Introduction to AI

Lecture 2: Traditional

Welcome to our course on Artificial Intelligence. This introductory lecture will cover the foundations, history, and key concepts of AI. We'll explore its evolution from early concepts to modern deep learning techniques.

by Hong-Han Shuai

National Yang Ming Chiao Tung University

https://tinyurl.com/NYCUIAI-2024

Syllabus

Week	Date	Contents
1	9/2	**Lecture 1: Class Overview and Unsupervised Learning** (HW#1) (HW#1)
2	9/9	Lecture 2: Traditional Classification-Part 1
3	9/16	**Lecture 3: Traditional Classification-Part 2**
4	9/23	Lecture 4: Neural Networks Basics (HW#2)
5	9/30	Hands-on Tutorials on PyTorch
6	10/7	Lecture 5: Deep Learning in Practice
7	10/14	Lecture 6: Introduction to Natural Language Processing
8	10/21	Midterm

Do not edit

How to change the design

Join at slido.com

#2829580

ⓘ The Slido app must be installed on every computer you’re presenting from

slido

你想到「分類」時，腦中第一個關鍵詞是什麼

Classification

Classification – Basic Concepts

Decision Tree Induction

Bayes Classification Methods

Rule-Based Classification

Techniques to Improve Classification Accuracy: Ensemble Methods

Lazy Learner

Support Vector Machine

Evaluations

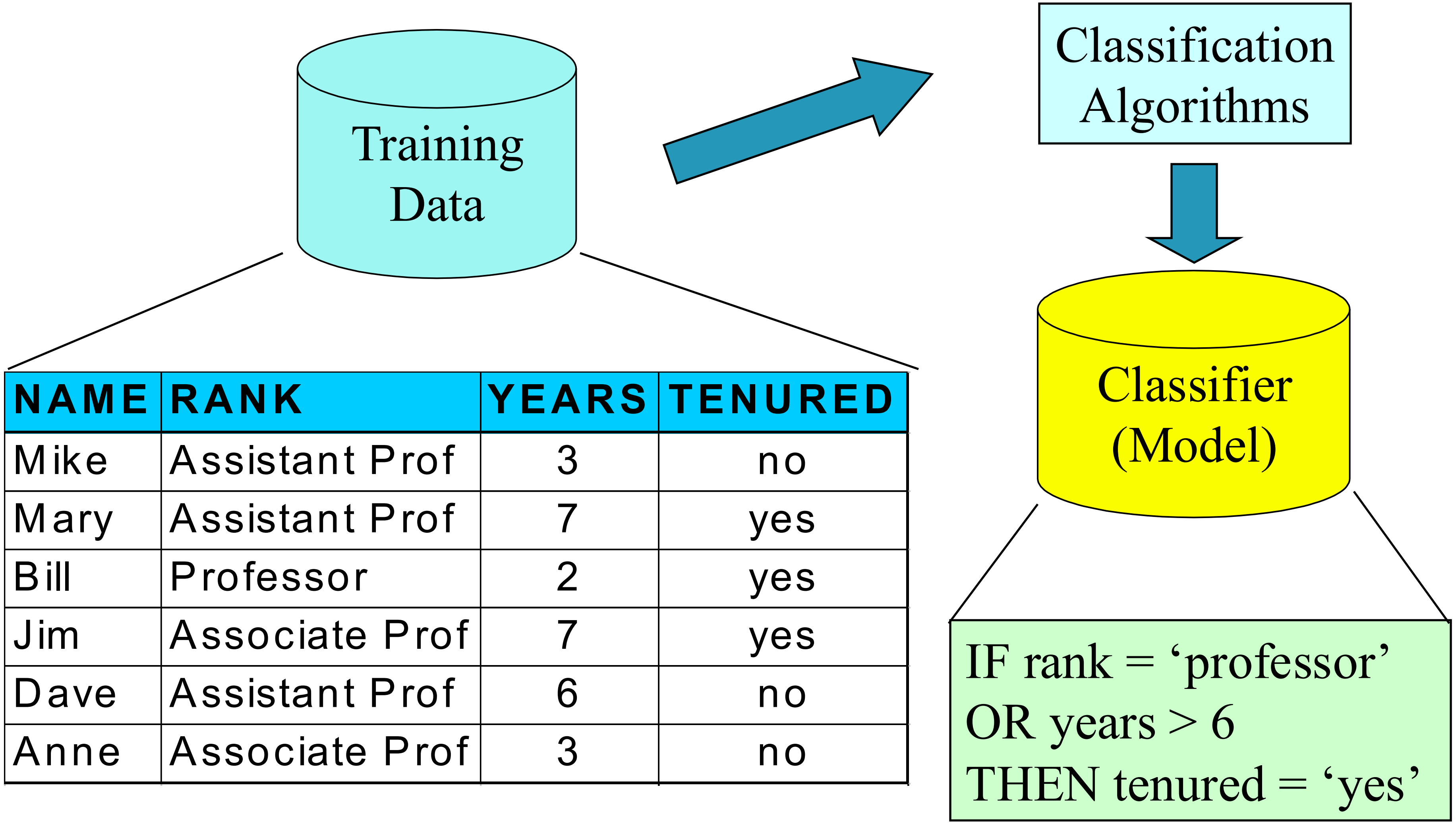
Prediction Problems: Classification vs. Numeric Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - 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
- Numeric Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit/loan approval: if an applicant is qualified
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - 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 is 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 overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- Note: If *the test set* is used to select models, it is called validation (test) set

Process (1): Model Construction



Training Data

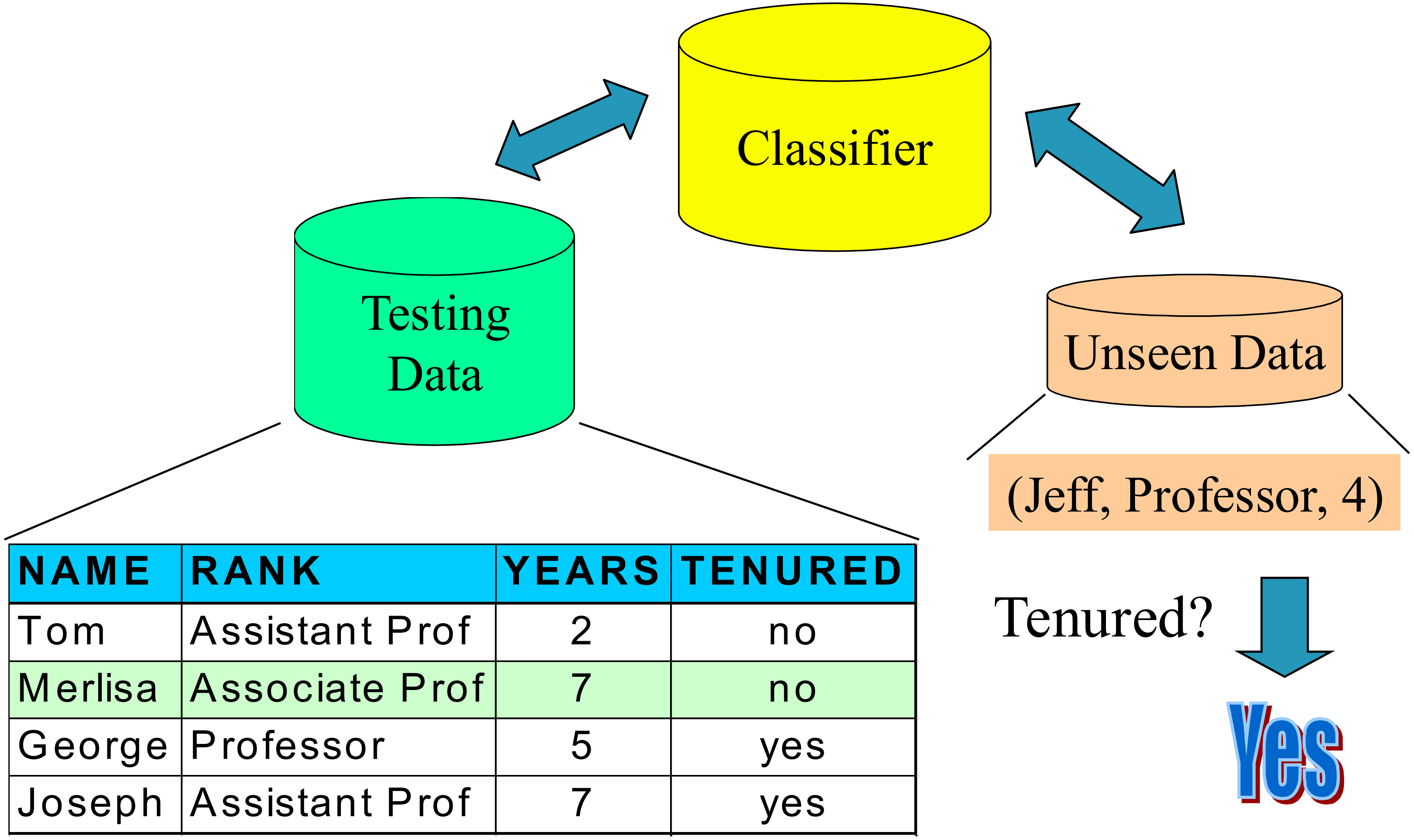
Classification Algorithms

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’

Process (2): Using the Model in Prediction



Classifier

Testing Data

Unseen Data

(Jeff, Professor, 4)

Tenured?

Yes

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

Classification

Classification – Basic Concepts

Decision Tree Induction

Bayes Classification Methods

Rule-Based Classification

Techniques to Improve Classification Accuracy: Ensemble Methods

Lazy Learner

Support Vector Machine

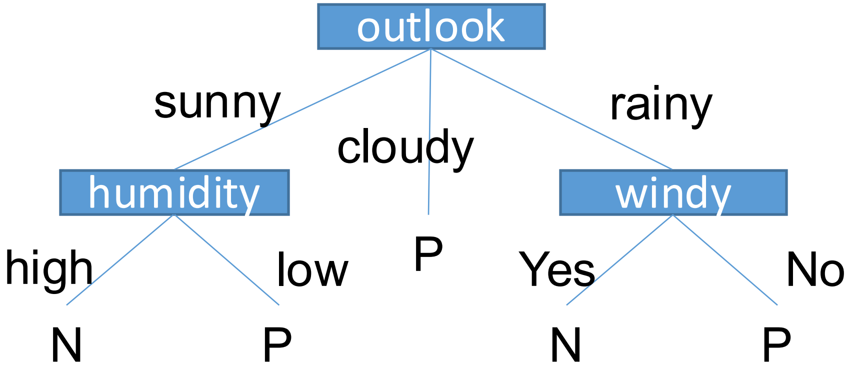
Evaluations

編號	價格 (Price)	地段 (Location)	是否靠近捷運 (MRT)	是否買房 (Buy?)
1	高 (High)	好 (Good)	是 (Yes)	否 (No)
2	高 (High)	普通 (Fair)	否 (No)	否 (No)
3	中 (Medium)	好 (Good)	是 (Yes)	是 (Yes)
4	低 (Low)	普通 (Fair)	是 (Yes)	是 (Yes)
5	低 (Low)	好 (Good)	否 (No)	是 (Yes)
6	中 (Medium)	普通 (Fair)	否 (No)	否 (No)

你覺得哪個屬性最能決定「買不買房」？

A Decision-Tree Based Classification

- A decision tree of whether going to play tennis or not:

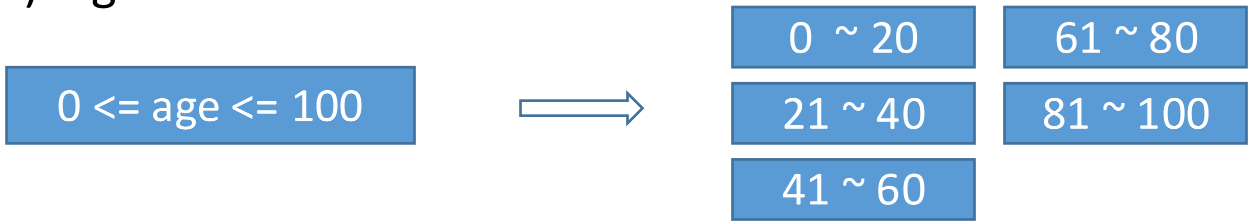


- **ID-3** and its extended version **C4.5** (Quinlan'93): A **top-down** decision tree generation algorithm

Algorithm for Decision Tree Induction

- Basic algorithm (a **greedy algorithm**)
  - Tree is constructed in a top-down recursive divide-and-conquer manner.
  - Attributes are categorical.

