2 - (4/6)^2 = 0.444$$

Splitting Based on GINI

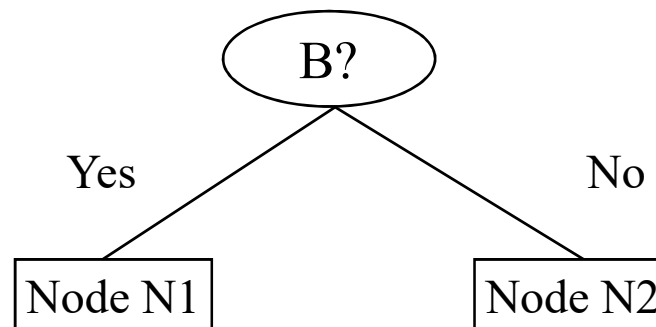
- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i ,
 n = number of records at node p .

Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
 - Larger and Purer Partitions are sought for.



	Parent
C1	6
C2	6
Gini = 0.500	

$$\begin{aligned} \text{Gini}(N1) &= 1 - (4/7)^2 - (3/7)^2 \\ &= 0.4898 \end{aligned}$$

$$\begin{aligned} \text{Gini}(N2) &= 1 - (2/5)^2 - (3/5)^2 \\ &= 0.48 \end{aligned}$$

	N1	N2
C1	4	2
C2	3	3
Gini=0.486		

$$\begin{aligned} \text{Gini(Children)} &= 7/12 * 0.4898 + \\ &\quad 5/12 * 0.480 \\ &= 0.486 \end{aligned}$$

Categorical Attributes: Computing Gini Index

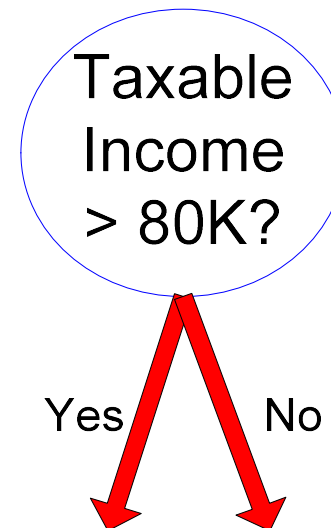
- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
 - Number of possible splitting values
= Number of distinct values
- Each splitting value has a count matrix associated with it
 - Class counts in each of the partitions, $A < v$ and $A \geq v$
- Simple method to choose best v
 - For each v , scan the database to gather count matrix and compute its Gini index
 - Computationally Inefficient! Repetition of work.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
 - Sort the attribute on values
 - Linearly scan these values, each time updating the count matrix and computing gini index
 - Choose the split position that has the least gini index

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

Splitting Criteria based on Classification Error

- Classification error at a node t :

$$Error(t) = 1 - \max_i P(i | t)$$

- Measures misclassification error made by a node.
 - Maximum $(1 - 1/n_c)$ when records are equally distributed among all classes, implying least interesting information
 - Minimum (0.0) when all records belong to one class, implying most interesting information

Examples for Computing Error

$$Error(t) = 1 - \max_i P(i | t)$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$Error = 1 - \max(0, 1) = 1 - 1 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

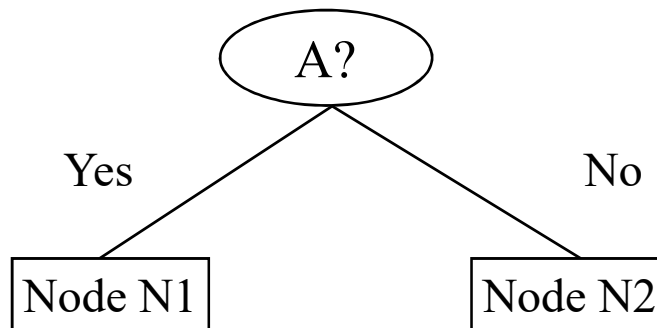
$$Error = 1 - \max(1/6, 5/6) = 1 - 5/6 = 1/6$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$Error = 1 - \max(2/6, 4/6) = 1 - 4/6 = 1/3$$

