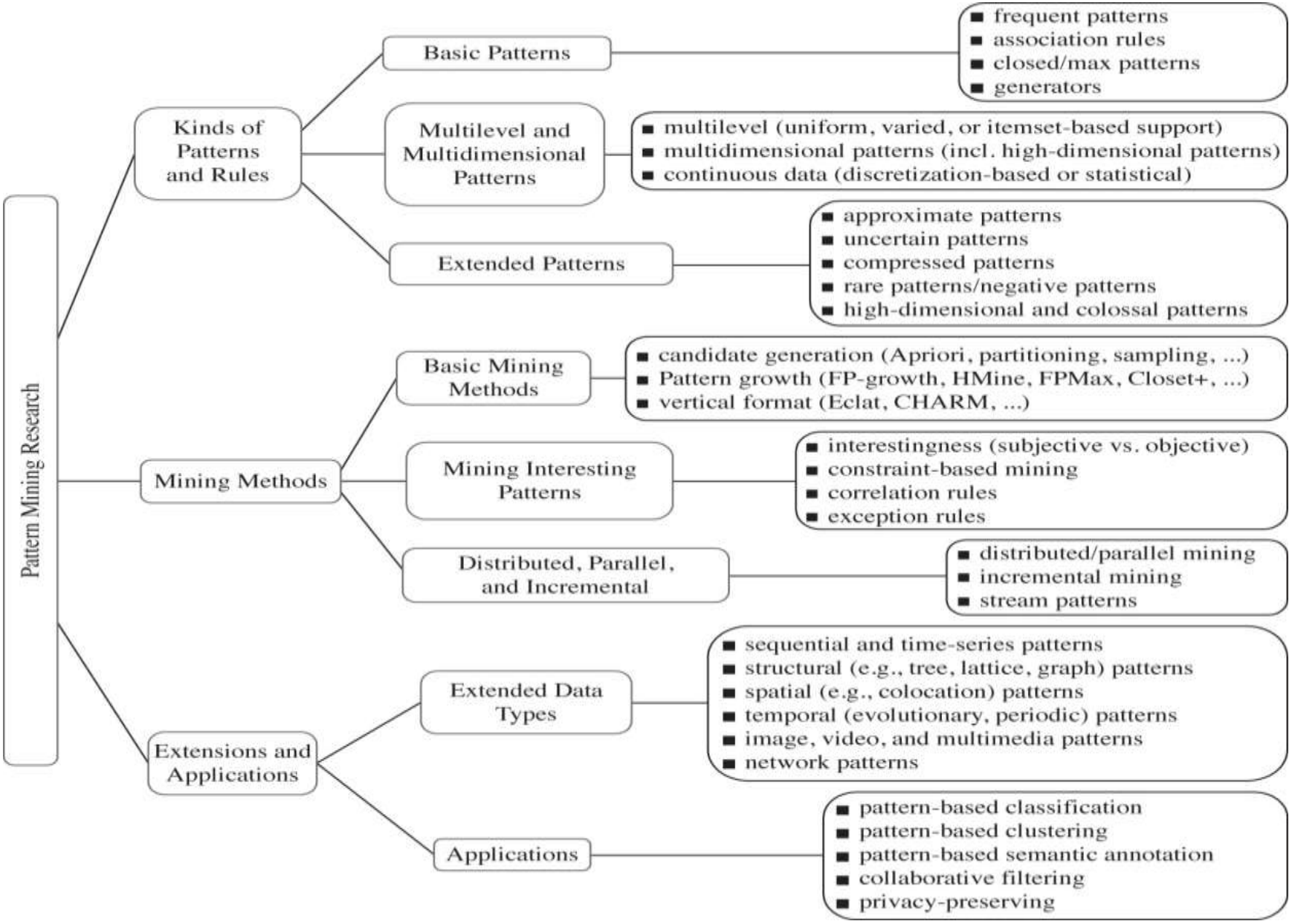
# Advanced Pattern Mining

P. Krishna Reddy, IIIT Hyderabad




Pattern Mining: A Road Map

# Outline

- Pattern Mining: A Road Map
- **Coverage Pattern Mining**
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Multidimensional mining
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

# Discovering Coverage Patterns for Banner Advertisement Placement (Class #13)

P. Gowtham Srinivas, P. Krishna Reddy, S. Bhargav,

R. Uday Kiran, and D. Satheesh Kumar

International Institute of Information Technology

Hyderabad (IIIT-H) , India

Presented by P.Krishna Reddy

E-mail: pkreddy@iiit.ac.in

# Outline

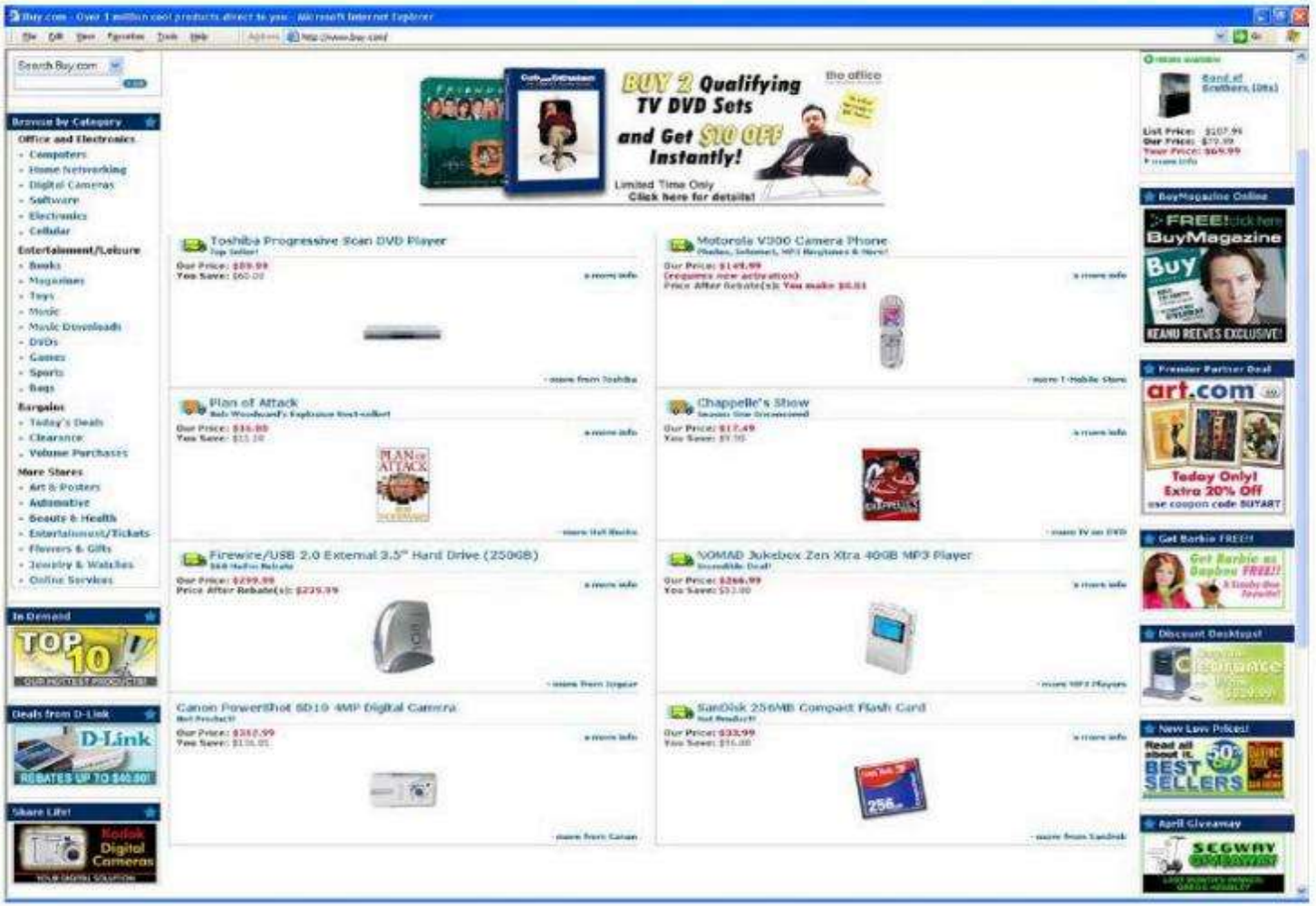
- **Introduction**
- Coverage patterns
- Mining Coverage Patterns
- Experimental Results
- Conclusions and future work

# Introduction

- **Online advertising** is a form of promotion that uses the Internet to deliver marketing messages to attract customers.
  - provides sustainability for the e-commerce enterprises
  - a key factor in the growth of internet economy.
- There are three modes of online advertising:
  - Banner Advertising.
    - hyperlinked to the advertisers primary page or one with more information about the specific product or service advertised.
  - Contextual advertising
    - Relevant advertisements are assigned to a web page based on the content of the web page.
  - Sponsored search advertising
    - Advertisements are placed alongside with the search results.

# About Banner Advertising

- Banner advertising is one of the dominant modes of online advertising.
- In this paper, we proposed a notion of coverage patterns to help to place the banner advertisements in a better manner.
- Three entities are involved in banner advertising:  advertiser, publisher and visitor.
  - An advertiser is interested in endorsing products through banner advertisements.
  - A publisher manages a web site or an advertisement network that sells banner advertisement space.
  - Finally, a visitor visits the web pages of a web site which contains banners.




Banner advertisements on www.buy.com

# Problem Description

- Goal of advertiser
  - spreading advertisement to a certain percentage of people visiting a web site.
- Goal of the publisher
  - efficiently sell the advertising space available in the web pages of a web site and meeting the demands of multiple advertisers.
- Opportunity:
  - For a given web site and period, one can analyse the visitors' behaviour by processing the transactions generated based on click stream dataset and identify the sets of web pages that cover a given percentage of visitors' population.
- Research Issue:
  - Investigate the approaches for discovering the sets of web pages which can cover a given percentage of visitors' population by analyzing the click stream transactions

# Contributions

- We have proposed methodology to discover coverage patterns from transactional databases.
  - Given a set of data items, a coverage pattern is a set of non-overlapping data items covered by a certain percentage of transactions.
  - An Apriori-like algorithm called CMine is proposed for mining coverage patterns.

# Related Work

- Coverage patterns
  - The notion of coverage is being used for solving the set cover problem [2] in set theory and node cover problem [3] in graphs respectively.
  - In [4], the notion of coverage and overlap is used to examine the creation of a tag cloud for exploring and understanding a set of objects.
  - In [5], the notion of coverage and overlap is used to solve the problem of topical query decomposition.
- Online Advertisement
  - Most of the research work on online advertisement has been focused on auction models [7], keyword or phrase identification based on user queries [8], contextual advertising [9] and allocation and scheduling of advertisements [10].
- To our knowledge, not much amount of research work has been carried out on improving the options offered by the publisher to the advertisers

# Advantages

- We have proposed a different kind of knowledge patterns.
- The proposed patterns can be employed in improving the performance of several applications such as banner advertisements.
  - help the advertiser by making his advertisement visible to a certain percentage of web site visitors.
  - Ensuring the publisher to meet the demands of multiple advertisers by considering several groups of potential pages.

# Outline

- Introduction
- **Coverage patterns**
- Mining Coverage Patterns
- Experimental Results
- Conclusions and future work

# Coverage Patterns and Banner Advertisement

- We identify the issue of banner advertisement placement as one of the potential application of coverage patterns.
  - For a given e-commerce web site, transactions generated from click stream dataset can be used to identify the sets of web pages that cover a given percentage of visitors' population.
  - We consider transactions generated from click stream data of a web site. However, the model can be extended to any transactional data set.

# Basic Terminology

- Let W = {w1,w2, · · · , wn} be a set of identifiers of web pages and D be a set of transactions, where each transaction T is a set of web pages such that $T \subseteq W$.
- Associated with each transaction is a unique transactional identifier called TID.
- A set of web pages in W i.e., X = {wp, ….. ,wq, wr},

  $1 \leq p \leq q \leq r \leq n$, is called a pattern.
- A pattern containing ‘k’ number of web pages is called a k-pattern. In other words, the length of k-pattern is k.
- The percentage of transactions in D that contain the web page $wi \in W$ is known as the “relative frequency of a web page $wi \in W$” and denoted as RF(wi).
- Let $|T^{wi}|$ indicates the total number of transactions that contain wi. The relative frequency of wi is denoted as RF(wi). That is, $RF(wi) = (|T^{wi}|)/|D|$.

# Proposed Model…

TRANSACTIONAL DATABASE

Table 1.

| TID | Web pages | TID | Web pages |
|---|---|---|---|
| 1 | a, b, c | 6 | b, d |
| 2 | a, c, e | 7 | b, d |
| 3 | a, c, e | 8 | b, e |
| 4 | a, c, d | 9 | b, e |
| 5 | b, d, f | 10 | a, b |

| Item | Relative Frequency (RF) |
|---|---|
| a | 0.5 |
| b | 0.7 |
| c | 0.4 |
| d | 0.4 |
| e | 0.4 |
| f | 0.1 |

- {a, b} is a pattern. Since there are two web pages in this pattern it is a 2-pattern.

# Frequent Page

- Note that from the advertisement point of view the pages that are visited by more number of users are interesting. We capture this aspect with the notion of frequent page.

- The frequent web pages are web pages which have relative frequency no less than the user-specified threshold value, called minimum relative frequency(*minRF*) . That is, $RF(w_i) \geq minRF.$

- Example 2: Continuing with the example, the relative frequency of 'a' i.e., $RF(a) = |T^a|/|D| = 5/10 = 0.5$. If the user-specified minRF = 0.5, then 'a' is called a frequent web page because RF(a) ≥ minRF.

# Coverage Set

- The coverage set is a set of transactions.
  - Notion: How many users visit at least one web page in the set.
- Coverage set of a pattern X = {$w_p$, ….. ,$w_q$, $w_r$}, $1 \leq p \leq q \leq r \leq n$:
  - The set of distinct TIDs containing at least one web page of X is called the coverage set of pattern X and is denoted as CSet(X). Therefore, $CSet(X) = \{T^{w_p} \cup \cdots \cup T^{w_q} \cup T^{w_r}\}$.

| TID | Web pages | TID | Web pages |
|---|---|---|---|
| 1 | a, b, c | 6 | b, d |
| 2 | a, c, e | 7 | b, d |
| 3 | a, c, e | 8 | b, e |
| 4 | a, c, d | 9 | b, e |
| 5 | b, d, f | 10 | a, b |

Example: The set of tids containing the web page `a' i.e., $T^a$ ={1,2,3,4,10}. Similarly, $T^b$={1,5,6,7,8,9,10}.

The coverage set of {a,b} i.e., CSet({a,b}) ={1,2,3,4,5,6,7,8,9,10} .

Note: In case of frequent patterns, the support of {a,b} ={1,10}=2

# Coverage Support

- A pattern will be interesting if its coverage set contains more than a threshold number of transactions. This aspect is captured through the notion of coverage support.
- ***Coverage-support of a pattern X*** *: The ratio* of size of coverage set of *X* to the transactional database size is called the coverage-support of pattern *X and denoted as CS(X).*
  - *CS(X) = |CSet(X)|/ |D|.*
- For a pattern *X, CS(X)* $\in$ *[0, 1].*
  - If CS(X) = 0, no single web page of X has appeared in the entire transactional database.
  - If CS(X) = 1, every transaction in T contains at least one web page wj $\in$ X.
- Example: Coverage support of {a,b} i.e., CS({a,b}) = |CSet({a,b})|/|D| = 10/10=1.