    (if an attribute is a continuous number, it needs to be discretized in advance.) E.g.



0 <= age <= 100

0 ~ 20

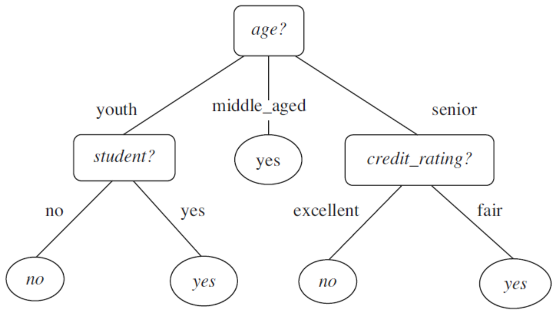
21 ~ 40

41 ~ 60

61 ~ 80

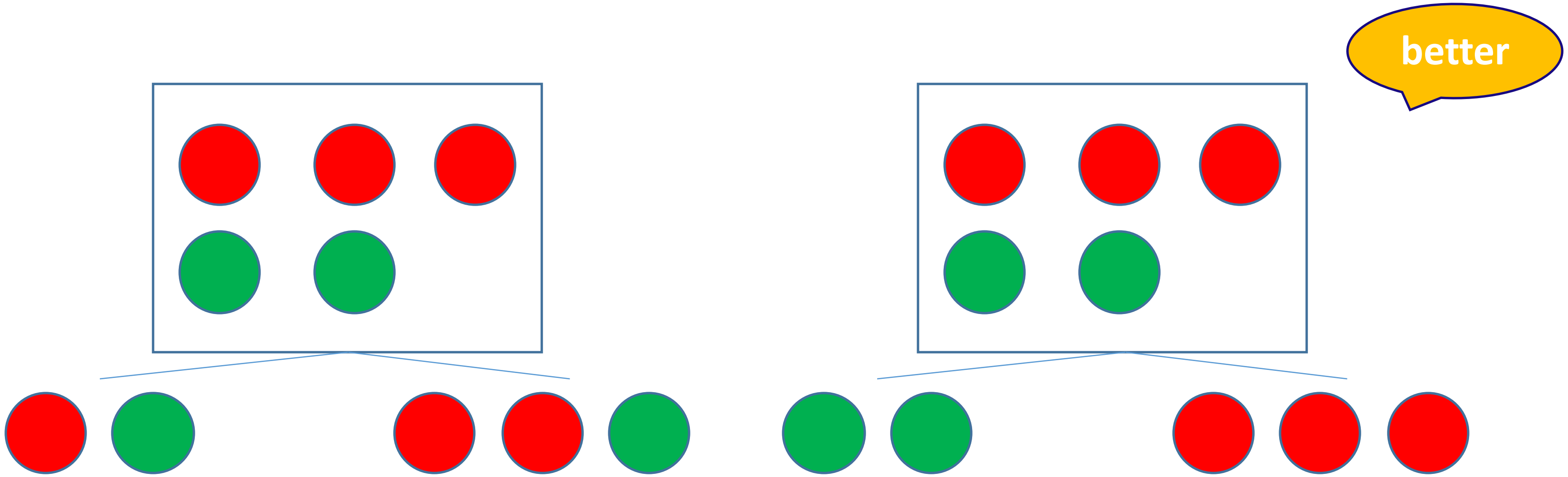
81 ~ 100

  - At start, all the training examples are at the root.
  - Examples are partitioned recursively based on selected attributes.



Algorithm for Decision Tree Induction

- Basic algorithm (a **greedy algorithm**)
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain): maximizing an information gain measure, i.e., **favoring the partitioning which makes the majority of examples belong to a single class**.



Algorithm for Decision Tree Induction

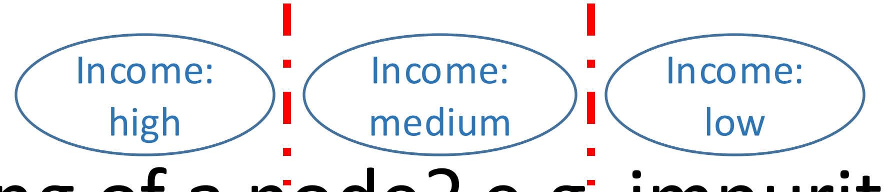
- Basic algorithm (a **greedy algorithm**)
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain): maximizing an information gain measure, i.e., **favoring the partitioning which makes the majority of examples belong to a single class**.
  - Conditions for stopping partitioning:
    - All samples for a given node belong to the same class
    - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
    - There are no samples left

Primary Issues in Tree Construction (1/2)

- **Split criterion**: *Goodness function*
  - Used to select the attribute to be split at a tree node during the tree generation phase
  - Different algorithms may use different goodness functions:
    - Information gain (used in ID3/C4.5)
    - Gain ratio
    - Gini index (used in CART)

Primary Issues in Tree Construction (2/2)

- **Branching scheme**:
  - Determining the tree branch to which a sample belongs
  - Binary vs. *k-ary* splitting



- When to stop the further splitting of a node? e.g. impurity measure
- Labeling rule: a node is labeled as the class to which most samples at the node belongs.

How to Use a Tree?

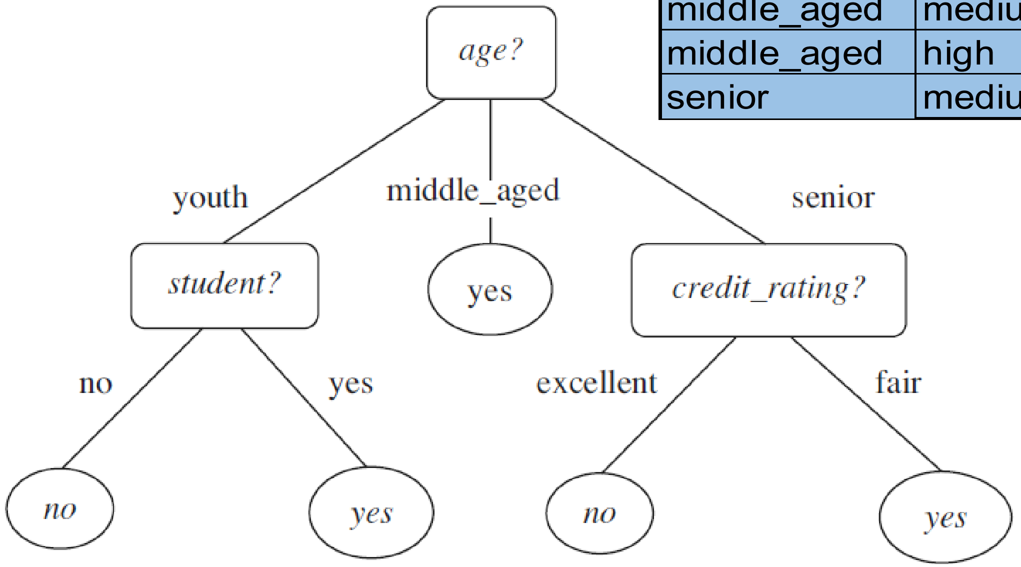
- Directly
  - Test the attribute value of unknown sample against the tree.
  - A path is traced from root to a leaf which holds the label.

- Indirectly
  - Decision tree is converted to classification rules.
  - One rule is created for each path from the root to a leaf.
  - **IF-THEN** is easier for humans to understand .

Decision Tree Induction: An Example

- Training data set: Buys_computer
- Resulting tree:

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

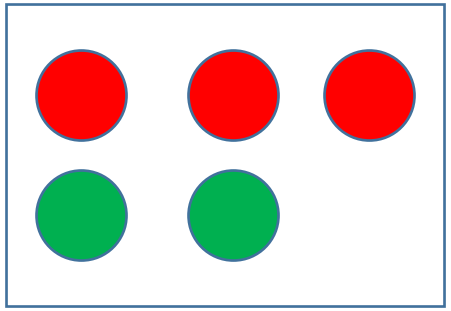


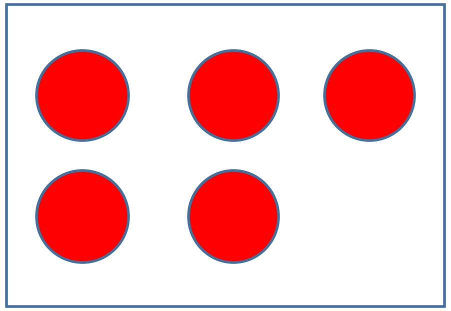
Expected Information (Entropy)

- Expected information (**entropy**) needed to classify a tuple in D:

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

(***p_i***: probability that a tuple in D belongs to class ***C_i***, ***m***: number of classes)