Misclassification Error vs Gini



	Parent
C1	7
C2	3
Gini = 0.42	

$$\text{Error} = 1 - \max(7/10, 3/10) = 0.3$$

$$\begin{aligned} \text{Gini(Children)} &= 3/10 * 0 \\ &+ 7/10 * 0.489 \\ &= 0.342 \end{aligned}$$

Gini improves !!

$$\begin{aligned} \text{Error(Children)} &= 3/10 * 0 \\ &+ (7/10) * (3/7) \\ &= 0.3 \end{aligned}$$

$$\begin{aligned} \text{Gini(N1)} &= 1 - (3/3)^2 - (0/3)^2 = 0 \end{aligned}$$

$$\text{Error (N1)} = 1 - \max(3/3, 0/3) = 0$$

$$\begin{aligned} \text{Gini(N2)} &= 1 - (4/7)^2 - (3/7)^2 = 0.489 \end{aligned}$$

$$\text{Error (N2)} = 1 - \max(4/7, 3/7) = 3/7$$

	N1	N2
C1	3	4
C2	0	3

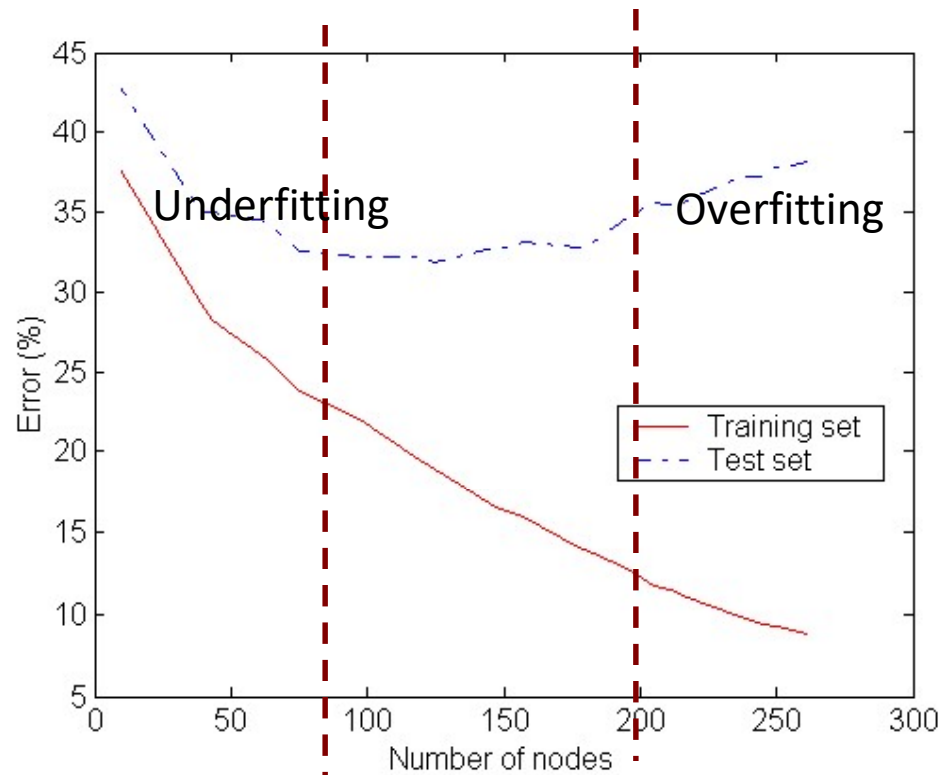
Decision Tree Based Classification

- Advantages:
 - Inexpensive to construct
 - Extremely fast at classifying unknown records
 - Easy to interpret for small-sized trees
 - Accuracy is comparable to other classification techniques for many simple data sets

Practical Issues of Classification

- Underfitting and Overfitting

Underfitting and Overfitting



Underfitting: when model is **too simple**, both training and test errors are large

Overfitting: when model is **too complex**(leaf nodes can be expanded until it perfectly fits the training data) , training error continues to decrease but test error begins to increase .