# Overlap ratio

- Adding other web pages to pattern X may not increase the coverage support, significantly.
  - Due to co-occurrence, there is an overlap of coverage set of X and coverage set of new single web page.
  - Such a pattern can be uninteresting to the advertiser because, the advertisement will be displayed to the same user multiple times.
- Example: $T^a$={1,2,3,4,10}. $T^b$={1,5,6,7,8,9,10}, and $T^c$={1,2,3,4}. CS(a)=0.5, CS(b)=0.7, CS(c)=0.4.
  - We can note that CS (a) and CS(a,c)=5/10=0.5. However, this pattern is uninteresting as the pattern `c' has not increased coverage-support of the pattern `a'.
  - We can note that CS(a,b)=1.As compared to {a,c}, the pattern {a,b} is interesting because, the web page `b' has increased the coverage support of a pattern.

# Overlap Ratio: first definition

**Definition 3** *(Overlap ratio (OR) of a pattern $Y$.) The OR of a pattern $Y = X \cup \{w_r\}$, where $X$ can be empty or $X = \{w_p, \cdots, w_q\}$, $1 \leq p, q, r \leq n$ is defined as the ratio of the number of transactions common in $CSet(X)$ and $CSet(\{w_r\})$ to $CSet(\{w_r\})$, i.e., $OR(Y) = \frac{|(Cset(X)) \cap (Cset\{w_r\})|}{|Cset\{w_r\}|}$.*

# …Overlap Ratio

- For a pattern X, OR(X) ∈ [0, 1]. If OR(X) = 0, there exists no common transactions between $X - \{w_r\}$ and $\{w_r\}$.
- If OR(X) = 1, then $w_r$ has occurred in the transactions where at least one web page $w_j \in (X - \{w_r\})$ has occurred.
- Continuing with the example,
  - The OR({a,b}) = |CSet(b) ∩ CSet(a)|/|CSet(a)| = 2/5 = 0.4.
- Note that a coverage pattern is interesting if it has high coverage support and low overlap ratio.
  - As a result an advertisement is exposed to more number of users by reducing repetitive display of the advertisement. The definition of coverage pattern is as follows

# Coverage Pattern

- ***Coverage pattern X*** : A pattern X ={$w_p$, …. , $w_q$, $w_r$}, where 1 ≤ p ≤ q ≤ r ≤ n, is said to be a coverage pattern if CS(x) ≥ minCS and OR(X) ≤ maxOR and RF($w_i$) ≥ minRF ∀ $w_i$ ∈ X.

  A coverage pattern *X* having CS(X) = a% and OR(X) = b% is expressed as

  X [CS = a%, OR = b%]

- If minRF=0.4, minCS=0.7 and maxOR=0.5, then the pattern {a,b} is a coverage pattern. It is because RF(a) >= minRF, RF(b) >= minRF, CS({a,b}) >= minCS and OR({a,b}) <= maxOR. This pattern is written as follows:
  - {a,b} [CS=1 (=100%), OR= 0.4 (=40%)]

# Problem Statement

- Given a transactional database *D*, set of web pages *W* and user-specified minimum relatively frequency (*minRF)*, minimum coverage support *(minCS)* and maximum overlap ratio (*maxOR)*, discover complete set of coverage patterns such that
  i. If X is a coverage 1-pattern (i.e., k = 1), then $RF(w_i) \geq minRF$, $w_i \in X$.
  ii. Otherwise (i.e., when k > 1), each coverage pattern X must have $CS(X) \geq minCS$, $OR(X) \leq maxOR$ and $RF(w_i) \geq minRF$, where $w_i \in X$.

# Outline

- Introduction
- Coverage patterns
- **Mining Coverage Patterns**
- Experimental Results
- Conclusions and future work

# Extracting Coverage Patterns

- Naïve approach
  - Extract all  possible patterns (leads to combinatorial explosion)
  - Each pattern in CP is added to the coverage pattern set if it satisfies minCS; minRF and maxOR constraints.
- Opportunity
  - The search space can be reduced if the coverage pattern satisfies downward closure property on either coverage support or overlap ratio.

# Extracting Coverage Patterns

- Coverage support does not satisfy downward closure property. That is, although a pattern satisfies minCS, it is not necessary that all its non-empty subsets will also satisfy minCS value.
  - Consider the patterns {a}, {e} and {a,e}. The coverage supports of these patterns are 0.5, 0.4 and 0.7, respectively. If the user-specified minCS=0.7, then the pattern {a,e} satisfies minCS value. However, its non-empty subsets do not satisfy minCS value.
- The parameter overlap ratio also does not satisfy downward closure property if a pattern is considered as an unordered set of web pages.
- The Overlap ratio satisfies  downward closure property if a pattern is an ordered set, where web pages are sorted in descending order of their frequencies. This property is known as the sorted closure property.

**Coverage patterns and sorted closure property** Extracting the complete set of CPs for a given $minRF$, $minCS$ and $maxOR$ is a challenging task, because the measure $CS$ does not satisfy the *downward closure property*. Although a pattern satisfies $minCS$, it is not necessary that all its non-empty subsets will also satisfy $minCS$ value. For example, consider the patterns $\{a\}$, $\{e\}$ and $\{a, e\}$ from Table 1. The coverage-supports of these patterns are 0.5, 0.4 and 0.7, respectively. If the user specifies $minCS = 0.7$, the pattern $\{a, e\}$ satisfies $minCS$ value. However, the patterns $\{a\}$ and $\{e\}$ do not satisfy $minCS$ value. It can also be noted that the preceding definition of $OR$ measure also does not satisfy *downward closure property*. From Table 1, $OR(\{a, c, b\}) = \frac{|CSet(\{a,c\}) \cap CSet(\{b\})|}{|CSet(\{b\})|} = \frac{2}{7}$ and $OR(\{a, c\}) = \frac{|CSet(\{a\}) \cap CSet(\{c\}|)}{|CSet(\{c\})|} = \frac{4}{4} = 1$. If the user specifies $maxOR = \frac{2}{7}$, $OR(\{a, c, b\}) \leq maxOR$ value. However, $OR(\{a, c\}) \not\leq maxOR$ value.

It was observed that the $OR$ satisfies *downward closure property*, if the items are ordered in descending order of their frequencies. This property is called *sorted-closure property* [13] which is given in Property 1 and the correctness is shown in Lemma 2.

**Property 1** *Sorted Closure Property: Let* $X = \{w_p, \cdots w_q, w_r\}$ *be a pattern such that* $RF(w_p) \geq \cdots RF(w_q) \geq RF(w_r)$ *and* $1 \leq p, q, r \leq n$. *If* $OR(X) \leq maxOR$, *then all its non-empty subsets containing* $w_r$ *and having size* $k \geq 2$ *will also have* overlap ratio *less than or equal to* $maxOR$.

**Lemma 1** *If* $X \subset Y$, *then* $CSet(X) \subseteq CSet(Y)$.

**Lemma 2** *If* $OR(X) \leq maxOR$, *then* $OR(Y) \leq maxOR$. $\forall\ Y \subset X, w_r \in Y$.

*Proof* Let $w_a$, $w_b$ and $w_c$ be the web pages having $RF(w_a) \geq RF(w_b) \geq RF(w_c)$. If $OR(\{w_a\} \cup \{w_c\}) > maxOR$, then $OR((\{w_a\} \cup \{w_b\}) \cup \{w_c\}) > maxOR$ because from Lemma 1, $\frac{|CSet(\{w_a\}) \cap CSet(\{w_c\})|}{|CSet(\{w_c\})|} \leq \frac{|CSet(\{w_a \cup w_b\}) \cap CSet(\{w_c\})|}{|CSet(\{w_c\})|}$.

# Overlap ratio: Second Definition

- If $X \subset Y$, then $CSet(X) \subset= CSet(Y)$.
- Sorted closure property: Let $X = \{w_p, \ldots.., w_q, w_r\}$, where $1 \leq p \leq q \leq r \leq n$ be a pattern such that $T^{w_p} >= \cdots >= T^{w_q} >= T^{w_r}$. If $OR(X) <= maxOR$, all its non-empty subsets containing $w_r$ and having size $k >= 2$ will also have overlap ratio less than or equal to maxOR.
- Rationale: Let $w_a$, $w_b$ and $w_c$ be the web pages having $RF(w_a) \geq RF(w_b) \geq RF(w_c)$. If $OR(w_a \cup w_c) > maxOR$, then $OR(\{w_a \cup w_b\} \cup w_c) > maxOR$ because from above property

  $|CSet(w_a) \cap CSet(w_c)| / |CSet(w_c)| <= CSet(\{w_a \cup w_b\}) \cap CSet(w_c)/CSet(w_c)$.
- (Non-overlap pattern X.) A pattern X is said to be *non-overlap* if $OR(X) \leq maxOR$ and $RF(w_i) \geq minRF \ \forall w_i \in X$.

# Extracting Coverage Patterns

- Every coverage pattern is a non-overlap pattern
- However, every non-overlap pattern may not be a coverage pattern.
- The sorted closure property of non-overlap patterns is used for minimizing the search space while mining complete set of coverage patterns by designing an algorithm similar to the Apriori algorithm .

# Proposed Algorithm

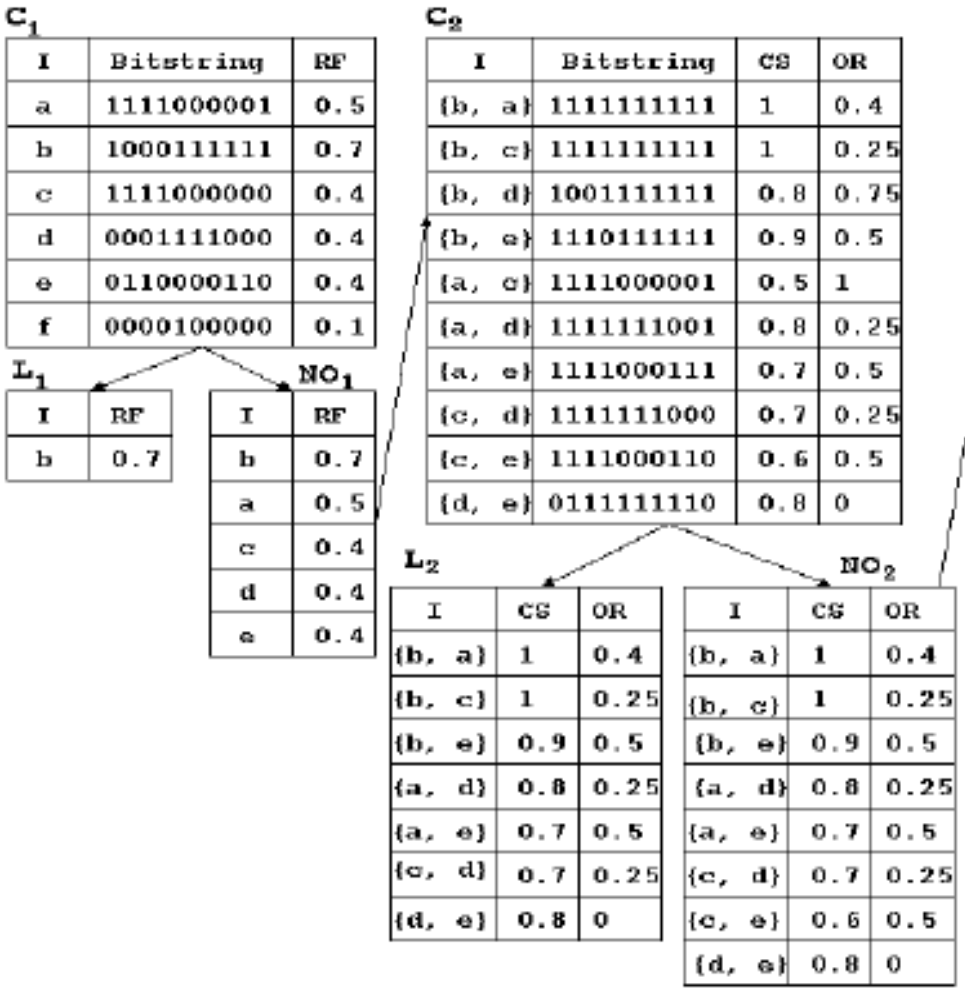
- The proposed algorithm *CMine* employs *level-wise* search to discover the complete set of coverage patterns.
- In *level-wise* search*, k*-patterns are used to explore *(k + 1)*-patterns.
- Notations

| | |
|---|---|
| F | Set of Frequent items |
| $C_k$ | Set of Candidate K-patterns |
| $L_k$ | Set of Coverage K-patterns |
| $NO_k$ | Set of non-overlap K-patterns |

# CMine Algorithm

- Step 1: In the first scan of the database, ***CMine*** discovers set of all frequent items (denoted as *F)* and coverage 1-patterns (denoted as $L_1$). Next, items in $NO_1$ are sorted in descending order of their frequencies.
- Step 2: Using $NO_1$ as a seed set, candidate patterns $C_2$ are generated.
- Step 3: From $C_2$, the patterns that satisfy *minCS* and *maxOR* are generated as *coverage* 2-patterns, $L_2$.
- **Simultaneously, all candidate 2-patterns that satisfy *maxOR* are generated as *non-overlap* 2-patterns $NO_2$.**
- Step 4 : $C_3$ *is* generated by combining $NO_2 \bowtie NO_2$.
- Step 5 : From $C_3$, $L_3$ *and* $NO_3$ are discovered.
- The above process is repeated until no new coverage pattern is found, or no new candidate pattern can be generated.

# CMine Algorithm Example



$C_1$

| I | Bitstring | RF |
|---|---|---|
| a | 1111000001 | 0.5 |
| b | 1000111111 | 0.7 |
| c | 1111000000 | 0.4 |
| d | 0001111000 | 0.4 |
| e | 0110000110 | 0.4 |
| f | 0000100000 | 0.1 |

$L_1$

| I | RF |
|---|---|
| b | 0.7 |

$NO_1$

| I | RF |
|---|---|
| b | 0.7 |
| a | 0.5 |
| c | 0.4 |
| d | 0.4 |
| e | 0.4 |

$C_2$

| I | Bitstring | CS | OR |
|---|---|---|---|
| {b, a} | 1111111111 | 1 | 0.4 |
| {b, c} | 1111111111 | 1 | 0.25 |
| {b, d} | 1001111111 | 0.8 | 0.75 |
| {b, e} | 1110111111 | 0.9 | 0.5 |
| {a, c} | 1111000001 | 0.5 | 1 |
| {a, d} | 1111111001 | 0.8 | 0.25 |
| {a, e} | 1111000111 | 0.7 | 0.5 |
| {c, d} | 1111111000 | 0.7 | 0.25 |
| {c, e} | 1111000110 | 0.6 | 0.5 |
| {d, e} | 0111111110 | 0.8 | 0 |

$L_2$

| I | CS | OR |
|---|---|---|
| {b, a} | 1 | 0.4 |
| {b, c} | 1 | 0.25 |
| {b, e} | 0.9 | 0.5 |
| {a, d} | 0.8 | 0.25 |
| {a, e} | 0.7 | 0.5 |
| {c, d} | 0.7 | 0.25 |
| {d, e} | 0.8 | 0 |

$NO_2$

| I | CS | OR |
|---|---|---|
| {b, a} | 1 | 0.4 |
| {b, c} | 1 | 0.25 |
| {b, e} | 0.9 | 0.5 |
| {a, d} | 0.8 | 0.25 |
| {a, e} | 0.7 | 0.5 |
| {c, d} | 0.7 | 0.25 |
| {c, e} | 0.6 | 0.5 |
| {d, e} | 0.8 | 0 |

$C_3$

| I | Bitstring | CS | OR |
|---|---|---|---|
| {b, a, c} | 1111111111 | 1 | 1 |
| {b, a, e} | 1111111111 | 1 | 1 |
| {b, c, e} | 1111111111 | 1 | 1 |
| {a, d, e} | 1111111111 | 1 | 0.5 |
| {c, d, e} | 1111111110 | 0.9 | 0.5 |

$L_3/NO_3$

| I | CS | OR |
|---|---|---|
| {a, d, e} | 1 | 0.5 |
| {c, d, e} | 0.9 | 0.5 |

Note:

(i){b,a} will be joined with {b,c} and forms a pattern {b,a,c} as RF(a) > RF(c).