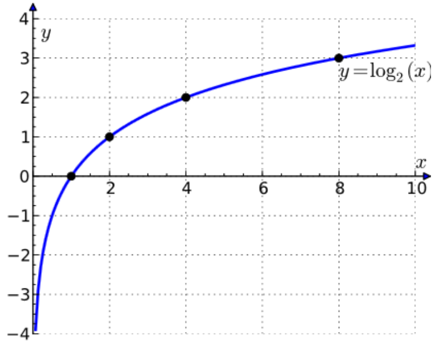
$$Info(D) = I(3,2) = -\frac{3}{5}\log_2(\frac{3}{5}) - \frac{2}{5}\log_2(\frac{2}{5})$$

$$\approx -\frac{3}{5}\times(-0.737) - \frac{2}{5}\times(-1.322)$$

$$\approx 0.971$$

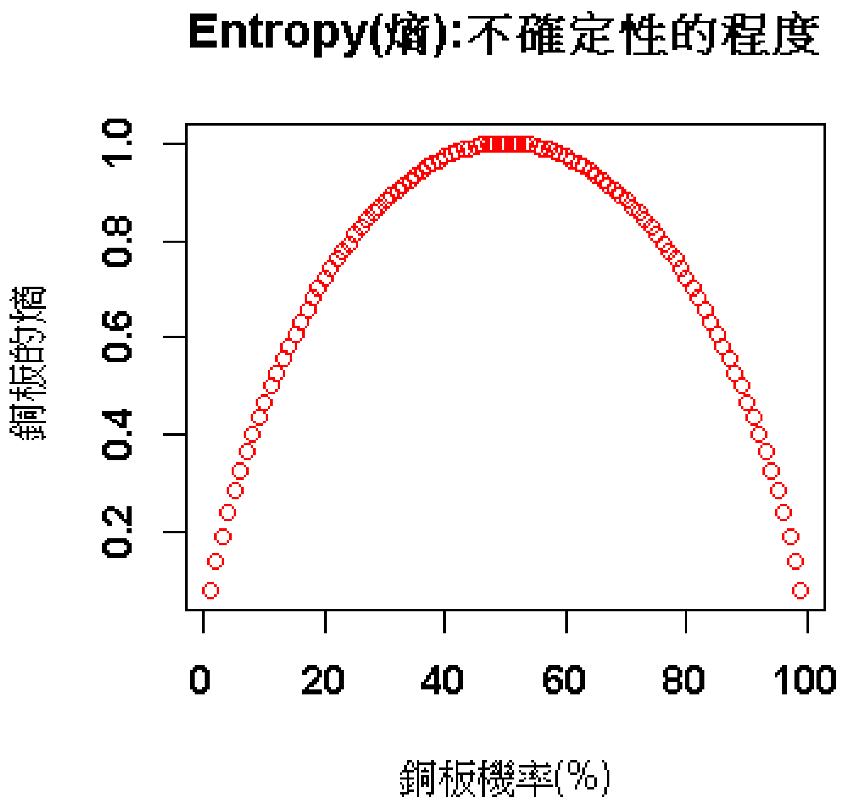
$$Info(D) = I(5,0) = -\frac{5}{5}\log_2(\frac{5}{5}) - \frac{0}{5}\log_2(\frac{0}{5})$$

$$= 0 - 0 = 0$$



Entropy?

$$H = -p*\log(p) - (1-p)*\log((1-p))$$



Entropy

- In short, logarithm is to make it growing linearly with system size and "behaving like information".

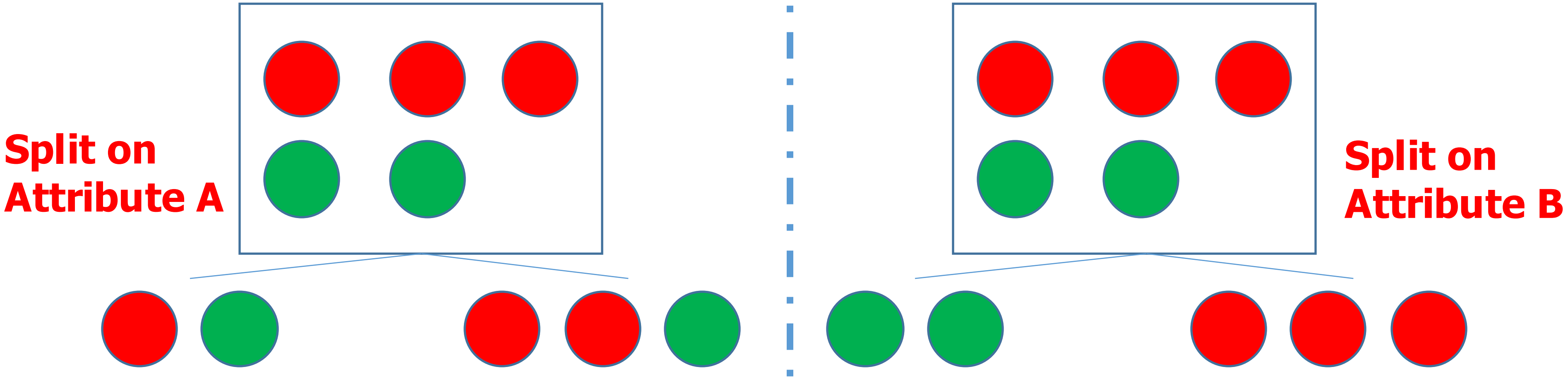
$$-\sum_{i=1}^{2^n}\frac{1}{2^n}\log\left(\frac{1}{2^n}\right) = -\sum_{i=1}^{2^n}\frac{1}{2^n}n\log\left(\frac{1}{2}\right) = n\left(-\sum_{i=1}^{2}\frac{1}{2}\log\left(\frac{1}{2}\right)\right) = n.$$

$$H = \sum_{i} p_i \log\left(\frac{1}{p_i}\right)$$

Expected Information (Entropy)

- Information needed (after using A to split D into v partitions) to classify D:

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



Split on Attribute A

Split on Attribute B

$$Info_A(D) = \frac{2}{5} Info(1,1) + \frac{3}{5} Info(2,1) = 0.918$$

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

$$= 0.971 - 0.918 = 0.053$$

$$Info_B(D) = \frac{2}{5} Info(2,0) + \frac{3}{5} Info(3,0) = 0$$

$$Gain(B) = Info(D) - Info_B(D)$$

$$= 0.971 - 0 = 0.971$$

Attribute Selection Measure: Information Gain (ID3)

- Select the attribute with the highest information gain
  - To minimize # of tests needed to classify a given tuple
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

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

- Information needed (after using A to split D into v partitions) to classify D:

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

- Information gained by branching on attribute A

We'd like to maximize ***Gain(A)***, i.e., to minimize ***Info$_A$(D)***, the information still required to finish classifying the tuples

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

Attribute Selection: Information Gain

- Class P: buys_computer = “yes”
- Class N: buys_computer = “no”

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

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

age	p_i	n_i	$I(p_i, n_i)$
youth	2	3	0.971
middle_aged	4	0	0
senior	3	2	0.971

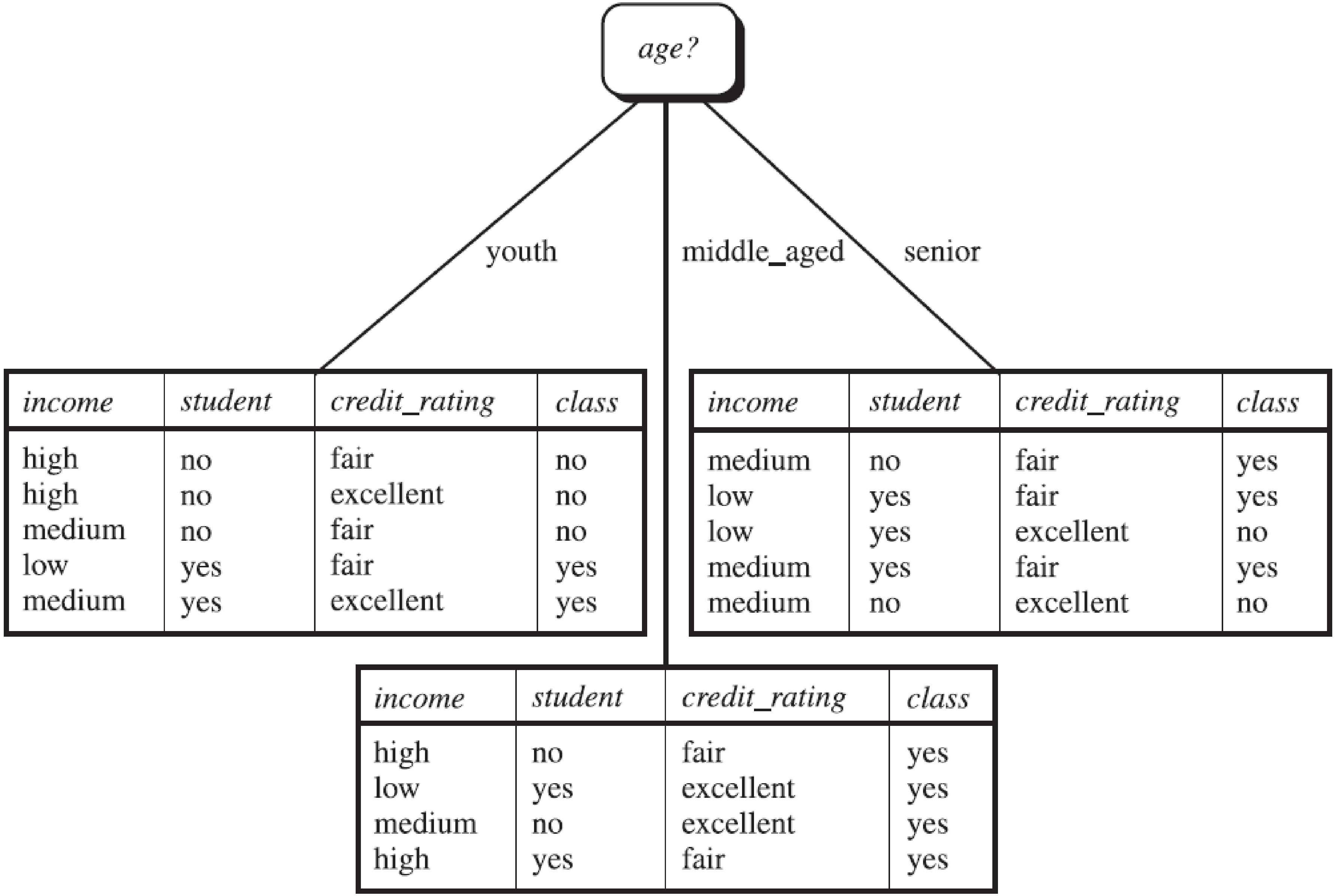
$\frac{5}{14}I(2,3)$ means “age = youth” has 5 out of 14 samples, with 2 yes’es and 3 no’s. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

Similarly,

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$



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

編號	價格 (Price)	地段 (Location)	是否靠近捷運 (MRT)	是否買房 (Buy?)
1	高 (High)	好 (Good)	是 (Yes)	否 (No)
2	高 (High)	普通 (Fair)	否 (No)	否 (No)
3	中 (Medium)	好 (Good)	是 (Yes)	是 (Yes)
4	低 (Low)	普通 (Fair)	是 (Yes)	是 (Yes)
5	低 (Low)	好 (Good)	否 (No)	是 (Yes)
6	中 (Medium)	普通 (Fair)	否 (No)	否 (No)

若用「價格 **(Price)**」作為第一個切分屬性，計算得到的資訊增益 **(Information Gain)** 是多少？ **(**四捨五入到小數點後兩位**)**

Computing Information-Gain for Continuous-Valued Attributes

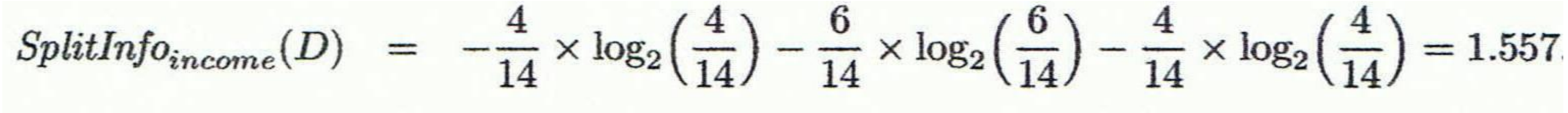
- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
  - E.g., unique pID -> split on pID results in large number of partitions, each containing just one tuple => each partition is pure
    => information required to classify this partition would be ***$Info_{pID}(D)=0$***,
    i.e., the information gain is maximal!!
- 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)