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
 - The model does not generalize well
- Need new ways for estimating errors.
 - Using validation set:
 - Split data into training, validation, test
 - Use validation dataset to estimate generalization error
 - Drawback: less data for training.
 - Using Pruning Approaches
 - Prepruning
 - Post-pruning

Pruning

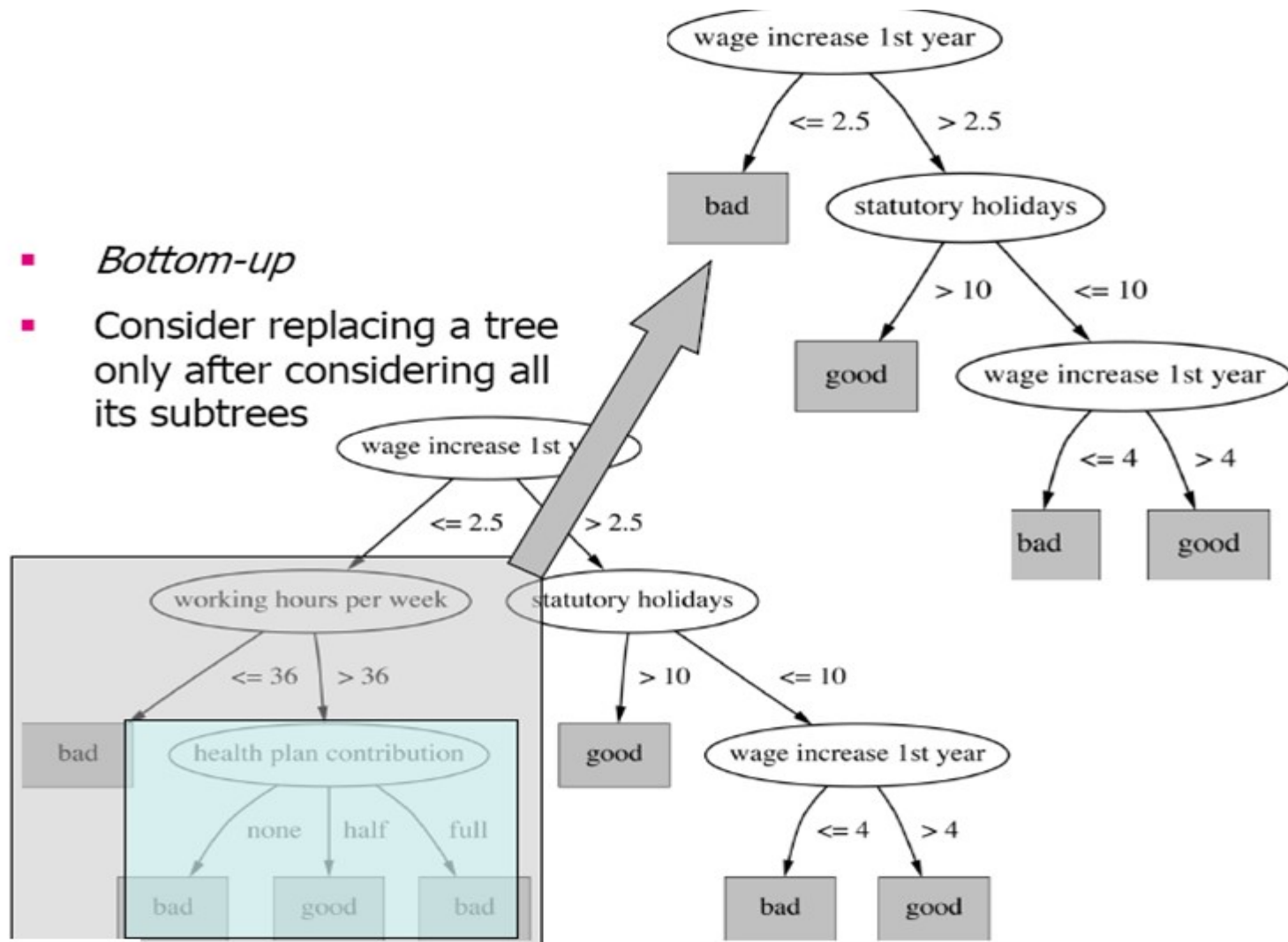
- First, build full tree
- Then, prune it
 - Fully-grown tree shows all attribute interactions
- Problem: some subtrees might be due to chance effects
- Two pruning operations:
 1. *Subtree replacement*
 2. *Subtree raising*
- Possible strategies:
 - error estimation
 - significance testing
 - MDL principle

Prepruning

- Based on statistical significance test
 - Stop growing the tree when there is no *statistically significant* association between any attribute and the class at a particular node
 - Most popular test: *chi-squared test*
 - ID3 used chi-squared test in addition to information gain
 - Only statistically significant attributes were allowed to be selected by information gain procedure
-
- Use **information gain**, **entropy** , and so on can be used to split the partitions. If partition falls below a **threshold**, then further partitioning is halted. The leaf node may hold the most frequent class among the subset. It is difficult to choose **appropriate threshold**.