(ii) {b,c} can not be joined with {b,a} to form a pattern {b,c,a} as RF(c) < RF(a).

minRF = 0.4, minCS = 0.7, maxOR = 0.5 for database given in Table 1.

# Outline

- Introduction
- Coverage patterns
- Mining Coverage Patterns
- **Experimental Results**
- Conclusions and future work

# Experimental Results

- For experimental purposes we have chosen two kinds of datasets: (i)real world dataset and (ii)synthetic-dataset.
- The four real world datasets considered for the experiments are described below.
  - Kosarak dataset is a sparse dataset with 9,90,002 number of transactions containing 41,270 distinct items.
  - MSNBC dataset contains data from Internet Information Server (IIS) logs for msnbc.com and news-related portions of msn.com for the entire day of September, 28, 1999 . Requests are at the level of page category. The number of categories are 17 and the number of transactions are 989,818
  - Mushroom dataset is a dense dataset containing 8,124 transactions and 119 distinct items.
  - BMS-POS dataset contains click stream data of a dotcom company . The dataset contains 515,597 number of transactions and 1656 distinct items.
- The synthetic dataset T40I10D100K dataset which is generated by the dataset generator [9]. The dataset contains 100000 transactions and 941 distinct items.
- The CMine algorithm was written in Java and run with Windows XP on a 2.66 GHz machine with 2GB memory.

# Usefulness of coverage patterns

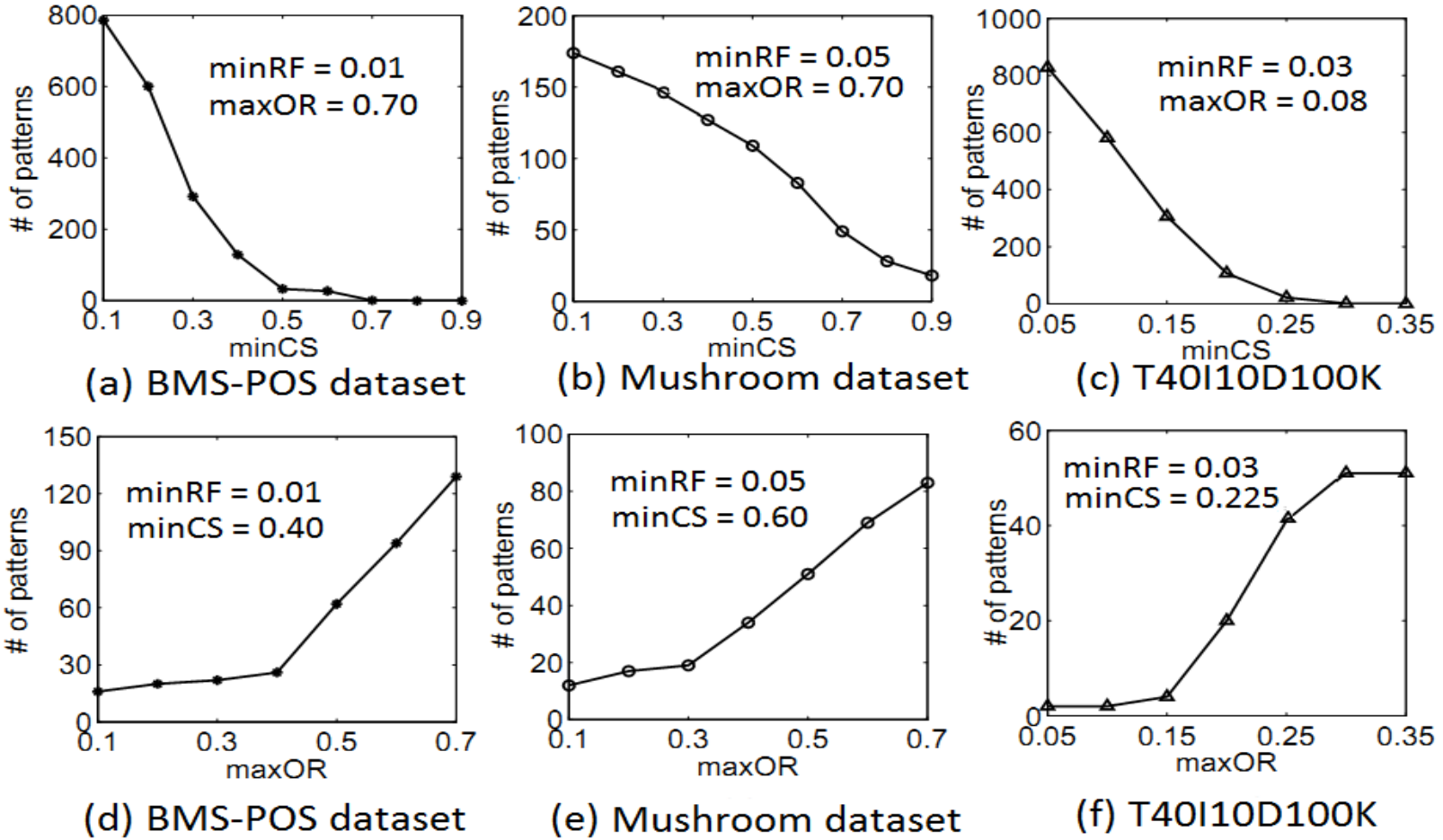
- The names of web page categories involved in MSNBC are
  - "frontpage", "news", "tech", "local", "opinion", "on-air", "misc", "weather", "health", "living", "business", "sports", "summary", "bbs" (bulletin board service), "travel", "msn-news", and "msn-sports".

| S.No | Coverage Pattern | CS |
|---|---|---|
| 1 | {local, misc, front page} | 0.42 |
| 2 | {news, health, front page} | 0.43 |
| 3 | {tech, opinion, front page} | 0.41 |
| 4 | {on-air, news, misc} | 0.40 |
| 5 | {tech, weather, on-air} | 0.41 |
| 6 | {sports, misc, opinion} | 0.43 |

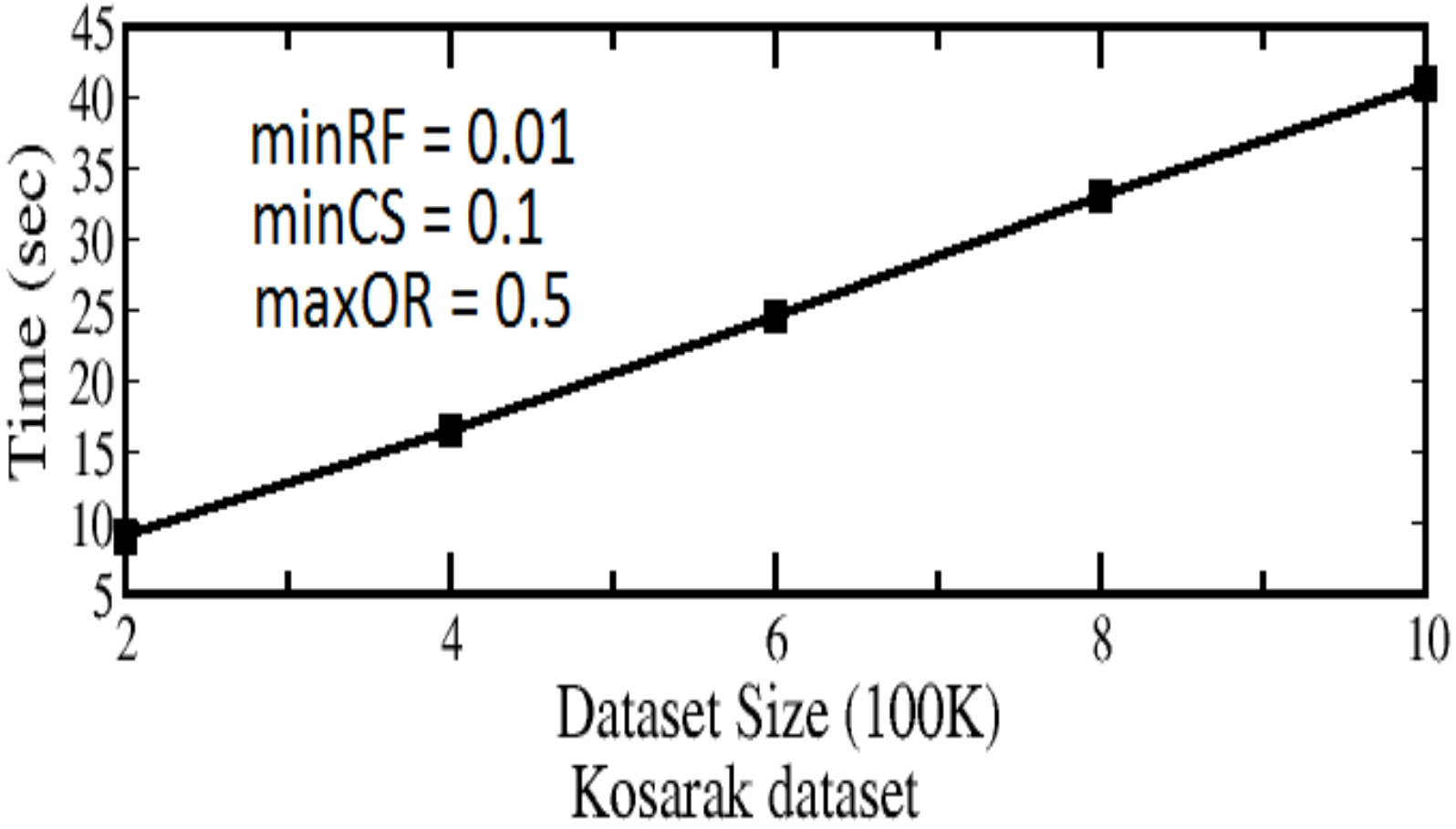
- Advantage:

Proposed approach provides flexibility to the publisher to meet the demands of multiple advertisers by considering different sets of web pages.

# Coverage pattern generation



# Scalability of CMine Algorithm



# Conclusions and Future Work

- Proposed a framework of data mining pattern called “coverage pattern”
- Proposed a methodology to extract the same from transactional databases.
- Through experimental results, we have shown that the proposed model and methodology can effectively discover coverage patterns.
- The coverage patterns help the publisher to meet the demands of multiple advertisers.

# Conclusions and Future Work

- We are going to investigate how both frequent and coverage pattern knowledge can be used for efficient banner advertisement placement.
- How the content of the web page and search query can be exploited to explore content specific coverage patterns.
- The notion of coverage patterns can be extended to other domains like bio-informatics for extracting potential knowledge patterns.

# Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- **Multilevel Pattern Mining**
- Quantitative Pattern Mining
- Mining Multidimensional Associations
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

# About Multilevel Pattern Mining

- Applications require association rules at many levels of abstraction
  - “80 percent of customers that purchase milk may also purchase Bread”
  - “75 percent of people buy wheat bread if they buy “milk(2%)”
- The association relationship in the latter statement is expressed at a lower level of abstraction but carries more specific and concrete information than that in the former.
- Therefore, a data mining system should provide efficient methods for mining multiple-level association rules.

# Why Multiple-Level Association Rules?

| TID | items |
|---|---|
| T1 | {m1, b2} |
| T2 | {m2, b1} |
| T3 | {b2} |

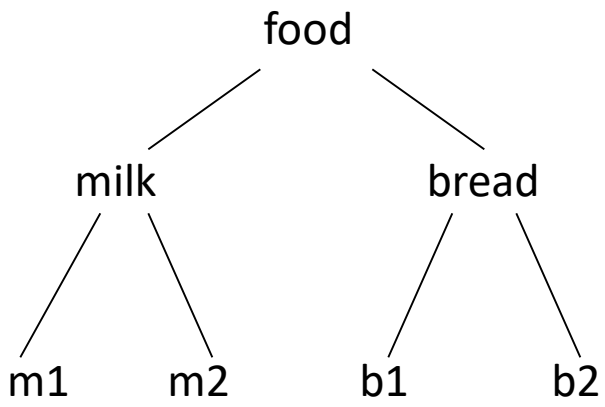
Frequent itemset: {b2}

Association rules: none

Is this database useless?

# Why Multiple-Level Association Rules?

What if we have this abstraction tree?




| TID | items |
|---|---|
| T1 | {milk, bread} |
| T2 | {milk, bread} |
| T3 | {bread} |

minisup = 50%  miniconf = 50%

Frequent itemset: {milk, bread}

A.A rules: milk <=> bread

## Why Multiple-Level Association Rules?

- Sometimes, at primitive data level, data does not show any significant pattern. But there are useful information hiding behind.

- The goal of Multiple-Level Association Analysis is to find the hidden information in or between levels of abstraction

# Input

- Input to Multi-Level Association Rule Mining
  - 1) data at multiple levels of abstraction, and
  - 2) efficient methods for multiple-level rule mining.
- The first requirement can be satisfied by providing concept taxonomies from the primitive level concepts to higher levels.
- In many applications, the taxonomy information is either
  - stored implicitly in the database
  - provided by experts or users,
  - or computed by applying some cluster analysis methods
- We assume that concept taxonomies exist.

## Requirements in Multiple-Level Association Analysis

Two general requirements to do multiple-level association rule mining:

1) Provide data at multiple levels of abstraction. (a common practice now)

2) Find efficient methods for multiple-level rule mining.

# Observation

- One choice
  - Direct application of the existing single-level association rule mining methods to multiple-level association mining.
  - For example, one may apply the Apriori algorithm to examine data items at multiple levels of abstraction under the same minimum support and minimum confidence thresholds.
- Apriori with single minimum support and minimum confidence may lead to some undesirable results.