- E.g., 

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

  - gain_ratio(income) = 0.029/1.557 = 0.019
- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini Index (CART, IBM IntelligentMiner)

- Gini index considers a binary split for each attribute
  - To construct a binary decision tree
- Consider A: discrete-valued attribute
  - with v distinct values $\{a_1, a_2, \ldots, a_v\}$
- Examine all possible subsets that can be formed using values of A
  - Each subset S_A is considered as a binary test: "$\mathrm{A} \in S_A$"
- E.g., if income has three possible values {low, medium, high}
  - 2^3 possible subsets
  - ~~{low, medium, high}~~, {low, medium}, {low, high}, {medium, high}, {low}, {medium}, {high}, ~~{}~~
  - There are $(2^v - 2)/2$ possible ways to form two partition of the data D, based on a binary split A

Gini Index (CART, IBM IntelligentMiner)

- If a data set *D* contains examples from *m* classes, Gini index, *Gini*(*D*) is defined as

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

where p_i is the probability of class C_i in *D*, estimated by $|C_{i,D}|/|D|$

- If a data set *D* is split on A into two subsets D_1 and D_2, the *Gini* index $Gini_A(D)$ given the split on A is:

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

- Reduction in Impurity:

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

- The attribute provides the largest reduction in impurity, i.e., maximized $\Delta Gini(A)$, is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

Computation of Gini Index

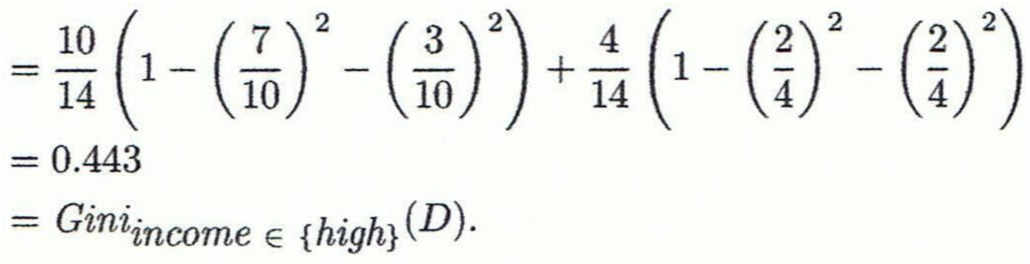
- E.g., D has **9** tuples in **buys_computer = "yes"** and **5** in **"no"**

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute ***income*** partitions D into **10** in **D$_1$**: **{low, medium}** and **4** in **D$_2$**

$$Gini_{income \in \{low,medium\}}(D)$$

$$= \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$



$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443$$

$$= Gini_{income\ \in\ \{high\}}(D).$$

Gini$_{\{low,high\}}$ is ***0.458***; ***Gini$_{\{medium,high\}}$*** is ***0.450***. Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - **Information gain**:
    - biased towards multivalued attributes
  - **Gain ratio**:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - **Gini index**:
    - biased to multivalued attributes
    - has difficulty when # of classes is large (high computation overhead)
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistic: has a close approximation to χ^2 distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

Pre - Pruning

- **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
  - Use all available data for training and apply the statistical test to estimate whether expanding/pruning a node is to produce an improvement beyond the training set
- **Most popular test:** chi-squared test
- $\text{chi}^2 = \text{sum}(\ (O\text{-}E)^2\ /\ E\ )$

Where, O = observed data, E = expected values based on hypothesis.

Example

- Example : 5 schools have the same test. Total score is 375, individual results are: 50, 93, 67, 78 and 87. Is this distribution significant, or was it just luck? Average is 75.

  $(50\text{-}75)^2/75 + (93\text{-}75)^2/75 + (67\text{-}75)^2/75 + (78\text{-}75)^2/75 + (87\text{-}75)^2/75 = 15.55$

  This distribution is significant !

OK?

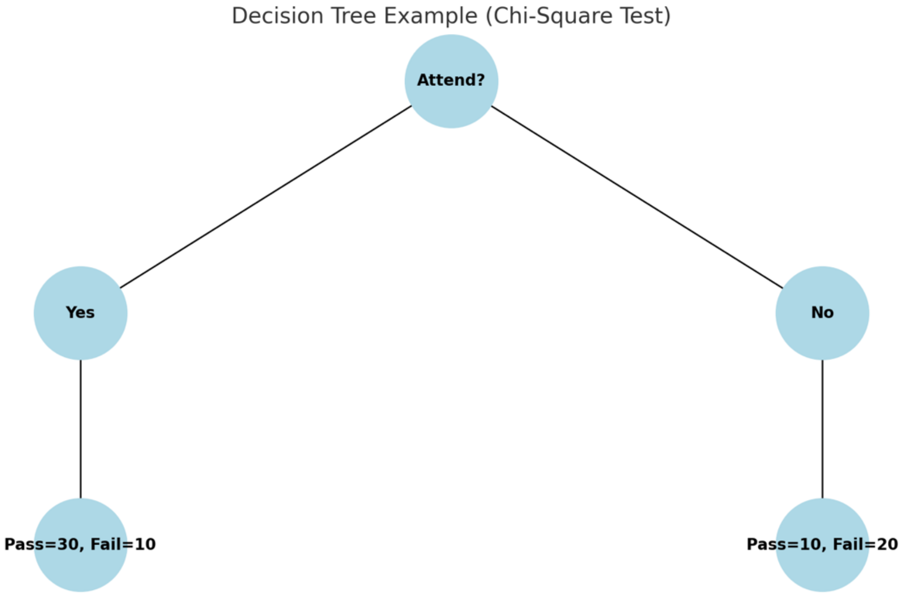
統計顯著？

假設我們要用 decision tree 預測「學生是否考試及格 (Pass/Fail)」，屬性有一個：「是否有上課 (Attend = Yes/No)」。

Attend	Pass	Fail	Total
Yes	30	10	40
No	10	20	30
Total	40	30	70

Step 1: 建立假設

- ➢H_0:「是否上課」和「是否及格」無關。
- ➢H_1:「是否上課」和「是否及格」有關。



Step 2: 計算期望值 E

期望值 = (列合計 × 行合計) / 總合計

- Attend=Yes, Pass: 40×(40/70) = 22.86
- Attend=Yes, Fail: 40×(30/70)= 17.14
- Attend=No, Pass: 30×(40/70) = 17.14
- Attend=No, Fail: 30×(30/70)= 12.86

Step 3: 計算 χ^2

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

$(30-22.86)^2 / 22.86 = 2.23$

$(10-17.14)^2 / 17.14 = 2.97$

$(10-17.14)^2 / 17.14 = 2.97$

$(20-12.86)^2 / 12.86 = 3.96$

- 合計 **χ^2 = 12.13**

Step 4: 查自由度 df

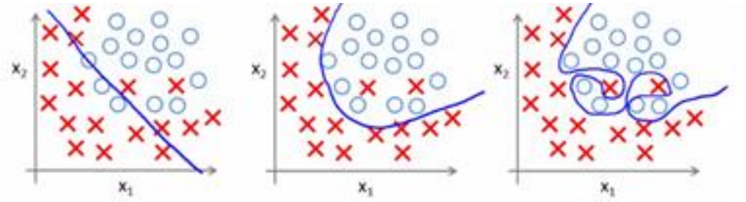
- df = (行數−1) × (列數−1) = (2−1)×(2−1) = 1

Step 5: 判斷顯著性

$\chi^2 = 12.12$, df = 1
查表或用電腦算 → p-value ≈ 0.0005
因為 **p < 0.05** → 顯著差異 → 「是否上課」跟「是否及格」有關 → 可以用這個屬性來分裂 decision tree。

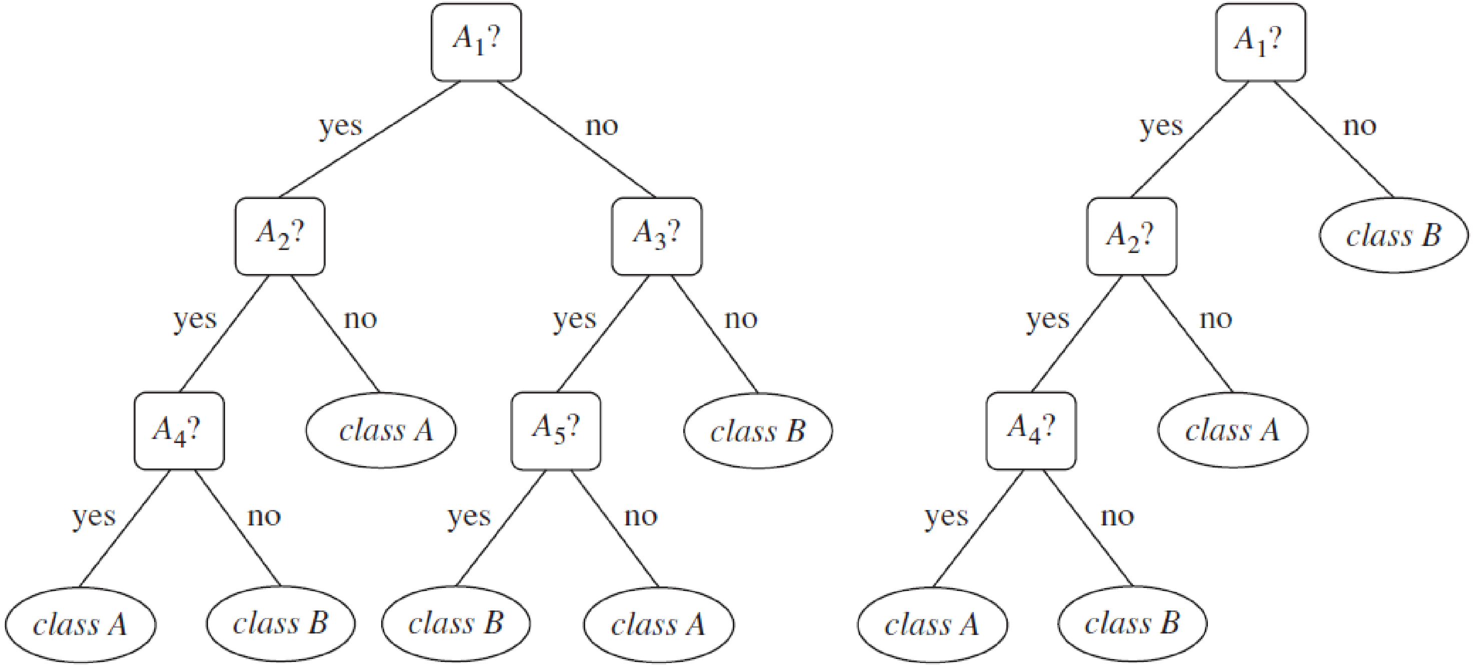
df=1	α=0.05	α=0.01	α=0.001
χ^2	3.84	6.63	10.83

Overfitting and Tree Pruning



underfitting overfitting

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples, i.e., lose the ability of generalization



Overfitting and Tree Pruning

- 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"