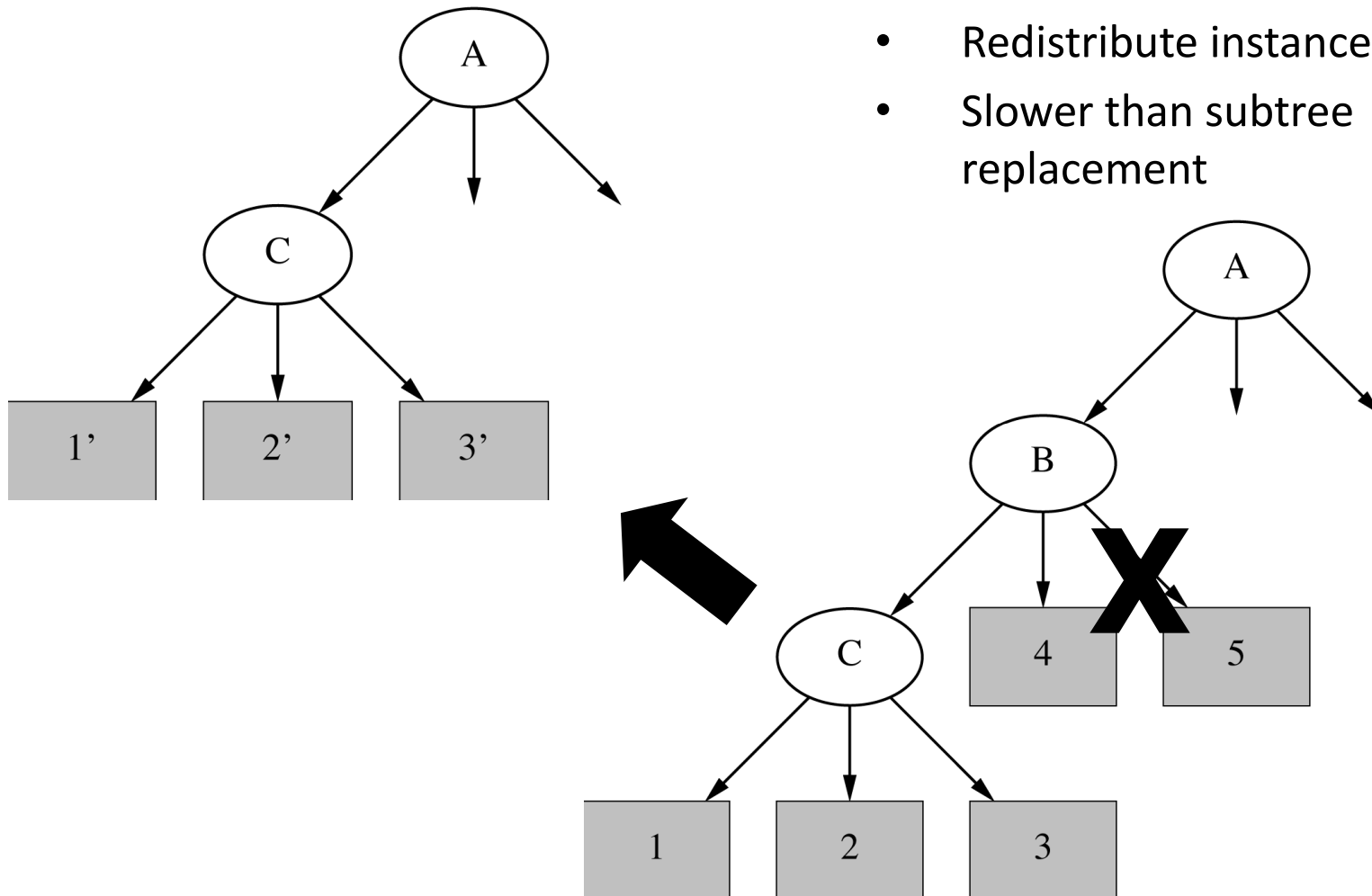
Postpruning :Subtree Replacement

- *Bottom-up*
- Consider replacing a tree only after considering all its subtrees