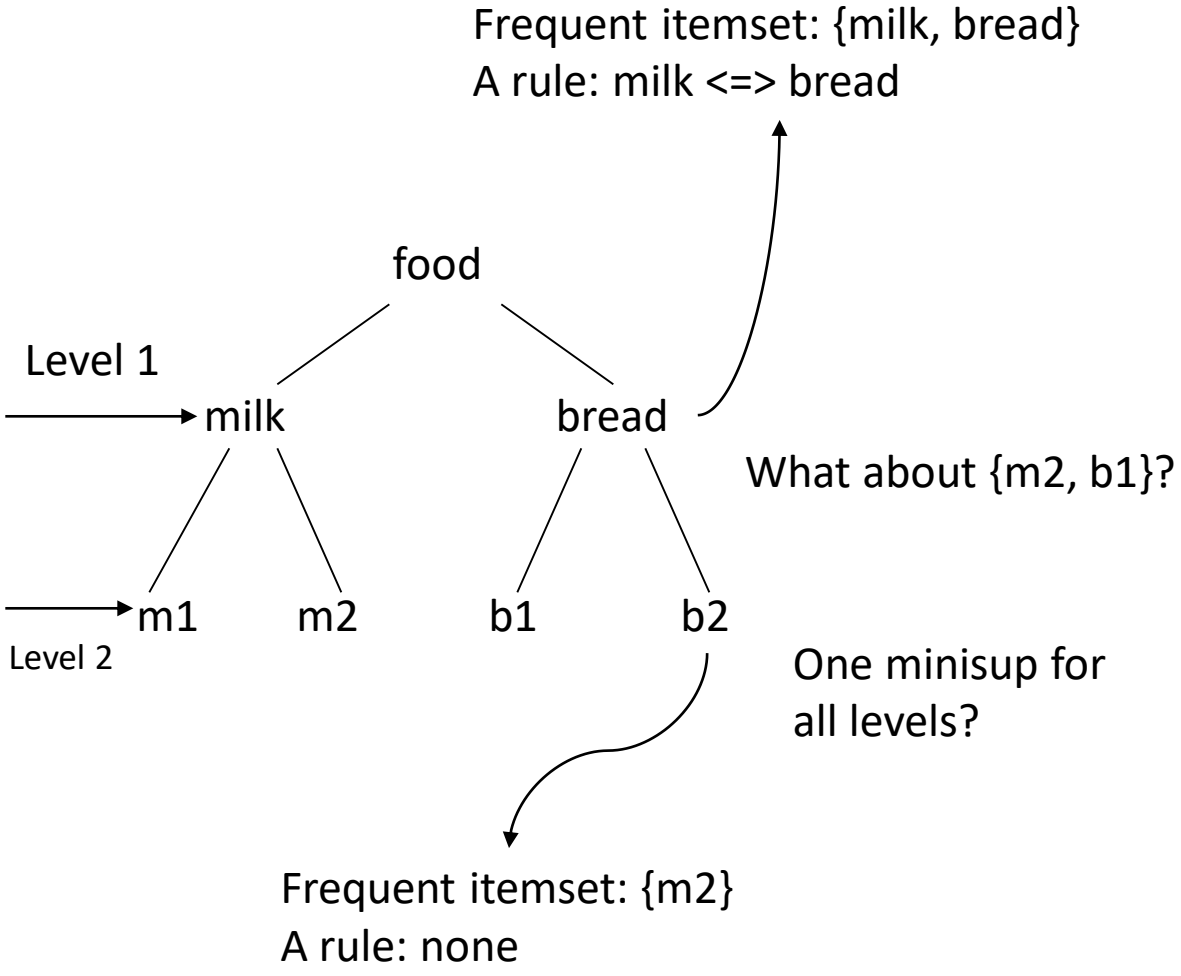
# Observation..

- Undesirable results with apriori
  - First, large support is more likely to exist at high levels of abstraction.
    - If one wants to find strong associations at relatively low levels of abstraction, the minimum support threshold must be reduced substantially, which may lead to the generation of many uninteresting associations at high or intermediate levels.
  - Second, since it is unlikely to find many strong association rules at a primitive concept level, mining strong associations should be performed at a rather high concept level. But, rules at high concept levels,
    - may often lead to the rules corresponding to prior knowledge and expectations, such as “milk->bread”, (which could be common sense),
    - or lead to some uninteresting attribute combinations if the minimum support is allowed to be rather small, such as **“toy -> milk**”,

# Algorithm : observation

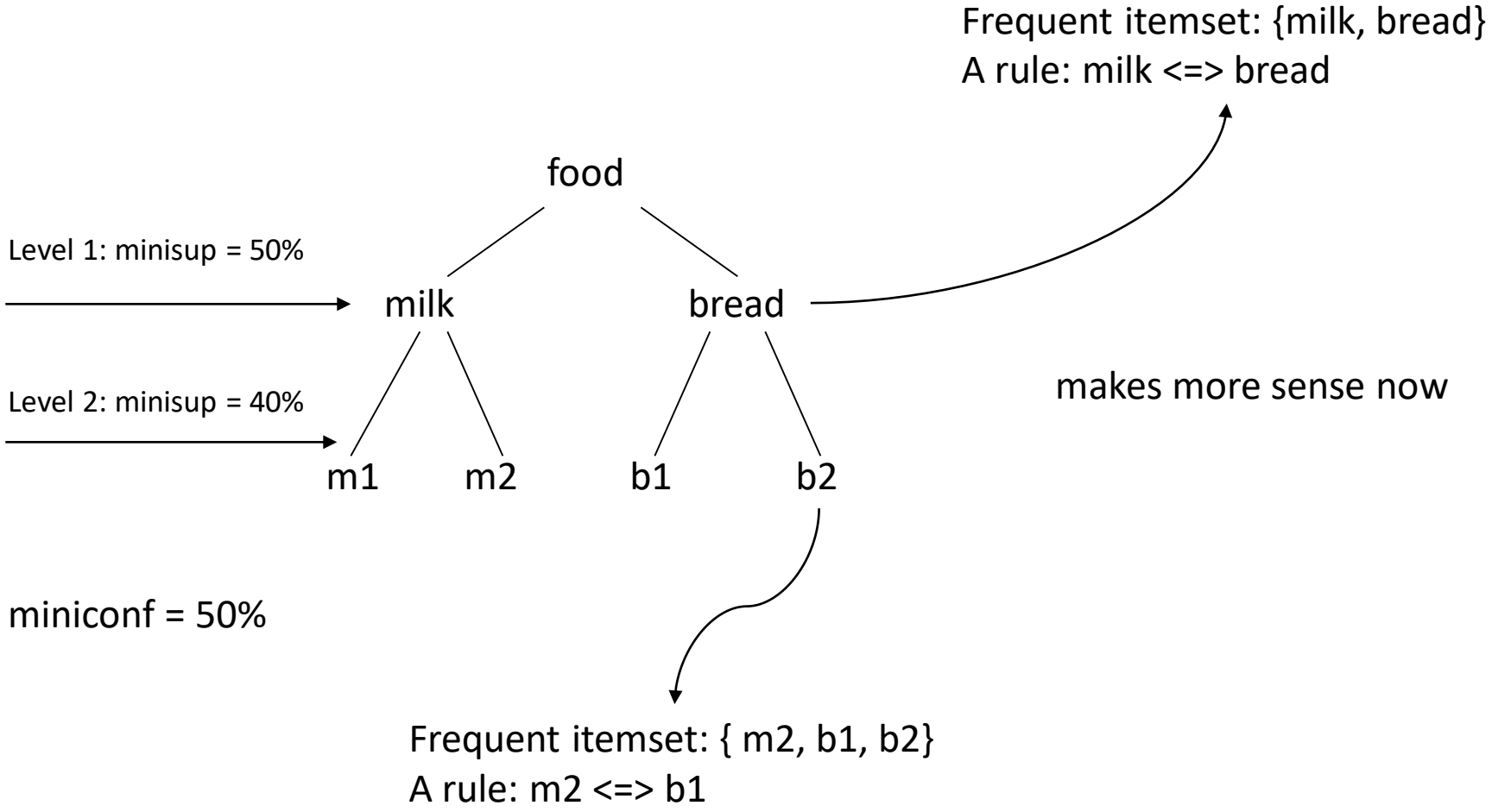
| TID | items |
|---|---|
| T1 | {milk, bread} |
| T2 | {milk, bread} |
| T3 | {bread} |
| T4 | {milk, bread} |
| T5 | {milk} |

| TID | items |
|---|---|
| T1 | {m1, b2} |
| T2 | {m2, b1} |
| T3 | {b2} |
| T4 | {m2, b1} |
| T5 | {m2} |




minisup = 50%  miniconf = 50%

# Algorithm : observation



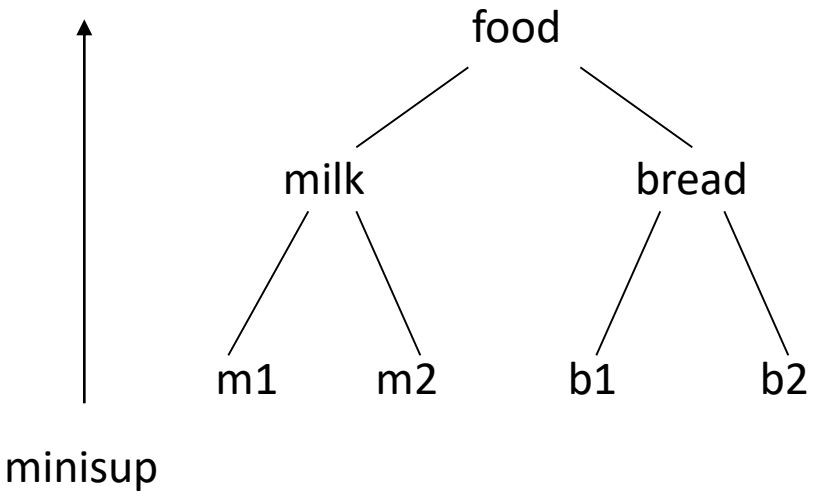
# Algorithm: observation

Drawbacks to use only one minisup:

- If the minisup is too high, we are losing information from lower levels

- If the minisup is too low, we are gaining too many rules from higher levels, many of them are useless

Approach: ascending minisup on each level



# Research Paper: Mining Multiple-level Association Rules in Large Databases

*IEEE Transactions on Knowledge and Data Engineering*, 1999

JIAWEI HAN and YONGJIAN FU

# Multi-level Association Rules

We assume that the database contains: 1) an item data set which contains the description of each item in $\mathcal{I}$ in the form of $\langle A_i, description_i \rangle$, where $A_i \in \mathcal{I}$, and 2) a transaction data set, $\mathcal{T}$, which consists of a set of transactions $\langle T_i, \{A_p, \ldots, A_q\} \rangle$, where $T_i$ is a transaction identifier and $A_i \in \mathcal{I}$ (for $i = p, \ldots, q$).

**Definition 2.2.** *A pattern $A$ is **frequent** in set $S$ at level $l$ if the support of $A$ is no less than its corresponding minimum support threshold $\sigma'_l$. A rule "$A \Rightarrow B/S$" is **strong** if, for a set $S$, each ancestor (i.e., the corresponding high-level item) of every item in $A$ and $B$, if any, is frequent at its corresponding level, "$A \wedge B/S$" is frequent (at the current level), and the confidence of "$A \Rightarrow B/S$" is no less than minimum confidence threshold at the current level.*

# Multi-level Association Rules..

- The definition 2.2
  - patterns to be examined at lower levels to be only those with large supports at their corresponding high levels (and thus avoids the generation of many meaningless combinations formed by the descendants of the infrequent patterns).
- For example, in a sales transaction data set,
  - if milk is a frequent pattern, its lower level patterns, such as 2 percent milk, will be examined;
  - whereas if fish is an infrequent pattern, its descendants, such as salmon, will not be examined further.

# About the Method

- Used hierarchy information encoded transaction table
- Collect task relevant data and mine
- Encoding can be done during the collection

# Encoding Method

- Purpose: To find multiple-level frequent item sets for mining strong association rules in a transaction database
- Input
  - T[1]: a hierarchy-information encoded transaction table of form <TID,Item-set>
  - minisup threshold for each level L in the form: (minsup[L])
- Output: Multiple-level frequent item sets
- Method: A top-down, progressively deepening process which collects frequent item sets at different concept levels as follows.

# Algorithm: An Example

An entry of **sales_transaction** Table

| Transaction_id | Bar_code_set |
|---|---|
| 351428 | {17325,92108,55349,88157,…} |
| | |

A **sales_item** Description Relation

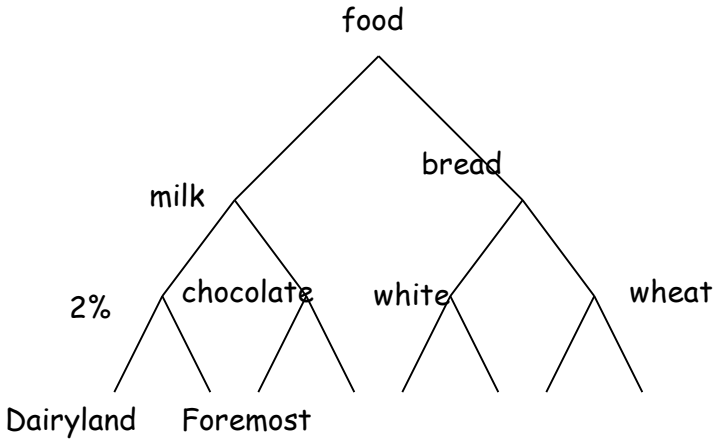
| Bar_code | category | Brand | Content | Size | Storage_pd | price |
|---|---|---|---|---|---|---|
| 17325 | Milk | Foremost | 2% | 1ga. | 14(days) | $3.89 |

Example 2.1. Let the query be to find multiple-level strong associations in the database in Table 1 for the purchase patterns related to category, content, and brand of the foods which can only be stored for less than three weeks.

# Algorithm: An Example

Encode the database with layer information

| GID | bar_code | category | content | brand |
|---|---|---|---|---|
| 112 | 17325 | Milk | 2% | Foremost |




First 1: implies milk

Second 1: implies 2% content

2: implies Foremost brand

# Algorithm: An Example

Encoded Transaction Table:T[1]

| TID | Items |
|---|---|
| T1 | {111,121,211,221} |
| T2 | {111,211,222,323} |
| T3 | {112,122,221,411} |
| T4 | {111,121} |
| T5 | {111,122,211,221,413} |
| T6 | {211,323,524} |
| T7 | {323,411,524,713} |

# Algorithm: An Example

The frequent 1-itemset on level 1

L[1,1]

Level-1 minsup = 4

T[2]

| Itemset | Support |
|---|---|
| {1**} | 5 |
| {2**} | 5 |

only keep items
in L[1,1] from T[1]

| TID | Items |
|---|---|
| T1 | {111,121,211,221} |
| T2 | {111,211,222} |
| T3 | {112,122,221} |
| T4 | {111,121} |
| T5 | {111,122,211,221} |
| T6 | {211} |

L[1,2]

| Itemset | Support |
|---|---|
| {1**,2**} | 4 |

Use Apriori on each level

# Algorithm: An Example

Level-2 minsup = 3

L[2,1]

| Itemset | Support |
|---|---|
| {11*} | 5 |
| {12*} | 4 |
| {21*} | 4 |
| {22*} | 4 |

L[2,2]

| Itemset | Support |
|---|---|
| {11*,12*} | 4 |
| {11*,21*} | 3 |
| {11*,22*} | 4 |
| {12*,22*} | 3 |
| {21*,22*} | 3 |

L[2,3]

| Itemset | Support |
|---|---|
| {11*,12*,22*} | 3 |
| {11*,21*,22*} | 3 |

# Frequent Item Sets at Level 3

Level-3 minsup = 3

L[3,1]

| Itemset | Support |
|---|---|
| {111} | 4 |
| {211} | 4 |
| {221} | 3 |

L[3,2]

| Itemset | Support |
|---|---|
| {111,211} | 3 |

Only generate T[1] & T[2], all frequent itemsets after level 2 is generated from T[2]

E.g.

Level-1:

80% of customers that purchase milk also purchase bread.

milk ➔ bread with Confidence= 80%

Level-2:

75% of people who buy 2% milk also buy wheat bread.