Enhancements to Basic Decision Tree Induction

- Allow for **continuous-valued attributes**
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle **missing attribute values**
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- **Attribute construction**
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

Classification

Classification – Basic Concepts

Decision Tree Induction

Bayes Classification Methods

Rule-Based Classification

Techniques to Improve Classification Accuracy: Ensemble Methods

Lazy Learner

Support Vector Machine

Evaluations

Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayes' Theorem: Basics

- Total probability Theorem: $P(B) = \sum_{i=1}^{M} P(B|A_i)P(A_i)$

- Bayes' Theorem: $P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$

  - Let **X** be a data sample ("*evidence*"): class label is unknown
  - Let H be a *hypothesis* that X belongs to class C
  - Classification is to determine P(H|**X**), (i.e., *posteriori probability):* the probability that the hypothesis holds given the observed data sample **X**
  - P(H) (*prior probability*): the initial probability
    - E.g., **X** will buy computer, regardless of age, income, …
  - P(**X**): probability that sample data is observed
  - P(**X**|H) (likelihood): the probability of observing the sample **X**, given that the hypothesis holds
    - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Summary

後驗 (posterior) = 在看到資料後，假設成立的可能性

似然 (likelihood) = 如果假設成立，看到這些資料的機率

先驗 (prior) = 在看資料前，你覺得這假設有多可能

證據 (evidence) = 所有假設都考慮後，這些資料出現的總機率

Example:「今天下雨」這個假設 H。

P(H)：台北九月平均有 30% 的日子會下雨。
P(X|H)：如果今天真的下雨，那麼「我帶雨傘」的機率是 90%。
P(X)：不管有沒有下雨，隨機看人帶雨傘的機率是 40%。

P(H/X)=P(X|H)P(H)/P(X)
=0.9*0.3/0.4=0.675

Prediction Based on Bayes' Theorem

- Given training data **X**, *posteriori probability of a hypothesis* H, P(H|**X**), follows the Bayes' theorem

$$P(H\,|\,\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

- Informally, this can be viewed as

  posteriori = likelihood x prior/evidence

- Predicts **X** belongs to C_i iff the probability P(C_i|**X**) is the highest among all the P(C_k|X) for all the *k* classes
- Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost

Classification is to Derive the Maximum Posteriori

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector **X** = (x_1, x_2, …, x_n)
- Suppose there are *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal P(C_i|**X**)
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(**X**) is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution
- If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i, D}|$ (# of tuples of C_i in D)
- If A_k is continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

and $P(x_k|C_i)$ is

$$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Naïve Bayes Classifier: Training Dataset

Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

Data to be classified:

X = (age = youth,

income = medium,

student = yes

credit_rating = Fair)

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

We need to maximize $P(X|C_i)P(C_i)$, for $i = 1, 2$

Naïve Bayes Classifier: An Example

- The prior probability of each class:
  $P(C_i)$: P(buys_computer = “yes”) = 9/14 = 0.643

  P(buys_computer = “no”) = 5/14= 0.357
- Compute **$P(X|C_i)$** for each class

  P(age = youth | buys_computer = “yes”) = 2/9 = 0.222
  P(age = youth | buys_computer = “no”) = 3/5 = 0.6
  P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444
  P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4
  P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667
  P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2
  P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667
  P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4
- **X** = (age = youth, income = medium, student = yes, credit_rating = fair)

$P(X|C_i)$: P(X|buys_computer = “yes”) = 0.222 x 0.444 x 0.667 x 0.667 = **0.044**

P(X|buys_computer = “no”) = 0.6 x 0.4 x 0.2 x 0.4 = **0.019**

$P(X|C_i)*P(C_i)$: P(**X**|buys_computer = “yes”) * P(buys_computer = “yes”) = 0.028

P(**X**|buys_computer = “no”) * P(buys_computer = “no”) = 0.007

Therefore, X belongs to class (“buys_computer = yes”)

Avoiding the Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be **non-zero**. Otherwise, the predicted prob. will be zero

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$

- E.g., Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)
- Use **Laplacian correction** (or Laplacian estimator)
  - *Adding* ***1*** *to each case*

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - The “corrected” prob. estimates are close to their “uncorrected” counterparts

Naïve Bayes Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals, patients: Profile: age, family history, etc.
      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier

Classification

Classification – Basic Concepts

Decision Tree Induction

Bayes Classification Methods

Rule-Based Classification

Techniques to Improve Classification Accuracy: Ensemble Methods

Lazy Learner

Support Vector Machine

Evaluations

Using IF-THEN Rules for Classification

- Represent the knowledge in the form of IF-THEN rules

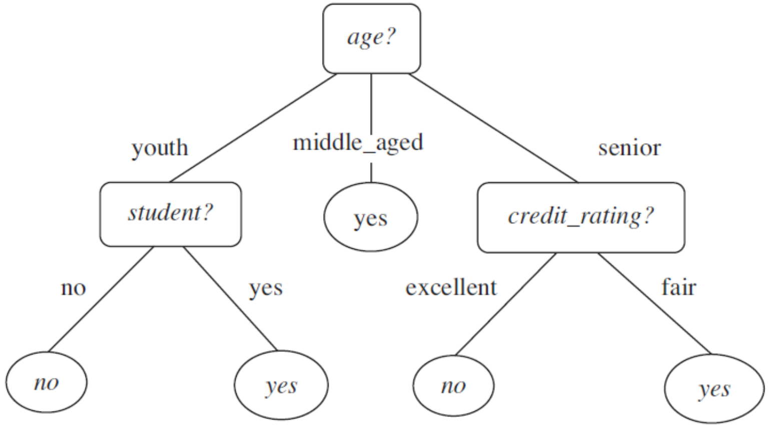
  R: **IF** *age* = youth **AND** *student* = yes **THEN** *buys_computer* = yes

  - Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: *coverage* and *accuracy*
  - n_{covers} = # of tuples covered by R
  - $n_{correct}$ = # of tuples correctly classified by R

  coverage(R) = $\mathbf{n_{covers}} \,/\, \mathbf{|D|}$ /* D: training data set */

  accuracy(R) = $\mathbf{n_{correct}} \,/\, \mathbf{n_{covers}}$
- If more than one rule are triggered, need **conflict resolution**
  - Size ordering: assign the highest priority to the triggering rules that has the "toughest" requirement (i.e., with the ***most attribute tests***)
  - Class-based ordering: decreasing order of *prevalence or misclassification cost per class*
  - Rule-based ordering (**decision list**): rules are organized into one long priority list, according to some measure of rule quality or by experts

Rule Extraction from a Decision Tree



- Rules are *easier to understand* than large trees
- One rule is created *for each path* from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive

No rule conflicts

One rule for each attribute-value combination

- Example: Rule extraction from our *buys_computer* decision-tree

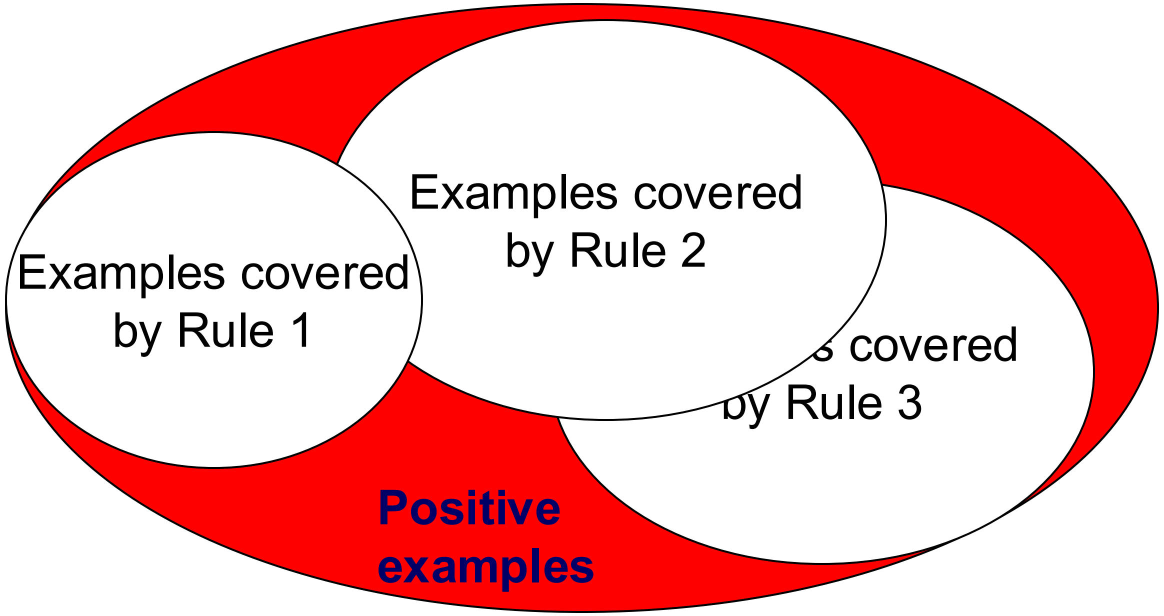
IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*
IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*
IF *age* = mid-age THEN *buys_computer* = *yes*
IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *no*
IF *age* = old AND *credit_rating* = *fair* THEN *buys_computer* = *yes*

Rule Induction: Sequential Covering Method

- Sequential covering algorithm =>
  Extracts rules directly from training data
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned *sequentially*, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Steps:
  - Rules are learned one at a time
    (i.e., procedure *Learn-One-Rule*, detailed later)
  - Each time a rule is learned, the tuples covered by the rules are removed
  - Repeat the process on the remaining tuples until *termination condition*, e.g., when no more training examples or when the quality of a rule returned is below a user-specified threshold
- Comp. w. decision-tree induction:
  Decision-Trees learn a set of rules *simultaneously*

Sequential Covering Algorithm

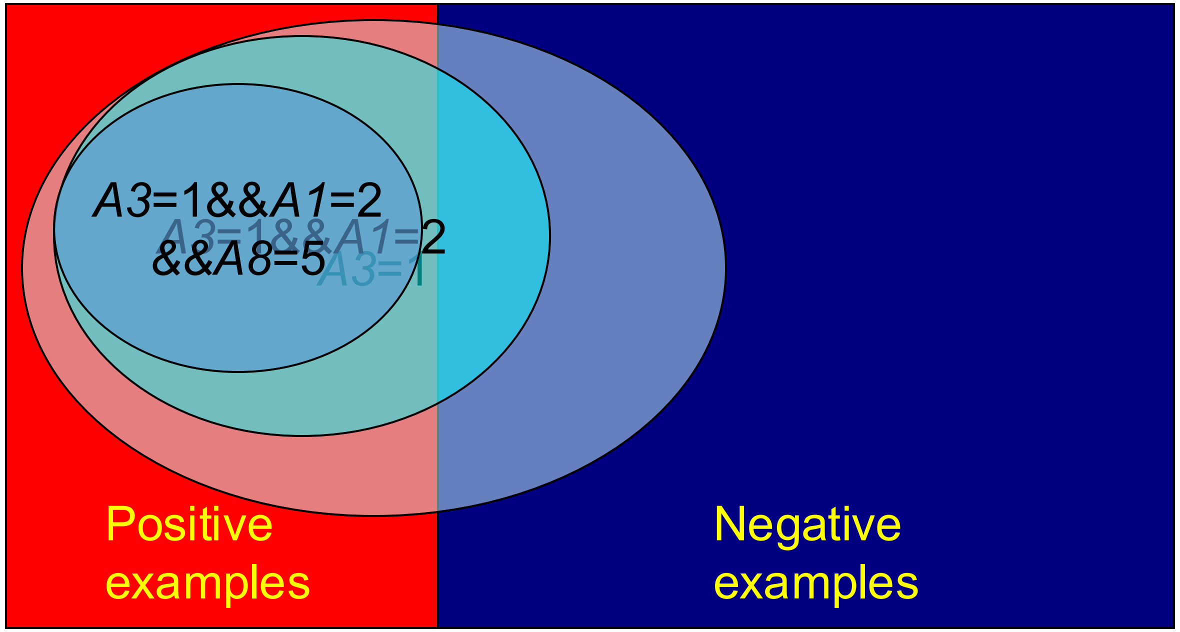
```
while (enough target tuples left)
    generate a rule
    remove positive target tuples satisfying this rule
```