Postpruning : Subtree raising

- Delete node
- Redistribute instances
- Slower than subtree replacement



Model Evaluation

- Metrics for Performance Evaluation
 - How to evaluate the performance of a model?
- Methods for Performance Evaluation
 - How to obtain reliable estimates?
- Methods for Model Comparison
 - How to compare the relative performance among competing models?

Metrics for Performance Evaluation

- Focus on the predictive capability of a model
 - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)
b: FN (false negative)
c: FP (false positive)
d: TN (true negative)

Metrics for Performance Evaluation...

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}$$

Limitation of Accuracy

- Consider a 2-class problem
 - Number of Class 0 examples = 9990
 - Number of Class 1 examples = 10
- If model predicts everything to be class 0, accuracy is $9990/10000 = 99.9\%$
 - Accuracy is misleading because model does not detect any class 1 example

Other Measures

$$\text{Precision (p)} = \frac{a}{a + c}$$

$$\text{Recall (r)} = \frac{a}{a + b}$$

$$\text{F - measure (F)} = \frac{2rp}{r + p} = \frac{2a}{2a + b + c}$$

	PREDICTED CLASS		
ACTUAL CLASS		Class= Yes	Class= No
	Class= Yes	a (TP)	b (FN)
	Class= No	c (FP)	d (TN)

- **Specificity(True Negative Rate)**= $TN / (TN + FP)$
- Recall is also known as **sensitivity(True Positive Rate)**
- Precision is ability of a classification model to return only relevant instances
- Recall is ability of a classification model to identify all relevant instances.
- F1-measure is single metric that combines recall and precision using the harmonic mean.

Model Evaluation

- Metrics for Performance Evaluation
 - How to evaluate the performance of a model?
- Methods for Performance Evaluation
 - How to obtain reliable estimates?

Cross Validation

- There are two possible goals in cross-validation:
 - To estimate performance of the learned model from available data using one algorithm. In other words, to gauge the generalizability of an algorithm.
 - To compare the performance of two or more different algorithms and find out the best algorithm for the available data, or alternatively to compare the performance of two or more variants of a parameterized model.
- Type of cross-validation procedures:
 - Resubstitution Validation
 - Hold-Out Validation
 - K-Fold Cross-Validation
 - Leave-One-Out Cross-Validation
 - Repeated K-Fold Cross-Validation

Resubstitution Validation

- In resubstitution validation, the model is learned from all the available data and then tested on the same set of data.
- This validation process uses all the available data but suffers seriously from over-fitting.
- That is, the algorithm might perform well on the available data yet poorly on future unseen test data.

Hold-Out Validation (1)

- To avoid over-fitting, an independent test set is preferred.
- A natural approach is to split the available data into two non-overlapped parts:
 - one for training and
 - the other for testing.
- The test data is held out and not looked at during training.
- Hold-out validation avoids the overlap between training data and test data, yielding a more accurate estimate for the generalization performance of the algorithm.