2% milk ➔ wheat bread with Confidence= 75%

# Multi-level Association: Flexible Support and Redundancy filtering

- Flexible min-support thresholds: Some items are more valuable but less frequent
  - Use non-uniform, group-based min-support
  - E.g., {diamond, watch, camera}: 0.05%; {bread, milk}: 5%; …
- Redundancy Filtering: Some rules may be redundant due to "ancestor" relationships between items
  - milk $\Rightarrow$ wheat bread [support = 8%, confidence = 70%]
  - 2% milk $\Rightarrow$ wheat bread [support = 2%, confidence = 72%]

  The first rule is an ancestor of the second rule
- A rule is *redundant* if its support is close to the "expected" value, based on the rule's ancestor

# MIDSEM
# (UP TO The Preceding Slide)

# Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- **Quantitative Pattern Mining**
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

- Class #14: no class (mid preparation)
- Class #15

# Boolean Association Rules

| Trans-ID | Items |
|---|---|
| 1 | A C D |
| 2 | B C E |
| 3 | A B C E |
| 4 | B E |
| 5 | A B C E |

- The table has an attribute correspond to each item. A record correspond to each transaction.
- Value of attribute is “1” if it exists on record.
- All the attributes are Boolean.

# Boolean Association Rules

| TID | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 | 1 |
| 3 | 1 | 1 | 1 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 | 1 |
| 5 | 1 | 1 | 1 | 0 | 1 |

- Attribute has a value of “1” if the transaction contains the corresponding item; “0” otherwise.

# Large Relational Tables in real-life.

- Relational tables in most business and scientific domains have richer attribute types.
  - Quantitative attributes (e.g. age, income)
  - Categorical attributes (e.g. zip code, marriage status, make of car)
- Can not apply existing methods to extract Boolean association rules to mine association rules over quantitative and categorical attributes.
  - This paper present techniques for discovering such rules.

# Quantitative Association Rules: Example

| RecordID | Age | Married | NumCars |
|---|---|---|---|
| 100 | 23 | No | 1 |
| 200 | 25 | Yes | 1 |
| 300 | 29 | No | 0 |
| 400 | 34 | Yes | 2 |
| 500 | 38 | Yes | 2 |

| Sample Rules | Support | Confidence |
|---|---|---|
| <age:30..39> and <married: yes> ==> <numCars:2> | 40% | 100% |
| <NumCars: 0..1> ==> <Married: No> | 40% | 66.70% |

# Mapping to Boolean Association Rules Problem

- Using <attribute: value> as new attribute, which has only Boolean values
- Categorical: The value of Boolean field corresponding to <attribute1, value1> would be “1”, if atribute1 had value 1 in the original record, and “0” otherwise.
- If the domain of values is large, partition the values into intervals and map each pair to Boolean attribute.

- Use any exiting algorithm a priori or FP Tree.

| RecordID | Age: 20..29 | Age: 30..39 | Married: Yes | Married: No | NumCars: 0 | NumCars: 1 |
|---|---|---|---|---|---|---|
| 100 | 1 | 0 | 0 | 1 | 0 | 1 |
| 200 | 1 | 0 | 1 | 0 | 0 | 1 |
| 300 | 1 | 0 | 0 | 1 | 1 | 0 |
| 400 | 0 | 1 | 1 | 0 | 0 | 0 |
| 500 | 0 | 1 | 1 | 0 | 0 | 0 |

# First Problem with Direct Mapping

- **MinSup**: If the number of intervals for a quantitative attribute is large, the support for any single interval can be low. Some rules involving this attribute may not be found because they lack minimum support.

# Second Problem with Direct Mapping

- **MinConf.** There is some information lost whenever we partition values into intervals. This information loss increases as the interval sizes become larger.

# Example "*MinConf*" problem.

| Rec-ID | Age: 20..29 | Age: 30..39 | Married: Yes | Married: No | NumCars: 0 | NumCars: 1 | NumCars: 2 |
|---|---|---|---|---|---|---|---|
| 100 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 200 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 300 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 400 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 500 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

- The rule

  < NumCars: 0> => <Married: No> has 100% confidence.

# Example "*MinConf*" problem. (con't)

| Rec-ID | Age: 20..29 | Age: 30..39 | Married: Yes | Married: No | NumCars: 0..1 | NumCars: 2..3 |
|---|---|---|---|---|---|---|
| 100 | 1 | 0 | 0 | 1 | 1 | 1 |
| 200 | 1 | 0 | 1 | 0 | 1 | 1 |
| 300 | 1 | 0 | 0 | 1 | 1 | 0 |
| 400 | 0 | 1 | 1 | 0 | 0 | 0 |

- Closest rule is

  <NumCars: 0..1> => <Married: No>

  has only 66.66% confidence.

# “catch-22” Situation.

- Mapping problems  create “catch-22” situation.
  - If the intervals are too large, some rules may not have minimum confidence.
  - If the intervals are too small, some rules may not have minimum support.

# Dynamic discretization based on data distribution (quantitative rules, e.g., Agrawal & Srikant@SIGMOD96)

- Minsupport problem: Adjacent intervals can be combined till minsup is satisfies
- Minconfidence problem: Exploit generalization/specialization, Expected support/expected confidence

# Mining Quantitative Associations

Techniques can be categorized by how numerical attributes, such as age or salary are treated

1. Static discretization based on predefined concept hierarchies (data cube methods)
2. Dynamic discretization based on data distribution (quantitative rules, e.g., Agrawal & Srikant@SIGMOD96)
3. Clustering: Distance-based association (e.g., Yang & Miller@SIGMOD97)
   - One dimensional clustering then association
4. Deviation: (such as Aumann and Lindell@KDD99)

   Sex = female => Wage: mean=$7/hr (overall mean = $9)

# Mining Multi-Dimensional Association

- Single-dimensional rules:

  $buys(X, \text{“milk”}) \Rightarrow buys(X, \text{“bread”})$

- Multi-dimensional rules: $\geq 2$ dimensions or predicates
  - Inter-dimension assoc. rules (*no repeated predicates*)

    $age(X,\text{”19-25”}) \wedge occupation(X,\text{“student”}) \Rightarrow buys(X, \text{“coke”})$
  - hybrid-dimension assoc. rules (*repeated predicates*)

    $age(X,\text{”19-25”}) \wedge buys(X, \text{“popcorn”}) \Rightarrow buys(X, \text{“coke”})$

- Categorical Attributes: finite number of possible values, no ordering among values—data cube approach
- Quantitative Attributes: Numeric, implicit ordering among values—discretization, clustering, and gradient approaches

# Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- **Diverse Frequent Pattern Mining**
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

# Model of Diverse Frequent Patterns

International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-019-00203-2

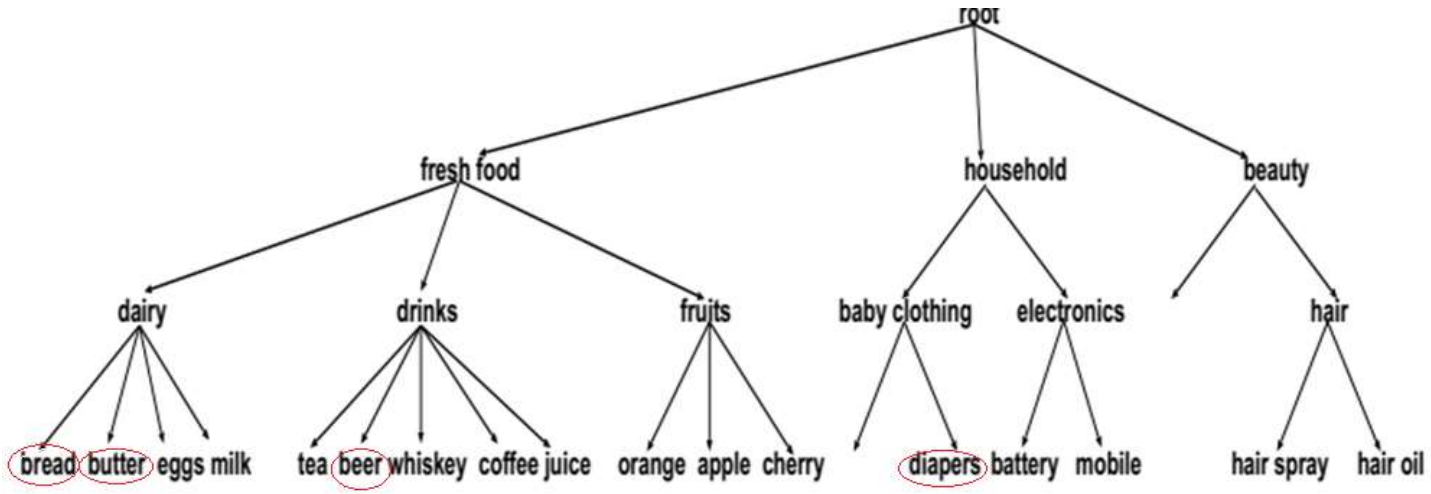
REGULAR PAPER

## A model of concept hierarchy-based diverse patterns with applications to recommender system

M. Kumara Swamy[1] · P. Krishna Reddy[1]

# Notion of Diversity

- Consider patterns, {***bread***, ***butter***} and {***beer***, ***diaper***} with the same *support*
- Normally {***beer***, ***diaper***} is more interesting than {***bread***, ***butter***}
- Reason
  - The items ***bread*** and ***butter*** are belong to same category ***Dairy*** (***Food Product***)
  - The items ***beer*** and ***diaper*** belong to different categories ***drinks*** and ***baby clothing***
- We say that {***beer***, ***diaper***} is more ***diverse*** than {***bread***, ***butter***}



# Notion of Diversity …

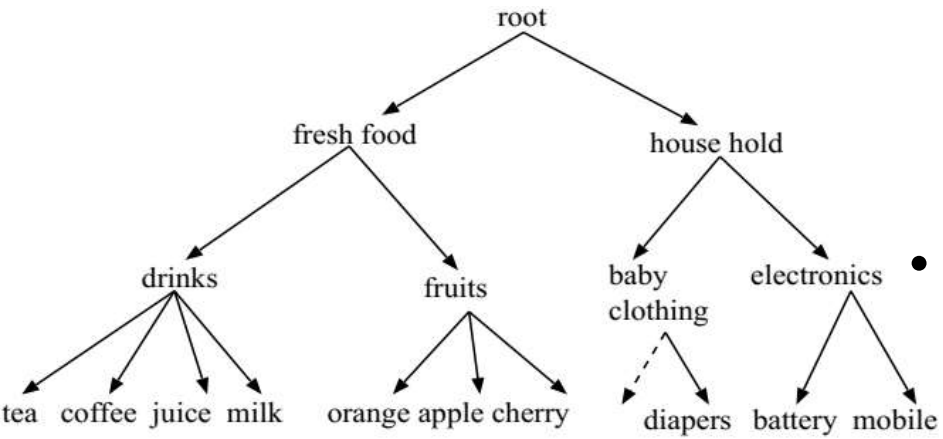
- For certain types of applications, it may be useful to distinguish between the pattern with items belonging to different categories and the pattern with items belonging to few categories.
- The existing pattern extraction approaches (frequent, sequential, periodic, etc.) fail to make such distinction.
- It is possible to rank the patterns by analysing the extent to which the items in the patterns belong to different categories.

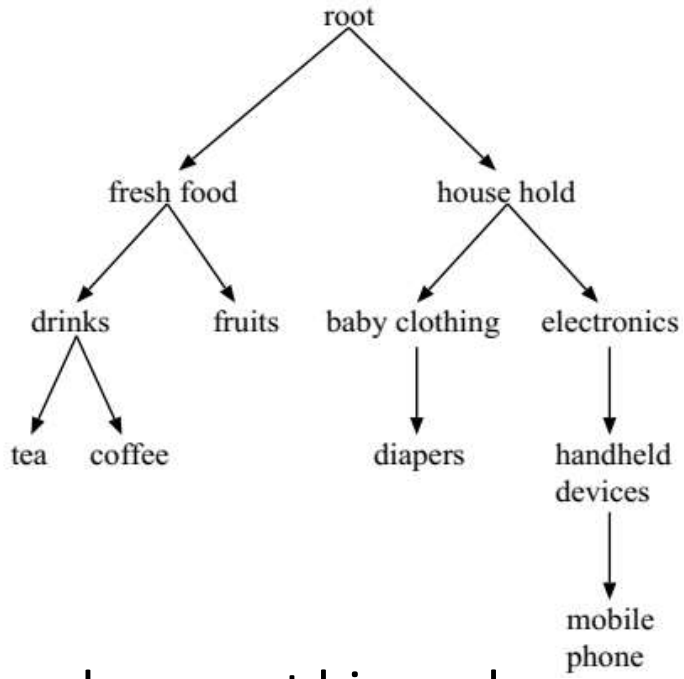
# About Computation of Diversity

- A pattern is an itemset (set of items)
- The diversity value of a pattern is called diverse rank (*drank*)
- To compute the *drank* of a pattern, the following issues are to be resolved
  - To determine the category of items, category level relationship among the items is needed
    - We employ *concept hierarchy* to find the relationship of the items at higher level categories
  - Given a concept hierarchy, a framework is required to compute the *drank* value of the pattern
  - Given a set of items, concept hierarchy, and transactional database, an approach has to be developed to extract *diverse patterns* and *diverse frequent patterns*

# About Balanced and Unbalanced Concept Hierarchy

- Balanced concept hierarchy
  - The depth of all leaf-level items are equal




- Unbalanced concept hierarchy
  - The depth of all leaf-level items are not equal

# Model of Diverse Pattern

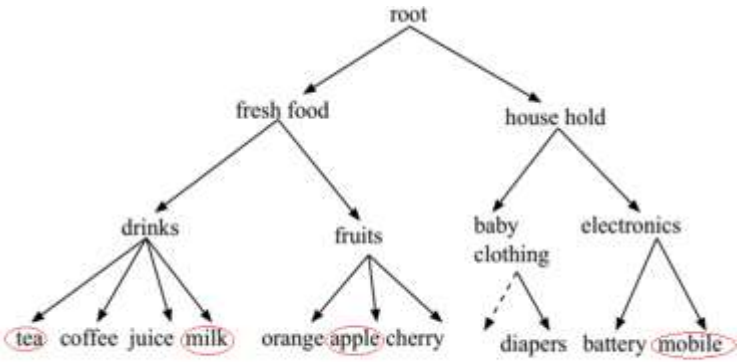
- The diversity of a pattern measures the extent of the items belong to multiple higher level categories.
  - The items of a pattern are mapped to the same/few categories in a concept hierarchy, the pattern has low diversity
  - Relatively, the items of a pattern are mapped to multiple categories, the pattern has more diversity

- Merging behavior: The speed of mapping of items from low level items to high level categories

- The question: how to measure the merging behavior?
  - The pattern merges into one/few higher level categories quickly, and ultimately all paths join at the *root*, the pattern has low diversity value
  - The pattern merges directly at *root* slowly by crossing several intermediary categories, it has relatively high diversity values

# Model of Diverse of Pattern …

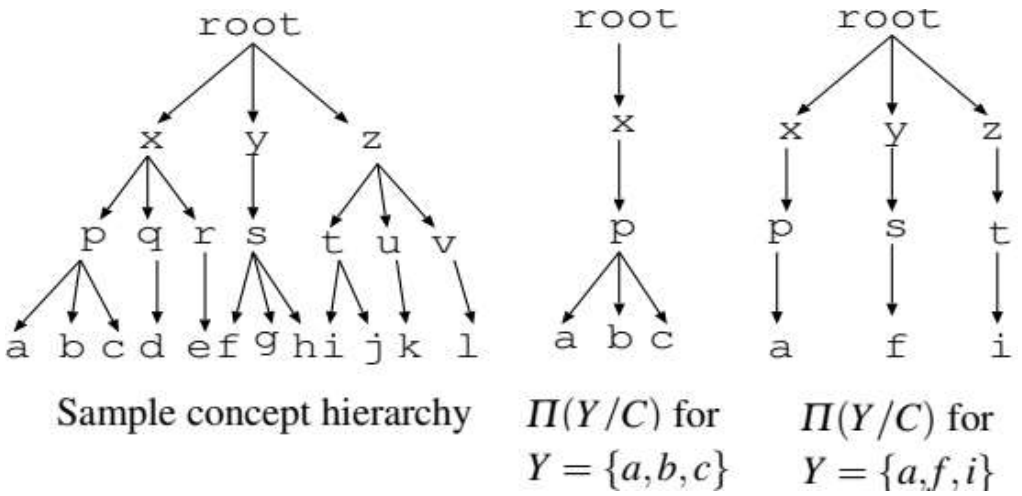
- Example
  - Consider the patterns {*tea*, *milk*}, {*milk*, *apple*} and {*milk*, *mobile*}
  - The pattern {*tea*, *milk*} quickly merges to parent, the diversity of the pattern is low.
  - The pattern {*milk*, *apple*} merges slowly as compared to the pattern {*tea*, *milk*}, the diversity of the pattern is medium.
  - Similarly, the pattern {*milk*, *mobile*} merges slowly at *root* crossing several internal nodes, the diversity of the pattern is high