Rule Generation

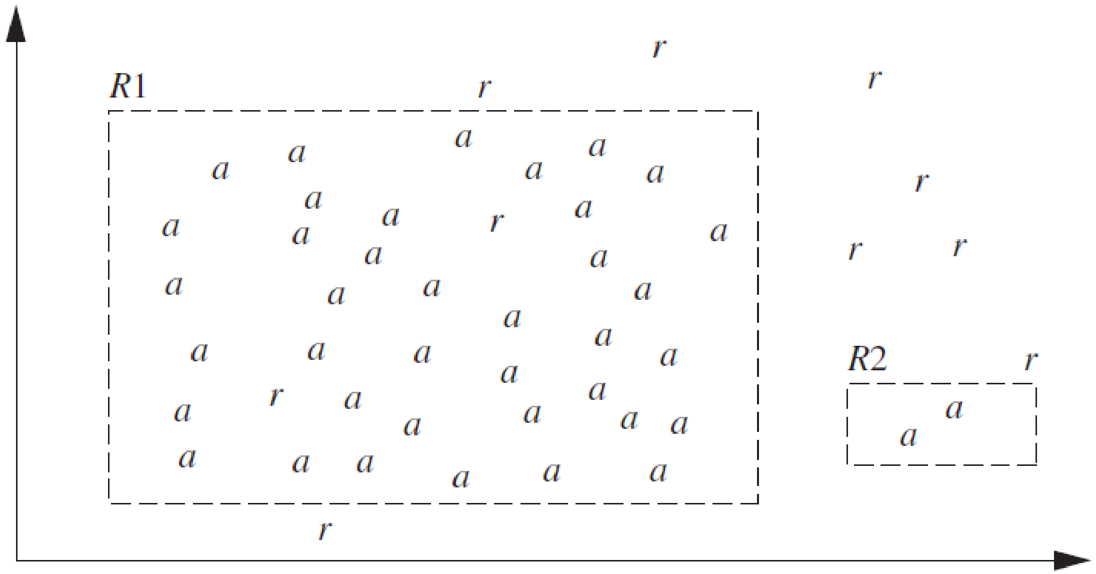
- To generate a rule

```
while(true)
  find the best predicate p
  if foil-gain(p) > threshold then add p to current rule
  else break
```



How to *Learn-One-Rule*?

- Start with the *most general rule* possible: condition = empty
- *Adding new attributes* by adopting a greedy depth-first strategy
    - Picks the one that most improves the rule quality
- Rule-Quality measures: consider both coverage and accuracy



Rule R1 has accuracy 95%, while R2 has 100% accuracy.
However, R2 is not better because of its poor coverage

How to *Learn-One-Rule*?

- Rule-Quality measures: consider both coverage and accuracy
  - Foil-gain (in FOIL & RIPPER): assesses info_gain by extending condition
    (pos’ and neg’ are the # of positive, negative tuples covered by our new rule R’, respectively)

$$FOIL\_Gain = pos' \times (\log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg})$$

  - favors rules that have high accuracy and cover many positive tuples

Classification

Classification – Basic Concepts

Decision Tree Induction

Bayes Classification Methods

Rule-Based Classification

Techniques to Improve Classification Accuracy: Ensemble Methods

Lazy Learner

Support Vector Machine

Evaluations

Motivations: Pikachu Recognition Problem

- Is this a picture of a pikachu?
- Teacher: Please look at the pictures of Pikachu. Based on those pictures, how would you describe Pikachu? Ted?
- Ted: Yellow creature.

http://youtubepoop.wikia.com/wiki/Pikachu

However,

http://t17.techbang.com/topics/36265-did-you-find-these-little-details-the-little-soldier-9-cold-knowledge

http://blog.xuite.net/w260905/blog/204405165-%E8%80%81%E7%9A%AE

Motivations: Pikachu Recognition Problem

- Teacher: As you can see, if you only use colors, you could make several mistakes. What else can we say for Pikachu? Christina?
- Christina: Shape.

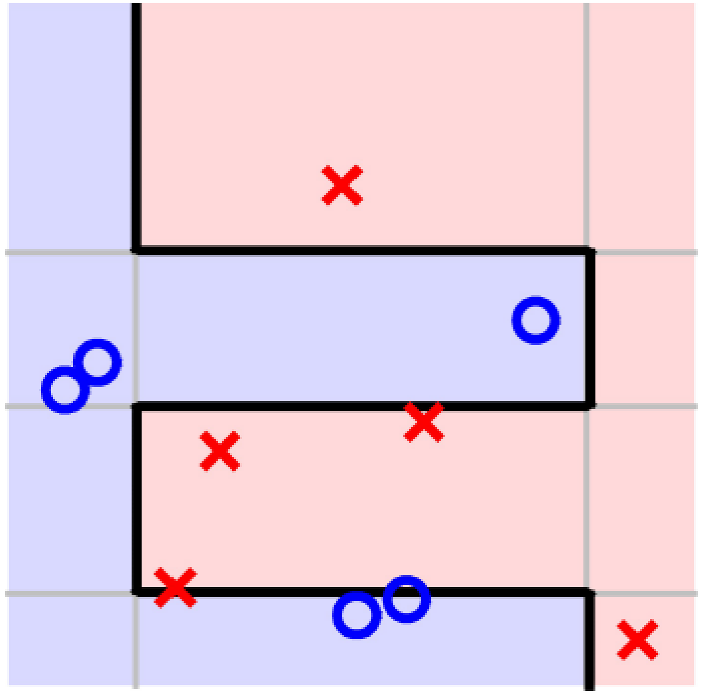
http://youtubepoop.wikia.com/wiki/Pikachu

?
POKÉMON

However,

Motivation

- students: simple hypotheses g_t (like vertical/horizontal lines)
- (Class): sophisticated hypothesis G (like black curve)
- Teacher: a tactic learning algorithm that **directs the students to focus on key examples**



Goal of Supervised Learning?

- Minimize the probability of model prediction errors on ***future*** data

- Two Competing Methodologies
  - Build **one** really good model
    - Traditional approach
  - Build **many** models and average the results
    - Ensemble learning (more recent)

The Single Model Philosophy

- Motivation: Occam’s Razor
  - “one should not increase, beyond what is necessary, the number of entities required to explain anything”
- Infinitely many models can explain any given dataset
  - Might as well pick the smallest one…

Which Model is Smaller?

$$\hat{y} = f_1(x) = \sin(x)$$

$$or$$

$$\hat{y} = f_2(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

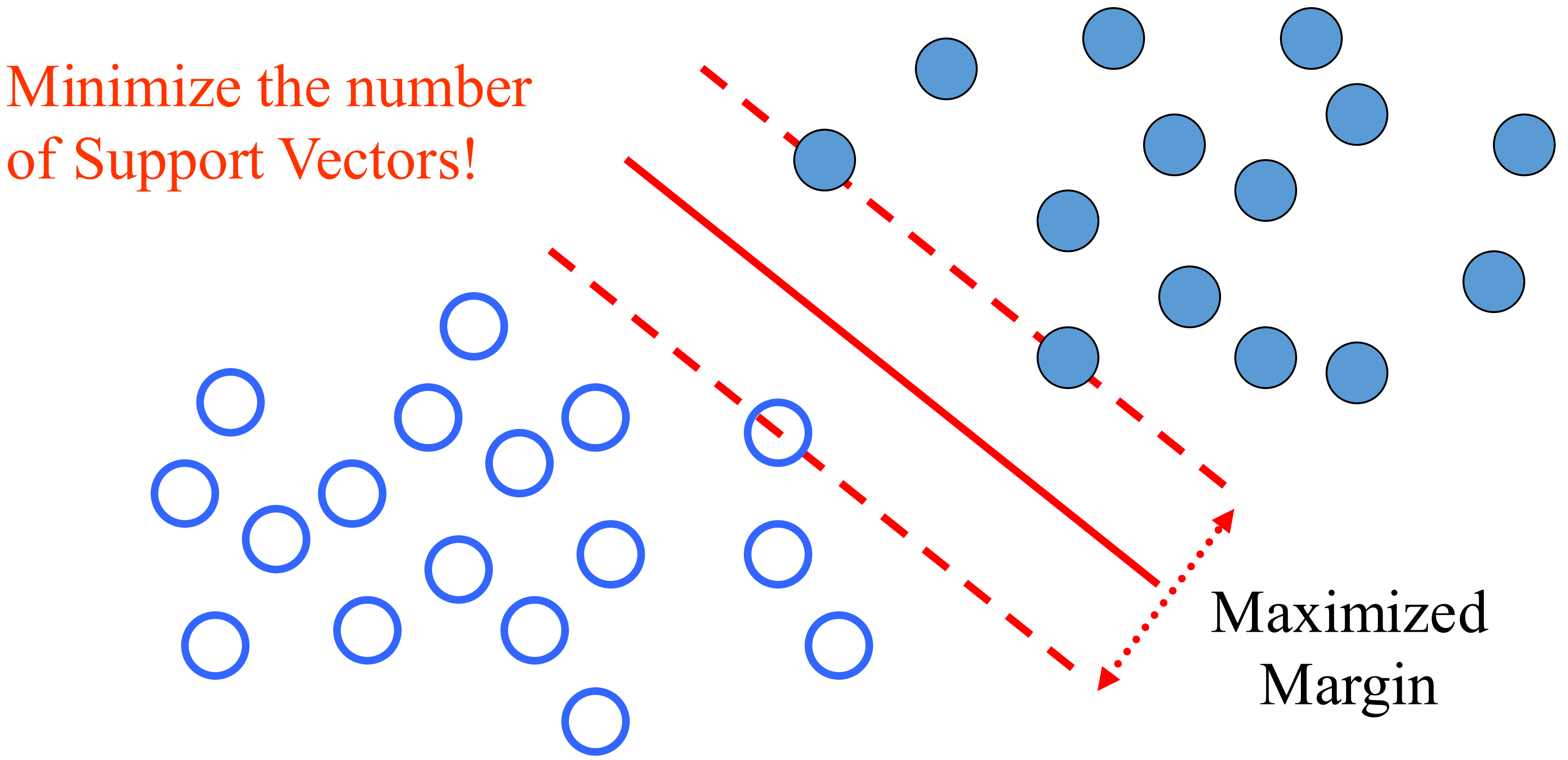
- In this case $f_1(x) \equiv f_2(x)$

- It's not always easy to define small!