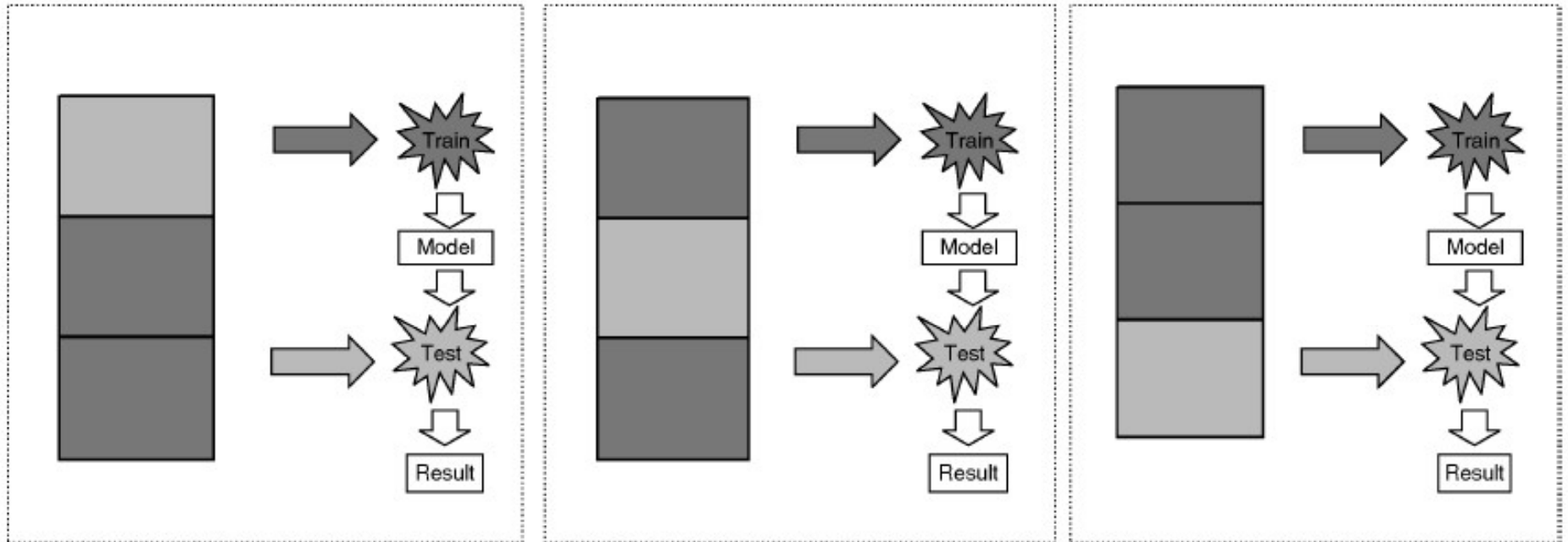
Hold-Out Validation(2)

- The downside is that this procedure does not use all the available data and the results are highly dependent on the choice for the training/test split.
- The instances chosen for inclusion in the test set may be too easy or too difficult to classify and this can skew the results.
- Furthermore, the data in the test set may be valuable for training and if it is heldout prediction performance may suffer, again leading to skewed results.

k-fold Cross Validation

- In k-fold cross-validation the data is first partitioned into k equally (or nearly equally) sized segments or folds.
- Subsequently k iterations of training and validation are performed such that within each iteration a different fold of the data is held-out for validation while the remaining k - 1 folds are used for learning.
- Fig. 1 demonstrates an example with k = 3. The darker section of the data are used for training while the lighter sections are used for validation.
- In data mining and machine learning 10-fold cross-validation (k = 10) is the most common.

k-fold Cross Validation



Cross-Validation. Figure 1. Procedure of three-fold cross-validation.

Leave-One-Out Cross-Validation

- Leave-one-out cross-validation (LOOCV) is a special case of k-fold cross-validation where k equals the number of instances in the data.
- In other words in each iteration nearly all the data except for a single observation are used for training and the model is tested on that single observation.
- An accuracy estimate obtained using LOOCV is known to be almost unbiased but it has high variance, leading to unreliable estimates [3].
- It is still widely used when the available data are very rare, especially in bioinformatics where only dozens of data samples are available.

Repeated K-Fold Cross-Validation

- To obtain reliable performance estimation or comparison, large number of estimates are always preferred.
- In k-fold cross-validation, only k estimates are obtained.
- A commonly used method to increase the number of estimates is to run k-fold cross-validation multiple times.
- The data is reshuffled and re-stratified before each round.

Summary

Validation method	Pros	Cons
Resubstitution Validation	Simple	Over-fitting
Hold-out Validation	Independent training and test	Reduced data for training and testing; Large variance
<i>k</i> -fold cross validation	Accurate performance estimation	Small samples of performance estimation; Overlapped training data; Elevated Type I error for comparison; Underestimated performance variance or overestimated degree of freedom for comparison
Leave-One-Out cross-validation	Unbiased performance estimation	Very large variance
Repeated <i>k</i> -fold cross-validation	Large number of performance estimates	Overlapped training and test data between each round; Underestimated performance variance or overestimated degree of freedom for comparison

References

- [1] Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.
- [2] Han, Jiawei, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.