A balanced concept hierarchy

# Computing the Diversity of a Pattern

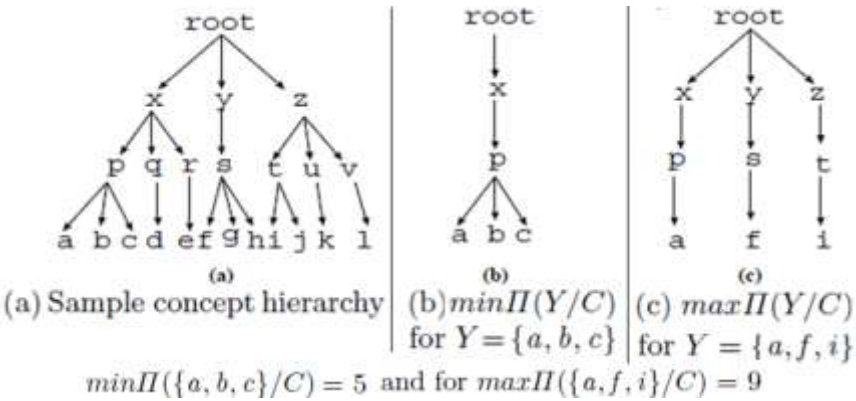
- Terminology
  - Balanced patterns (BP): Let $I = \{i_1, i_2 \cdots , i_n\}$ be a set of items, $D$ be a transactional database on $I$, and $C$ be a concept hierarchy of $I$ with height $h$. A pattern $Y = \{i_1, i_2 \cdots , i_m\} \subseteq I$ with $m$ items is called balanced pattern, if the height of all the items in $Y$ is equal to $h$.
  - Projection of Pattern (Y) on concept hierarchy (C): It is denoted by $\pi(Y/C)$. It is a sub-tree which contains the portion of $C$ concerning to the pattern $Y$. All the nodes and edges exist in the paths of the items of $Y$ to the *root.*




Sample concept hierarchy $\Pi(Y/C)$ for $Y = \{a,b,c\}$ $\Pi(Y/C)$ for $Y = \{a,f,i\}$

# Framework to Compute *drank* using Balanced Concept Hierarchy

- Given the pattern *Y* of certain size and concept hierarchy *C*, two extreme projections are possible
- Minimal projection
  - All the items of *Y* merge at immediate higher level
  - The number of edges $|min\ \pi(Y/C)|$ is equal to $(|Y| + h - 1)$
- Maximal projection
  - All the leaf level nodes merges at *root.*
  - The number of edges $|max\ \pi(Y/C)|$ is equal to $(|Y|\ \text{x}\ h)$
- Observation: Given a pattern Y of certain size
  - Its *minimal projection* contains minimum number of edges and *maximal project* contains the maximum number of edges



(a) Sample concept hierarchy (b) $minΠ(Y/C)$ for $Y = \{a, b, c\}$ (c) $maxΠ(Y/C)$ for $Y = \{a, f, i\}$

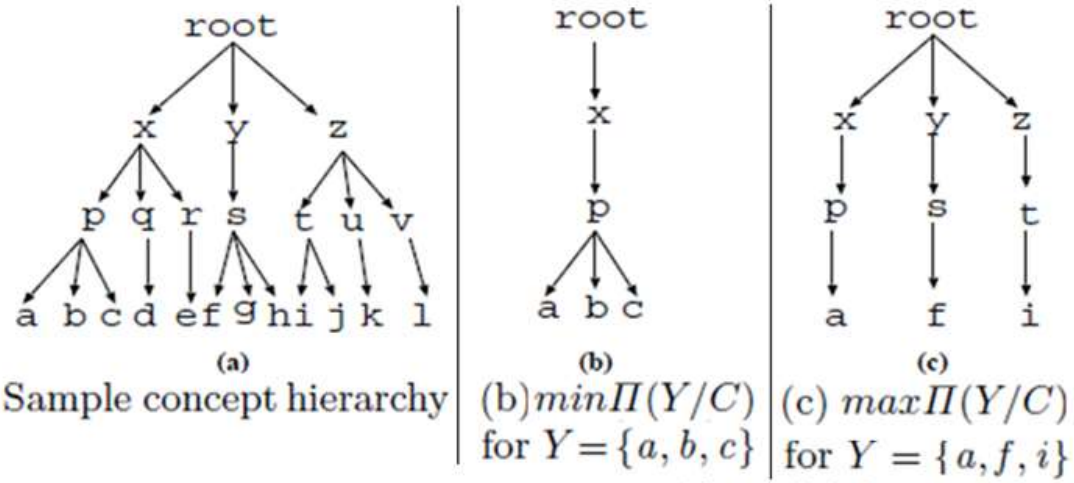
$minΠ(\{a, b, c\}/C) = 5$ and for $maxΠ(\{a, f, i\}/C) = 9$

# Framework to Compute *drank* using Balanced Concept Hierarchy …

- Given a pattern Y, the *drank*(*Y*) is the ratio of number of edges in its projection ($\pi(Y/C)$) and the maximal projection ($max\ \pi(Y/C)$) which is equal to $\frac{|\Pi(Y/C)|}{|max\Pi(Y/C)|}$
- The minimum value of this ratio is equal to $\frac{|min\Pi(Y/C)|}{|max\Pi(Y/C)|}$ which is equal to 0.
- The maximum value of this ratio is equal to $\frac{|max\Pi(Y/C)|}{|max\Pi(Y/C)|}$ which is equal to 1.
- On applying the min-max normalization, we get following formula. $drank(Y) = \frac{(|\Pi(Y/C)|) - (|Y| + h - 1)}{(h-1)(|Y|-1)}$

$$drank(Y) = \left( \frac{\left(\frac{|\Pi(Y/C)|}{|Y|\times h}\right) - \left(\frac{|Y|+h-1}{|Y|\times h}\right)}{1 - \left(\frac{|Y|+h-1}{|Y|\times h}\right)} \right)(1-0) + 0$$

# Example




(a) Sample concept hierarchy | (b) $minΠ(Y/C)$ for $Y=\{a,b,c\}$ | (c) $maxΠ(Y/C)$ for $Y=\{a,f,i\}$

$minΠ(\{a,b,c\}/C)=5$ and for $maxΠ(\{a,f,i\}/C)=9$

$$drank(Y)=\frac{(|Π(Y/C)|)-(|Y|+h-1)}{(h-1)(|Y|-1)}$$

$drank(\{a,b,c\})=\frac{5-5}{2\times 2}=0.0$

$drank(\{a,b,e\})=0.25$

$drank(\{a,b,f\})=0.5$

$drank(\{a,e,f\})=0.75$

$drank(\{a,f,i\})=1.0$

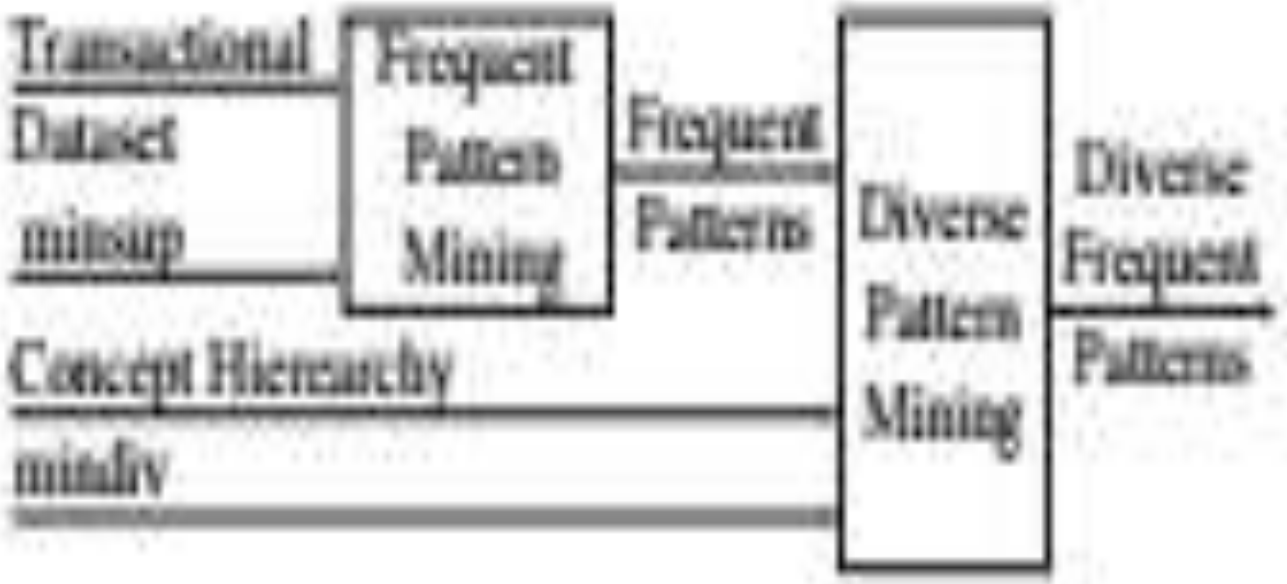



Fig. 3 Extraction of diverse frequent patterns

## Experiments

- Dataset from MovieLens project (http://www.grouplens.org)
  : contains 100,000 ratings of 943 users on 1682 movies

Table 1 Top-10 patterns of size 3 sorted w.r.t. *drank*

| Top 10 diverse patterns | *drank* | *Support* (%) |
|---|---|---|
| {G.I. Jane, The Game, Contact} | 1 | 5.5 |
| {G.I. Jane, Titanic, Contact} | 1 | 5.3 |
| {Amistad, L.A. Confidential, Titanic} | 1 | 5.3 |
| {One Fine Day, Return of the Jedi, Star Wars} | 1 | 5.1 |
| {One Fine Day, Star Wars, Return of the Jedi} | 1 | 5.1 |
| {The First Wives Club, Return of the Jedi, Star Wars} | 1 | 5.1 |
| {Volcano, Star Wars, Return of the Jedi} | 1 | 5.0 |
| {Welcome to the Dollhouse, Trainspotting, Star Wars} | 1 | 4.8 |
| {Austin Powers: International Man of Mystery, Fargo, Star Wars} | 1 | 4.6 |
| {Austin Powers: Int'l. Man of Mystery, Return of the Jedi, Star Wars} | 1 | 4.3 |

Table 2 Top 10 patterns of size 3 sorted w.r.t. support value

| Top 10 frequent patterns | *Support* (%) | *drank* |
|---|---|---|
| {Raiders of the Lost Ark, Return of the Jedi, Star Wars} | 23.2 | 0.33 |
| {The Empire Strikes Back, Raiders of the Lost Ark, Star Wars} | 20.6 | 0.33 |
| {The Rock, Return of the Jedi, Star Wars} | 17.7 | 0.33 |
| {Independence Day, Return of the Jedi, Star Wars} | 18.2 | 0.33 |
| {Aliens, Star Wars, Raiders of the Lost Ark} | 16.1 | 0.75 |
| {Braveheart, Indiana Jones and the Last Crusade, Return of the Jedi} | 14.9 | 0.33 |
| {Aliens, Return of the Jedi, Star Wars} | 14.8 | 0.33 |
| {The Terminator, Pulp Fiction, Raiders of the Lost Ark} | 14.8 | 0.33 |
| {The Princess Bride, The Empire Strikes Back, Raiders of the Lost Ark} | 14.5 | 0.33 |
| {Independence Day, Raiders of the Lost Ark, Return of the Jedi} | 14.4 | 0.33 |

# Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Diverse Frequent Pattern Mining
- **High Utility Pattern mining**
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

# HIGH UTILITY MINING

**A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets**

Ying Liu, Wei-keng Liao, and Alok Choudhary

PAKDD 2005

# Limitations of Frequent Itemset Mining

- Purchase quantities are not taken into account.
  - Thus, an item may only appear once or zero time in a transaction. Thus, if a customer has bought five breads, ten breads or twenty breads, it is viewed as the same.
- All items are viewed as having the same importance, utility of weight.
  - For example, if a customer buys a very expensive bottle of wine or just a piece of bread, it is viewed as being equally important.
- Frequent pattern mining may find many frequent patterns that are not interesting.
  - For example, one may find that {bread, milk} is a frequent pattern. However, from a business perspective, this pattern may be uninteresting because it does not generate much profit.
  - Moreover, frequent pattern mining algorithms may miss the rare patterns that generate a high profit such as perhaps {caviar, wine}

# High-utility itemset mining

- **High-utility itemset mining addresses the limitations** .
- A transaction database contains transactions where purchase quantities are taken into account as well as the unit profit of each item. For example, consider the following transaction database.
- Local transaction utility of an item
- External utility of an item
- Utility of the item in a transaction= Local utility * external utility

**Table 1.** A transaction database

(a) Transaction table. Each row is a transaction. The columns represent the number of items in a particular transaction. TID is the transaction identification number

| TID \ ITEM | A | B | C | D | E |
|---|---|---|---|---|---|
| $T_1$ | 0 | 0 | 18 | 0 | 1 |
| $T_2$ | 0 | 6 | 0 | 1 | 1 |
| $T_3$ | 2 | 0 | 1 | 0 | 1 |
| $T_4$ | 1 | 0 | 0 | 1 | 1 |
| $T_5$ | 0 | 0 | 4 | 0 | 2 |
| $T_6$ | 1 | 1 | 0 | 0 | 0 |
| $T_7$ | 0 | 10 | 0 | 1 | 1 |
| $T_8$ | 3 | 0 | 25 | 3 | 1 |
| $T_9$ | 1 | 1 | 0 | 0 | 0 |
| $T_{10}$ | 0 | 6 | 2 | 0 | 2 |

(b) The utility table. The right column displays the profit of each item per unit in dollars

| ITEM | PROFIT ($) (per unit) |
|---|---|
| A | 3 |
| B | 10 |
| C | 1 |
| D | 6 |
| E | 5 |

(c) Transaction utility (TU) of the transaction database

| TID | TU | TID | TU |
|---|---|---|---|
| $T_1$ | 23 | $T_6$ | 13 |
| $T_2$ | 71 | $T_7$ | 111 |
| $T_3$ | 12 | $T_8$ | 57 |
| $T_4$ | 14 | $T_9$ | 13 |
| $T_5$ | 14 | $T_{10}$ | 72 |

# Utility Mining Problem

- Utility mining is to find all the itemsets whose utility values are beyond a user-specified threshold. An itemset X is a high utility itemset if $u(X) \geq \varepsilon$, where $X \subseteq I$ and $\varepsilon$ is the minimum utility threshold, otherwise, it is a low utility itemset.
- Non-existence of "downward closure property" (anti-monotone property) in the *utility mining* model

- $I = \{i_1, i_2, \ldots, i_m\}$ is a set of items.
- $D = \{T_1, T_2, \ldots, T_n\}$ be a transaction database where each transaction $T_i \in D$ is a subset of $I$.
- $o(i_p, T_q)$, *local transaction utility value*, represents the quantity of item $i_p$ in transaction $T_q$. For example, $o(\text{A}, T_8) = 3$, in Table 1(a).
- $s(i_p)$, *external utility,* is the value associated with item $i_p$ in the Utility Table. This value reflects the importance of an item, which is independent of transactions. For example, in Table 1(b), the external utility of item A, $s(\text{A})$, is 3.
- $u(i_p, T_q)$, *utility*, the quantitative measure of utility for item $i_p$ in transaction $T_q$, is defined as $o(ip,Tq) \times s(ip)$. For example, $u(A, T_8) = 3 \times 3 = 9$, in Table 1
- $u(X, T_q)$, *utility of an itemset X in transaction* $T_q$, is defined as $\sum_{ip \in X} u(ip,Tq)$, where

  $X = \{i_1, i_2, \ldots, i_k\}$ is a $k$-itemset, $X \subseteq T_q$ and $1 \leq k \leq m$.
- $u(X)$, *utility of an itemset X*, is defined as $\sum_{T_q \in D \wedge X \subseteq T_q} u(X,T_q)$. (3.1)

# Two-Phase algorithm

- Transaction utility= sum of all utilities of items
- Transaction weighted utility of itemset X, twu(X)= Summation of utilities all transactions Ti such that X is belongs to Ti.
- High Transaction-weighted Utilization Itemset: Transaction weighted Utility of the itemset should be greater than the user given threshold.
- Transaction-weighted Downward Closure Property: If k-itemset is high transaction-weighted utilization itemset, k-1 itemset will be high transaction-weighted utilization itemset.
- Theorem: Let HTWU be the collection of all high transaction-weighted utilization itemsets in a transaction database D, and HU be the collection of high utility itemsets in D. Then HU is the subset of HTWU.