Exact Occam’s Razor Models

- Exact approaches find optimal solutions
- Examples:
  - Support Vector Machines
    - Find a model structure that uses the smallest percentage of training data (to explain the rest of it).
  - Bayesian approaches
    - Minimum description length

How Do Support Vector Machines Define Small?



Approximate Occam’s Razor Models

- Approximate solutions use a greedy search approach which is not optimal
- Examples
  - Kernel Projection Pursuit algorithms
    - Find a minimal set of kernel projections
  - Relevance Vector Machines
    - Approximate Bayesian approach
  - Sparse Minimax Probability Machine Classification
    - Find a minimum set of kernels and features

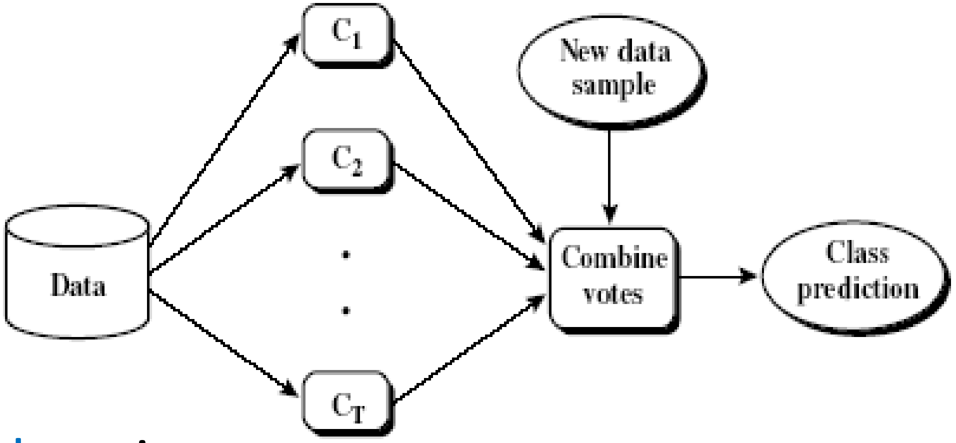
Other Single Models: Not Necessarily Motivated by Occam’s Razor

- Minimax Probability Machine (MPM)
- Trees
  - Greedy approach to sparseness
- Neural Networks
- Nearest Neighbor
- Basis Function Models
  - e.g. Kernel Ridge Regression

Ensemble Philosophy

- Build many models and combine them
- Only through averaging do we get at the truth!
- It’s too difficult (*impossible*?) to build a single model that works best
- Two types of approaches:
  - Models that don’t use randomness
  - Models that incorporate randomness

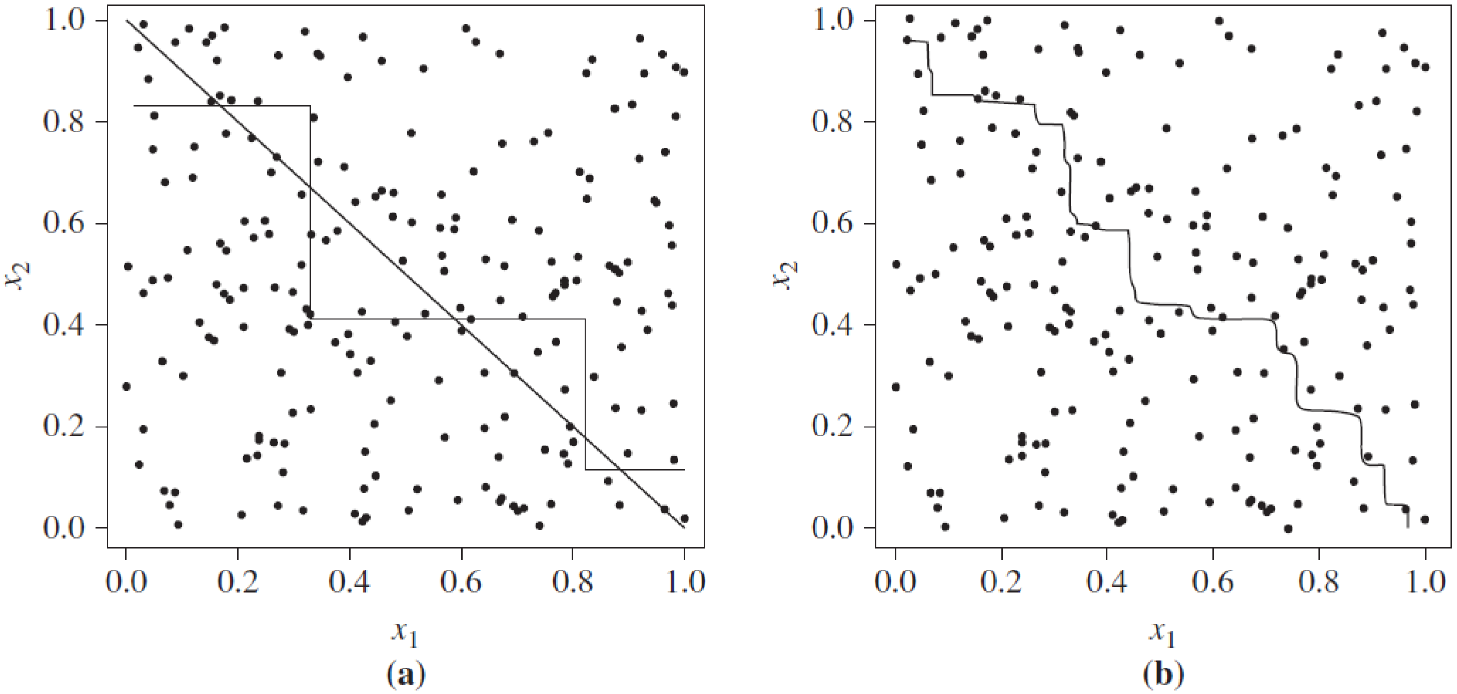
Ensemble Methods: Increasing the Accuracy



- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*
  - Ideally, significant diversity (little correlation) among the models
- Popular ensemble methods
  - **Bagging:** averaging the prediction over a collection of classifiers
  - **Boosting:** weighted vote with a collection of classifiers
  - **Ensemble:** combining a set of heterogeneous classifiers

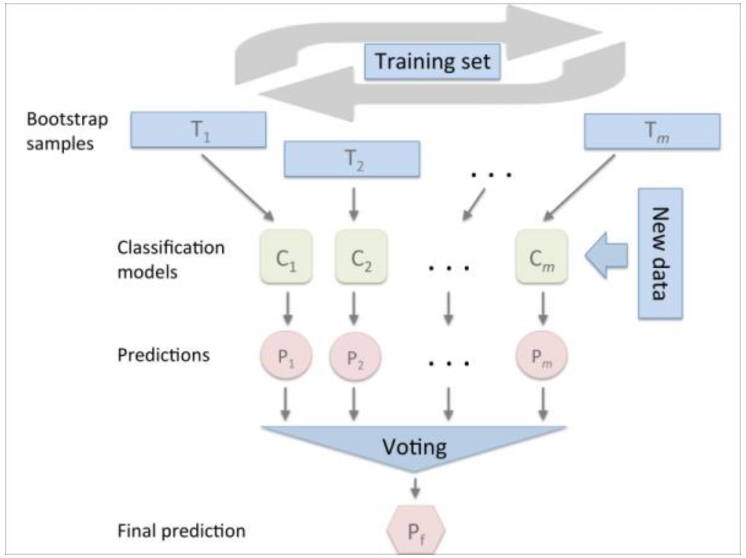
Ensemble methods

- For a two-class problem with attributes x_1, x_2



Decision boundary by (a) a single decision tree and (b) an ensemble of decision trees for a linearly separable problem (i.e., where the actual decision boundary is a straight line). The decision tree struggles with approximating a linear boundary. The decision boundary of the ensemble is closer to the true boundary. *Source:* From Seni and Elder [SE10]. © 2010 Morgan & Claypool Publishers; used with permission.

Bagging: Boostrap Aggregation



- Analogy: Diagnosis based on multiple doctors' **majority vote**
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., **bootstrap**)
  - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significantly better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - **Weights** are assigned to each training tuple
  - A series of k classifiers is iteratively learned
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to **pay more attention to the training tuples that were misclassified** by M_i
  - The final **M* combines the votes** of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- Boosting algorithm can be extended for numeric prediction
- Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

Let’s calculate the math!!

- Suppose that we have 5 models with an avg. accuracy of 0.6.

0.6

0.6

0.6

0.6

0.6

三個臭 裨將 勝過一個諸葛亮?

至少3個模型會猜對的機率?

$$\begin{aligned} 1-(0.4)^5-C_1^5(0.4)^4(0.6)-C_2^5(0.4)^3(0.6)^2 &= 1-0.01024-0.0768-0.2304 \\ &= 0.68256 \end{aligned}$$

Adaboost (Freund and Schapire, 1997)

- Given *d* class-labeled tuples (y_i is class label): ($\mathbf{X_1}$, y_1), …, ($\mathbf{X_d}$, y_d)
- Initially, all the weights of tuples are set the same (1/d)
- Generate k classifiers in k rounds. At round i,
    - Tuples from D are sampled (with replacement) to form a training set D_i of the same size
    - Each tuple's chance of being selected is based on its weight
    - A classification model M_i is derived from D_i
    - Its error rate is calculated using D_i as a test set
    - If a tuple is misclassified, its weight is increased, o.w. it is decreased
- Error rate: err($\mathbf{X_j}$) is the misclassification error of tuple $\mathbf{X_j}$. Classifier M_i error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_{j}^{d} w_j \times err(\mathbf{X_j})$$

Adaboost (Freund and Schapire, 1997)

Weight update

- If tuple in round *i* was correctly classified, its weight is multiplied by

$$\frac{error(M_i)}{1 - error(M_i)}$$

where M_i is the classification model derived from round *i.*

- The weights of all tuples will be normalized, which can be viewed as increasing the weights of misclassifies tuples

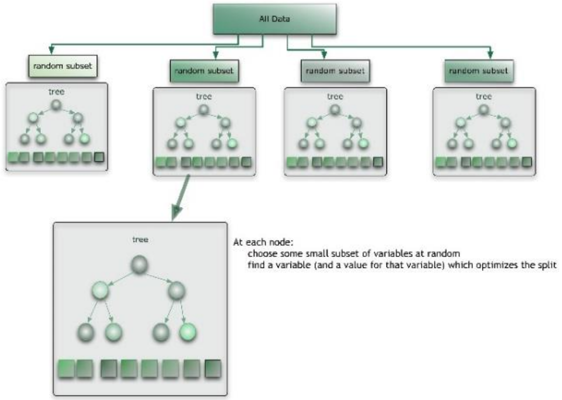
Prediction

- Adaboost assigns a weight to each classifier's M_i's vote
- The weight of classifier M_i's vote is:

$$\log \frac{1 - error(M_i)}{error(M_i)}$$

- The class with highest sum is returned as the prediction

Random Forest



At each node:
choose some small subset of variables at random
find a variable (and a value for that variable) which optimizes the split

- Random Forest:
  - Each classifier in the ensemble is a ***decision tree*** classifier and is generated using a random selection of attributes at each node to determine the split
  - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*)*:* Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

Beyond the Buzz

- Quotes from http://www.kdnuggets.com/2017/08/deep-learning-not-ai-future.html
- *"Too many startups and products are named "deep-something", just as buzzword: **very few** are using DL really"*
- *"**Decision Trees** like XGBoost are not making headlines, but silently beat DL at many Kaggle tabular data competitions."*
- *"**Legal nightmare** of DL decisions, that even if correct, can't be **explained** when legally questioned"*
- ***"**For many tasks, Deep Learning AI is or will become **illegal**, not compliant"*
  - *"**GDPR**, about automated decision-making, requires the right to an explanation, and to prevent discriminatory effects based on race, opinions, health, etc."*
  - *"Fines for noncompliance are **4% of global revenue**."*

Classification of Class-Imbalanced Data Sets

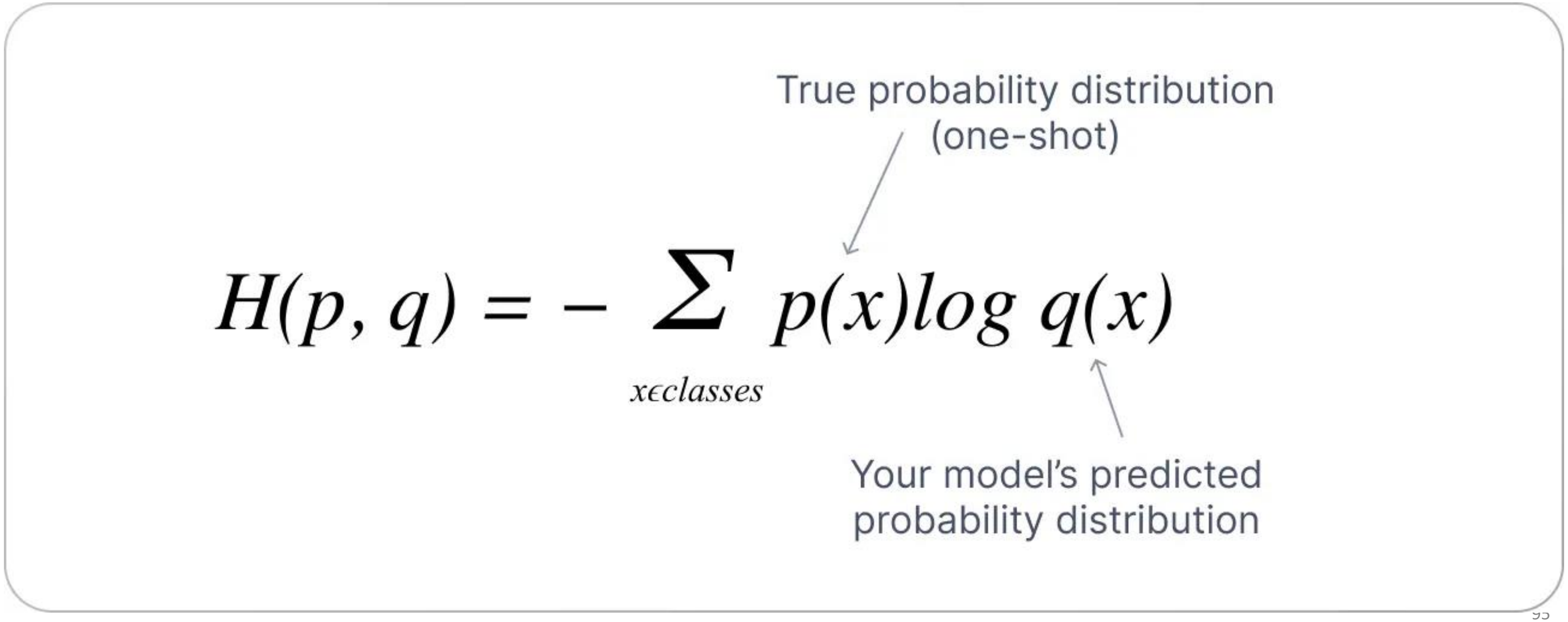
- Class-imbalance problem: Rare positive example but numerous negative ones, e.g., medical diagnosis, fraud, oil-spill, fault, etc.
- Traditional methods assume a balanced distribution of classes and equal error costs: not suitable for class-imbalanced data
- Typical methods for imbalance data in 2-class classification:
  - **Oversampling**: re-sampling of data from positive class
  - **Under-sampling**: randomly eliminate tuples from negative class
  - **Threshold-moving**: moves the decision threshold, t, so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
  - **Ensemble techniques:** Ensemble multiple classifiers introduced above
- Still difficult for class imbalance problem on multiclass tasks

Multiclass Classification

- Classification involving more than two classes (i.e., > 2 Classes)
- Method 1. **One-vs.-all** (OVA): Learn a classifier one at a time
  - Given m classes, train m classifiers: one for each class
  - Classifier j: treat tuples in class j as *positive* & all others as *negative*
  - To classify a tuple **X**, the set of classifiers vote as an ensemble
- Method 2. **All-vs.-all** (AVA): Learn a classifier for each pair of classes
  - Given m classes, construct m(m-1)/2 binary classifiers
  - A classifier is trained using tuples of the two classes
  - To classify a tuple **X**, each classifier votes. X is assigned to the class with maximal vote
- Comparison
  - All-vs.-all tends to be superior to one-vs.-all
  - Problem: Binary classifier is sensitive to errors, and errors affect vote count

Cross Entropy Loss

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{y}_i\right)^2}$$

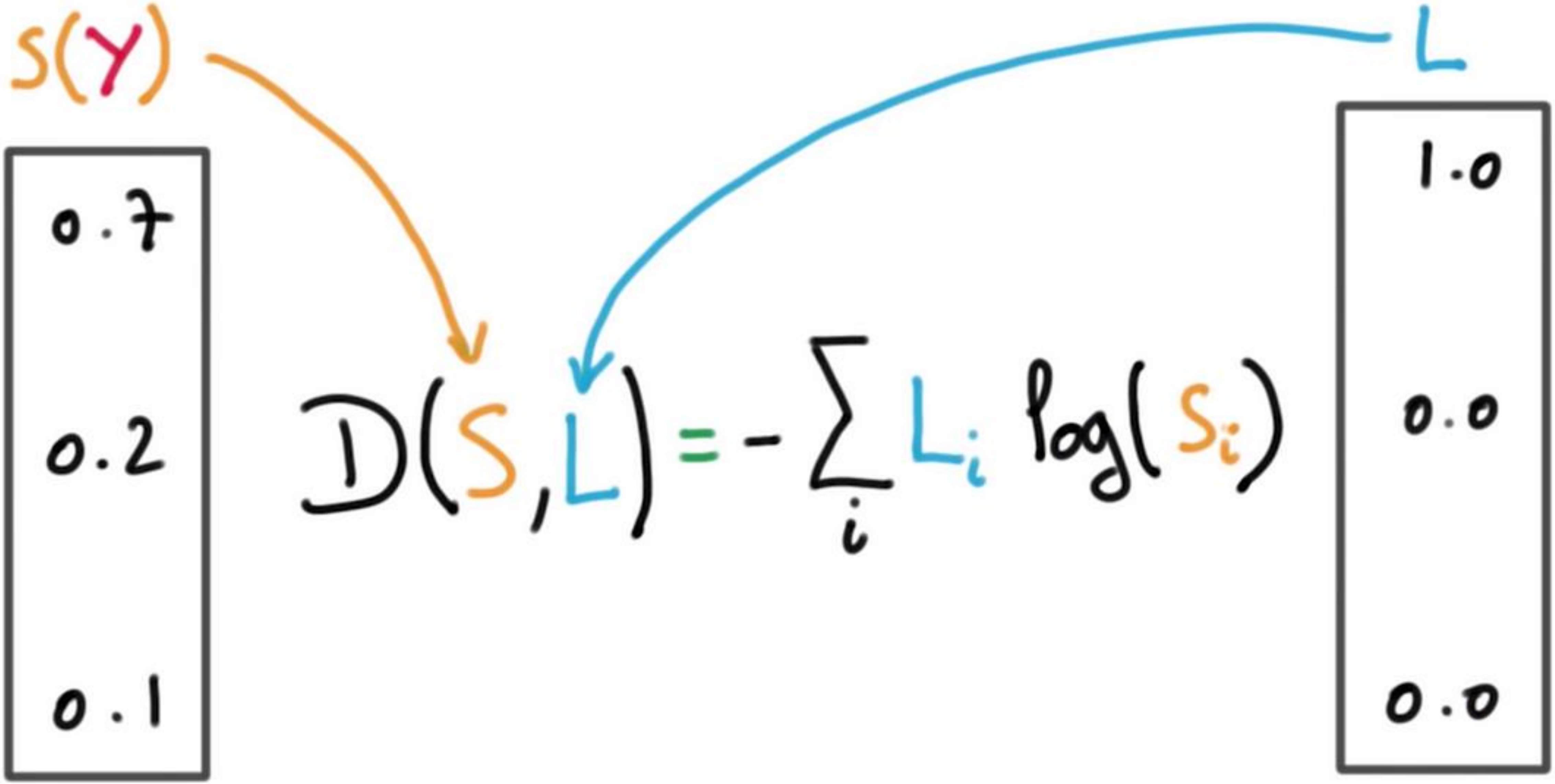


$$H(p, q) = -\sum_{x \epsilon classes} p(x) log\ q(x)$$

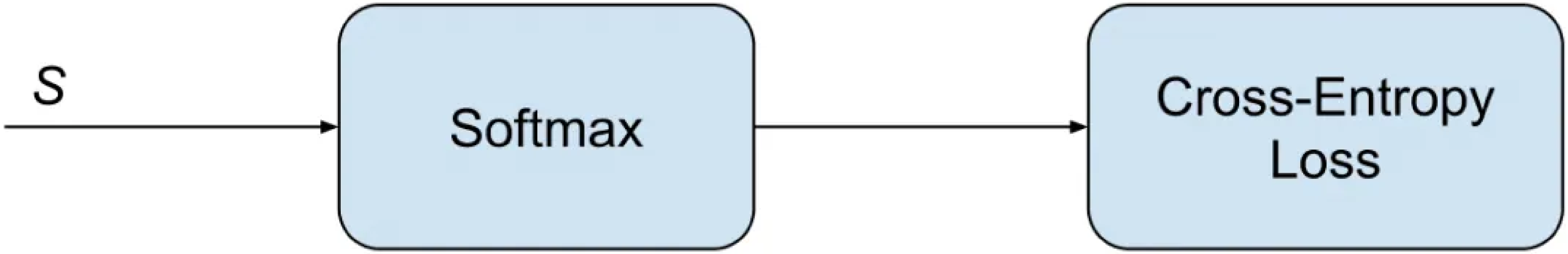
True probability distribution (one-shot)

Your model’s predicted probability distribution

CROSS-ENTROPY



$$D(S, L) = -\sum_i L_i \log(S_i)$$



s

Softmax

Cross-Entropy Loss

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

https://kahoot.it/

1st-3rd: 1 pt
4th and 5th: 0.5 pts