**Theorem 1. (Transaction-weighted Downward Closure Property)** Let $I^k$ be a $k$-itemset and $I^{k-1}$ be a $(k-1)$-itemset such that $I^{k-1} \subset I^k$. If $I^k$ is a high transaction-weighted utilization itemset, $I^{k-1}$ is a high transaction-weighted utilization itemset.

**Proof:** Let $T_{I^k}$ be the collection of the transactions containing $I^k$ and $T_{I^{k-1}}$ be the collection containing $I^{k-1}$. Since $I^{k-1} \subset I^k$, $T_{I^{k-1}}$ is a superset of $T_{I^k}$. According to Definition 2, $twu(I^{k-1}) = \sum_{I^{k-1} \subseteq T_q \in D} tu(T_q) \geq \sum_{I^k \subseteq T_p \in D} tu(T_p) = twu(I^k) \geq \varepsilon'$ ❑

The *Transaction-weighted Downward Closure Property* indicates that any superset of a low transaction-weighted utilization itemset is low in transaction-weighted utilization. That is, only the combinations of high transaction-weighted utilization $(k-1)$-itemsets could be added into the candidate set $C_k$ at each level.

**Theorem 2.** Let *HTWU* be the collection of all high transaction-weighted utilization itemsets in a transaction database *D*, and *HU* be the collection of high utility itemsets in *D*. If $\varepsilon' = \varepsilon$, then $HU \subseteq HTWU$.

**Proof:** $\forall X \in HU$, if $X$ is a high utility itemset, then

$$\varepsilon' = \varepsilon \leq u(X) = \sum_{X \subseteq T_q} u(X, T_q) = \sum_{X \subseteq T_q} \sum_{i_p \in X} u(i_p, T_q) \leq \sum_{X \subseteq T_q} \sum_{i_p \in T_q} u(i_p, T_q) = \sum_{X \subseteq T_q} tu(T_q) = twu(X)$$

Thus, $X$ is a high transaction-weighted utilization itemset and $X \in HTWU$. ❑

According to Theorem 2, we can utilize the *Transaction-weighted Downward Closure Property* in our *transaction-weighted utilization mining* in Phase I by assuming $\varepsilon' = \varepsilon$ and prune those overestimated itemsets in Phase II.

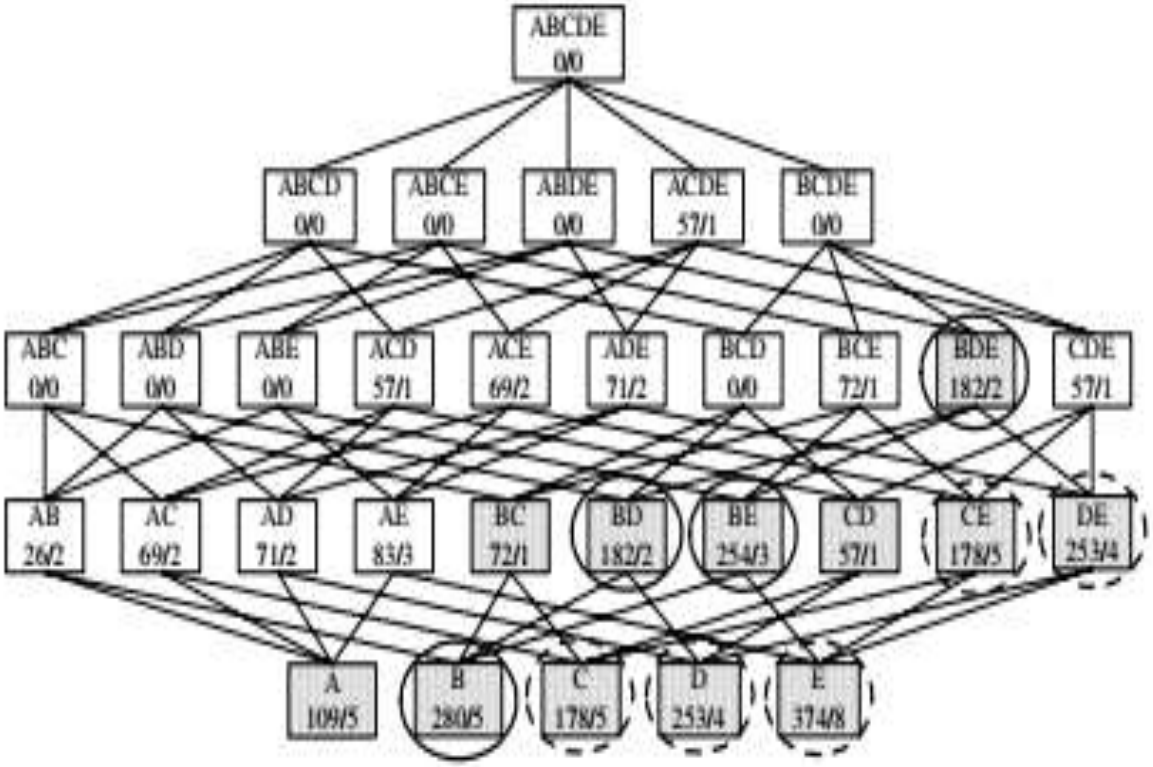



**Fig. 1.** Itemsets lattice related to the example in Table 1. $\varepsilon' = 120$. Itemsets in circles (solid and dashed) are the high transaction-weighted utilization itemsets in *transaction-weighted utilization mining model*. Gray-shaded boxes denote the search space. Itemsets in solid circles are high utility itemsets. Numbers in each box are transaction-weighted utilization / number of occurrence

# Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- **Mining Rare Patterns and Negative Patterns**
- Constraint-Based Frequent Pattern Mining
- Summary

# Negative and Rare Patterns

- Rare patterns: Very low support but interesting
  - E.g., buying Rolex watches
  - Mining: Setting individual-based or special group-based support threshold for valuable items
- Negative patterns
  - Since it is unlikely that one buys Ford Expedition (an SUV car) and Toyota Prius (a hybrid car) together, Ford Expedition and Toyota Prius are likely negatively correlated patterns
- Negatively correlated patterns that are infrequent tend to be more interesting than those that are frequent

# Defining Negative Correlated Patterns (I)

- Definition 1 (support-based)
  - If itemsets X and Y are both frequent but rarely occur together, i.e.,

    $sup(X \cup Y) < sup(X) * sup(Y)$
  - Then X and Y are negatively correlated
- Problem: A store sold two needle 100 packages A and B, only one transaction containing both A and B.
  - When there are in total 200 transactions, we have

    $s(A \cup B) = 0.005$, $s(A) * s(B) = 0.25$, $s(A \cup B) < s(A) * s(B)$
  - When there are $10^5$ transactions, we have

    $s(A \cup B) = 1/10^5$, $s(A) * s(B) = 1/10^3 * 1/10^3$, $s(A \cup B) > s(A) * s(B)$
  - Where is the problem? —Null transactions, i.e., the support-based definition is not null-invariant!

# Defining Negative Correlated Patterns (II)

- Definition 2 (negative itemset-based)
  - X is a *negative itemset* if (1) X = $\bar{A}$ U B, where B is a set of positive items, and $\bar{A}$ is a set of negative items, $|\bar{A}| \geq 1$, and (2) $s(X) \geq \mu$
  - Itemsets X is negatively correlated, if

$$s(X) < \prod_{i=1}^{k} s(x_i), where\ x_i \in X,\ and\ s(x_i)\ is\ the\ support\ of\ x_i$$

- This definition suffers a similar null-invariant problem
- Definition 3 (Kulzynski measure-based) If itemsets X and Y are frequent, but $(P(X|Y) + P(Y|X))/2 < \epsilon$, where $\epsilon$ is a negative pattern threshold, then X and Y are negatively correlated.
- Ex. For the same needle package problem, when no matter there are 200 or $10^5$ transactions, if $\epsilon = 0.01$, we have

$$(P(A|B) + P(B|A))/2 = (0.01 + 0.01)/2 < \epsilon$$

# Mining Compressed Patterns

| Pat-ID | Item-Sets | Support |
|---|---|---|
| P1 | {38,16,18,12} | 205227 |
| P2 | {38,16,18,12,17} | 205211 |
| P3 | {39,38,16,18,12,17} | 101758 |
| P4 | {39,16,18,12,17} | 161563 |

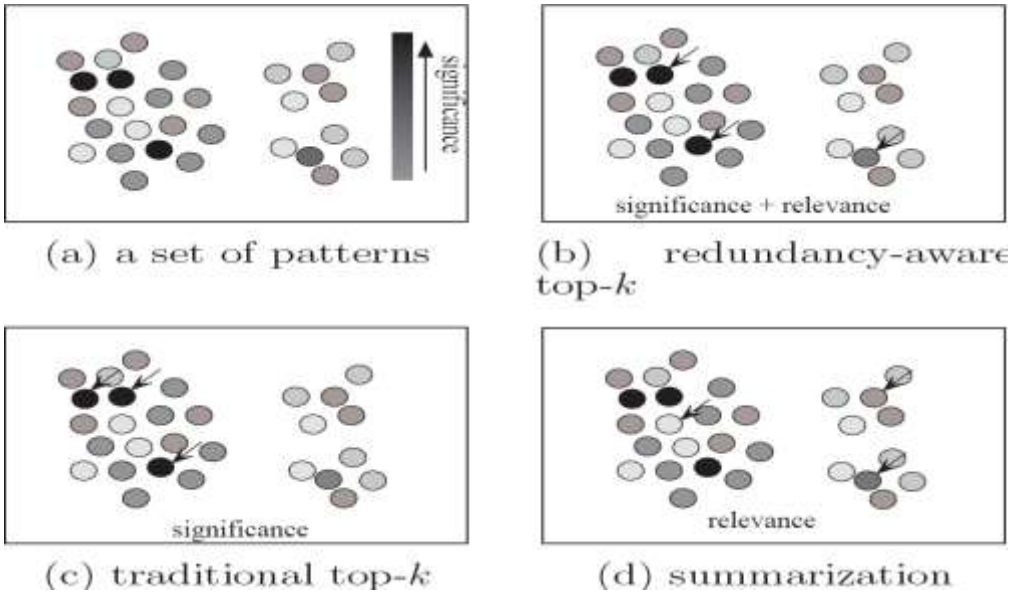
- Closed patterns
  - P1, P2, P3, P4, P5
  - Emphasizes too much on support
- Max-patterns
  - P3: information loss
- Desired output (a good balance):
  - P2, P3, P4

- Why mining compressed patterns? Too many scattered patterns but not so meaningful
- Pa $Dist(P_1, P_2) = 1 - \frac{|T(P_1) \cap T(P_2)|}{|T(P_1) \cup T(P_2)|}$

- δ-clustering: For each pattern P, find all patterns which can be expressed by P and whose distance to P is within δ (δ-cover)
- All patterns in the cluster can be represented by P

# Redundancy-Aware Top-k Patterns

- Desired patterns: high significance & low redundancy




(a) a set of patterns (b) redundancy-aware top-$k$

(c) traditional top-$k$ (d) summarization

- Method: Use MMS (Maximal Marginal Significance) for measuring the combined significance of a pattern set
- Xin et al., Extracting Redundancy-Aware Top-K Patterns, KDD'06

# Outline

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- **Constraint-Based Frequent Pattern Mining**
- Summary

# Constraint-based (Query-Directed) Mining

- Finding all the patterns in a database autonomously? — unrealistic!
  - The patterns could be too many but not focused!
- Data mining should be an interactive process
  - User directs what to be mined using a data mining query language (or a graphical user interface)
- Constraint-based mining
  - User flexibility: provides constraints on what to be mined
  - Optimization: explores such constraints for efficient mining — constraint-based mining: constraint-pushing, similar to push selection first in DB query processing
  - Note: still find all the answers satisfying constraints, not finding some answers in “heuristic search”

# Constraints in Data Mining

- Knowledge type constraint:
  - classification, association, etc.
- Data constraint — using SQL-like queries
  - find product pairs sold together in stores in Chicago this year
- Dimension/level constraint
  - in relevance to region, price, brand, customer category
- Rule (or pattern) constraint
  - small sales (price < $10) triggers big sales (sum > $200)
- Interestingness constraint
  - strong rules: min_support $\geq$ 3%, min_confidence $\geq$ 60%

# Meta-Rule Guided Mining

- Meta-rule can be in the rule form with partially instantiated predicates and constants

  $P_1(X, Y) \wedge P_2(X, W) \Rightarrow buys(X, \text{“iPad”})$

- The resulting rule derived can be

  $age(X, \text{“15-25”}) \wedge profession(X, \text{“student”}) \Rightarrow buys(X, \text{“iPad”})$

- In general, it can be in the form of

  $P_1 \wedge P_2 \wedge \ldots \wedge P_l \Rightarrow Q_1 \wedge Q_2 \wedge \ldots \wedge Q_r$

- Method to find meta-rules
  - Find frequent (l+r) predicates (based on min-support threshold)
  - Push constants deeply when possible into the mining process (see the remaining discussions on constraint-push techniques)
  - Use confidence, correlation, and other measures when possible

# Constraint-Based Frequent Pattern Mining

- Pattern space pruning constraints
  - Anti-monotonic: If constraint c is violated, its further mining can be terminated
  - Monotonic: If c is satisfied, no need to check c again
  - Succinct: c must be satisfied, so one can start with the data sets satisfying c
  - Convertible: c is not monotonic nor anti-monotonic, but it can be converted into it if items in the transaction can be properly ordered
- Data space pruning constraint
  - Data succinct: Data space can be pruned at the initial pattern mining process
  - Data anti-monotonic: If a transaction t does not satisfy c, t can be pruned from its further mining

# What Constraints Are Convertible?

| Constraint | Convertible anti-monotone | Convertible monotone | Strongly convertible |
|---|---|---|---|
| avg(S) ≤ , ≥ v | Yes | Yes | Yes |
| median(S) ≤ , ≥ v | Yes | Yes | Yes |
| sum(S) ≤ v (items could be of any value, v ≥ 0) | Yes | No | No |
| sum(S) ≤ v (items could be of any value, v ≤ 0) | No | Yes | No |
| sum(S) ≥ v (items could be of any value, v ≥ 0) | No | Yes | No |
| sum(S) ≥ v (items could be of any value, v ≤ 0) | Yes | No | No |
| …… | | | |

# Constraint-Based Mining — A General Picture

| Constraint | Anti-monotone | Monotone | Succinct |
|---|---|---|---|
| $v \in S$ | no | yes | yes |
| $S \supseteq V$ | no | yes | yes |
| $S \subseteq V$ | yes | no | yes |
| $\min(S) \leq v$ | no | yes | yes |
| $\min(S) \geq v$ | yes | no | yes |
| $\max(S) \leq v$ | yes | no | yes |
| $\max(S) \geq v$ | no | yes | yes |
| $\text{count}(S) \leq v$ | yes | no | weakly |
| $\text{count}(S) \geq v$ | no | yes | weakly |
| $\text{sum}(S) \leq v$ ( $a \in S, a \geq 0$ ) | yes | no | no |
| $\text{sum}(S) \geq v$ ( $a \in S, a \geq 0$ ) | no | yes | no |
| $\text{range}(S) \leq v$ | yes | no | no |
| $\text{range}(S) \geq v$ | no | yes | no |
| $\text{avg}(S)\ \theta\ v, \theta \in \{=, \leq, \geq\}$ | convertible | convertible | no |
| $\text{support}(S) \geq \xi$ | yes | no | no |
| $\text{support}(S) \leq \xi$ | no | yes | no |

# Summary

- Pattern Mining: A Road Map
- Coverage Pattern Mining
- Multilevel Pattern Mining
- Quantitative Pattern Mining
- Diverse Frequent Pattern Mining
- High Utility Pattern mining
- Mining Rare Patterns and Negative Patterns
- Constraint-Based Frequent Pattern Mining
- Summary

# Summary: Pattern Mining: Advanced Methods (I)

- Mining Diverse Patterns
  - Mining Multiple-Level Associations
  - Mining Multi-Dimensional Associations
  - Mining Quantitative Associations
  - Mining Negative Correlations
  - Mining Compressed and Redundancy-Aware Patterns

- Constraint-Based Frequent Pattern Mining
  - Why Constraint-Based Mining?
  - Constrained Mining with Pattern Anti-Monotonicity
  - Constrained Mining with Pattern Monotonicity
  - Constrained Mining with Data Anti-Monotonicity
  - Constrained Mining with Succinct Constraints
  - Constrained Mining with Convertible Constraints
  - Handling Multiple Constraints
  - Constraint-Based Sequential-Pattern Mining

# Summary: Pattern Mining: Advanced Methods (II)

- Sequential Pattern Mining
  - Sequential Pattern and Sequential Pattern Mining
  - GSP: Apriori-Based Sequential Pattern Mining
  - SPADE: Sequential Pattern Mining in Vertical Data Format
  - PrefixSpan: Sequential Pattern Mining by Pattern-Growth
  - CloSpan: Mining Closed Sequential Patterns
  - Constraint-Based Sequential Pattern Mining

- Graph Pattern Mining
  - Graph Pattern and Graph Pattern Mining
  - Apriori-Based Graph Pattern Mining Methods
  - gSpan: A Pattern-Growth-Based Method
- CloseGraph: Mining Closed Graph Patterns
  - Graph Pattern Applications
- Application I: Mining Software Copy-and-Paste Bugs
- Application II: Phrase Mining

# References: Mining Diverse Patterns


- R. Srikant and R. Agrawal, “Mining generalized association rules”, VLDB'95
- Y. Aumann and Y. Lindell, “A Statistical Theory for Quantitative Association Rules”, KDD'99
- K. Wang, Y. He, J. Han, “Pushing Support Constraints Into Association Rules Mining”, IEEE Trans. Knowledge and Data Eng. 15(3): 642-658, 2003
- D. Xin, J. Han, X. Yan and H. Cheng, "On Compressing Frequent Patterns", Knowledge and Data Engineering, 60(1): 5-29, 2007
- D. Xin, H. Cheng, X. Yan, and J. Han, "Extracting Redundancy-Aware Top-K Patterns", KDD'06
- J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent Pattern Mining: Current Status and Future Directions", Data Mining and Knowledge Discovery, 15(1): 55-86, 2007
- F. Zhu, X. Yan, J. Han, P. S. Yu, and H. Cheng, “Mining Colossal Frequent Patterns by Core Pattern Fusion”, ICDE'07

# References: Constraint-Based Frequent Pattern Mining


- R. Srikant, Q. Vu, and R. Agrawal, “Mining association rules with item constraints”, KDD'97
- R. Ng, L.V.S. Lakshmanan, J. Han & A. Pang, “Exploratory mining and pruning optimizations of constrained association rules”, SIGMOD’98
- G. Grahne, L. Lakshmanan, and X. Wang, “Efficient mining of constrained correlated sets”, ICDE'00
- J. Pei, J. Han, and L. V. S. Lakshmanan, “Mining Frequent Itemsets with Convertible Constraints”, ICDE'01
- J. Pei, J. Han, and W. Wang, “Mining Sequential Patterns with Constraints in Large Databases”, CIKM'02
- F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi, “ExAnte: Anticipated Data Reduction in Constrained Pattern Mining”, PKDD'03
- F. Zhu, X. Yan, J. Han, and P. S. Yu, “gPrune: A Constraint Pushing Framework for Graph Pattern Mining”, PAKDD'07

# References: Sequential Pattern Mining


- R. Srikant and R. Agrawal, “Mining sequential patterns: Generalizations and performance improvements”, EDBT’96
- M. Zaki, “SPADE: An Efficient Algorithm for Mining Frequent Sequences”, Machine Learning, 2001
- J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach", IEEE TKDE, 16(10), 2004
- X. Yan, J. Han, and R. Afshar, “CloSpan: Mining Closed Sequential Patterns in Large Datasets”, SDM'03
- J. Pei, J. Han, and W. Wang, "Constraint-based sequential pattern mining: the pattern-growth methods", J. Int. Inf. Sys., 28(2), 2007
- M. N. Garofalakis, R. Rastogi, K. Shim: Mining Sequential Patterns with Regular Expression Constraints. IEEE Trans. Knowl. Data Eng. 14(3), 2002
- H. Mannila, H. Toivonen, and A. I. Verkamo, “Discovery of frequent episodes in event sequences”, Data Mining and Knowledge Discovery, 1997

# References: Graph Pattern Mining


- C. Borgelt and M. R. Berthold, Mining molecular fragments: Finding relevant substructures of molecules, ICDM'02
- J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism, ICDM'03
- A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data, PKDD'00
- M. Kuramochi and G. Karypis. Frequent subgraph discovery, ICDM'01
- S. Nijssen and J. Kok. A Quickstart in Frequent Structure Mining can Make a Difference. KDD'04
- N. Vanetik, E. Gudes, and S. E. Shimony. Computing frequent graph patterns from semistructured data, ICDM'02
- X. Yan and J. Han, gSpan: Graph-Based Substructure Pattern Mining, ICDM'02
- X. Yan and J. Han, CloseGraph: Mining Closed Frequent Graph Patterns, KDD'03
- X. Yan, P. S. Yu, J. Han, Graph Indexing: A Frequent Structure-based Approach, SIGMOD'04
- X. Yan, P. S. Yu, and J. Han, Substructure Similarity Search in Graph Databases, SIGMOD'05

# References: Phrase Mining


- S. Bergsma, E. Pitler, D. Lin, Creating Robust Supervised Classifiers via Web-scale N-gram Data, ACL'2010
- D. M. Blei and J. D. Lafferty. Visualizing Topics with Multi-word Expressions. arXiv:0907.1013, 2009
- D.M. Blei, A. Y. Ng, M. I. Jordan, J. D. Lafferty, Latent Dirichlet Allocation. JMLR 2003
- M. Danilevsky, C. Wang, N. Desai, X. Ren, J. Guo, J. Han. Automatic Construction and Ranking of Topical Keyphrases on Collections of Short Documents. SDM'14
- A. El-Kishky, Y. Song, C. Wang, C. R. Voss, and J. Han. Scalable Topical Phrase Mining from Text Corpora. VLDB'15
- R. V. Lindsey, W. P. Headden, III, M. J. Stipicevic. A Phrase-Discovering Topic Model Using Hierarchical Pitman-Yor Processes. EMNLP-CoNLL'12.
- J. Liu, J. Shang, C. Wang, X. Ren, J. Han, Mining Quality Phrases from Massive Text Corpora. SIGMOD'15
- A. Parameswaran, H. Garcia-Molina, and A. Rajaraman. Towards the Web of Concepts: Extracting Concepts from Large Datasets. VLDB'10
- X. Wang, A. McCallum, X. Wei. Topical N-grams: Phrase and Topic Discovery, With and Application to Information Retrieval. ICDM'07
- J. Shang, J. Liu, M. Jiang, X. Ren, C. R Voss, J. Han, "Automated Phrase Mining from Massive Text Corpora", IEEE Transactions on Knowledge and Data Engineering, 30(10):1825-1837 (2018)

# Classification: Basic Concepts
# Class #16

P. Krishna Reddy, IIIT Hyderabad

# Topics

- Introduction (1.5 hour): Definition, KDD framework, Issues in data mining.
- Data summarization (7.5 hrs): Data Types, Preprocessing, Characterization, Discrimination, data warehousing techniques (Multidimensional data model, Data warehousing architecture, Data cube computation and OLAP technology)
- Concepts and algorithms for mining patterns and associations (9 hours) (Frequent item-set generation, A priori and FP-growth algorithm, Evaluation of Association patterns) and preprocessing
- **Concepts and algorithms related to classification and regression (9hrs) (Overview, Decision tree induction, Over-fitting and under-fitting, Scalable decision tree algorithms, Bayesian Classification, Regression-based Prediction methods (9 hours)**
- Concepts and algorithms for clustering the data (9 hours) (Overview, Types of Data, K-means, Aglomerative clustering, Clustering algorithms (DBSCAN, BIRCH, CURE, ROCK, CHAMELEON)).
- Outlier analysis and future trends (graph mining, spatio-temporal mining). (3 hours)

# Presentation Outline

- **Background**
- Basic concepts
- Decision Tree Induction
- Bayes Classification Methods
- Lazy Learners (k-nearest neighbours)
- Linear Classifiers
- Model Evaluation and Selection
- Techniques to Improve Accuracy
- Summary

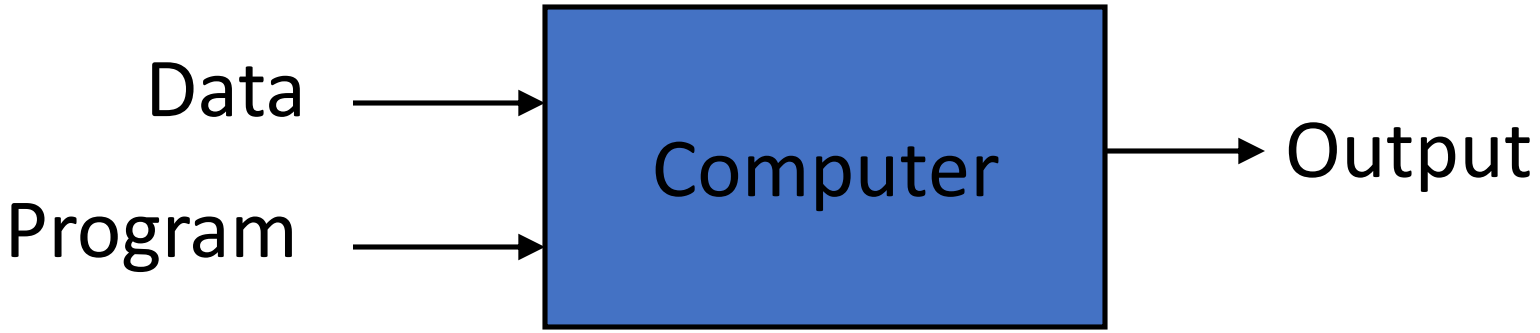
# A simple question

- 1, 3, 5, 7, 9, ….  What is the next number?
  - Ans: 11. ;  Odd numbers, Or 2n+1
- 1, 3, 9, 19, 33, ... What is the next number?
  - Ans: 51; $2n^2+1$
- How do we solve such problems?
  - Find a pattern from the examples.
    - (function f(n) = 2n+1. Or model the data)
  - Use it to predict the next number (or solve the problem)
- How do we design a computational procedure?

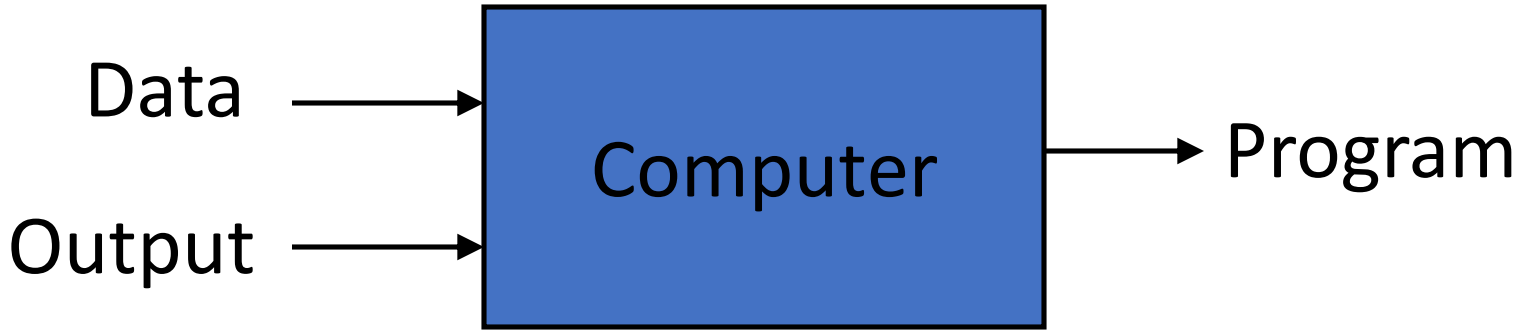
# A simple question (cont.)

- We know: 1, 3, 9, 19, 33, ... What is the next number?
  - Ans: 51; $2n^2+1$
- 0.99, 3.02, 9.00, 18.98, 33.01, ... What next?
- Consider a series of 2D points
  - (1,3), (2,6), (3,9), (4,12), ....
  - What is the next point?
- When does the problem become difficult?
  - When numbers are “uncertain”. Noise in measurements
  - When numbers are not just “simple numbers”?

## Traditional Programming




## Machine Learning



# The machine learning framework

- Apply a prediction function to a feature representation of the “sample” to get the desired output:

f(  ) = “apple”

f(  ) = “tomato”

f(  ) = “cow”



Slide credit: L. Lazebnik

# The machine learning/Classification framework

$$y = f(\mathbf{x})$$


- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function $f$ by minimizing the prediction error.

- **Testing:** apply $f$ to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$


Slide credit: L. Lazebnik

# Steps

## Training

Training Data

Training Labels

Features

Training

Learned model

## Testing

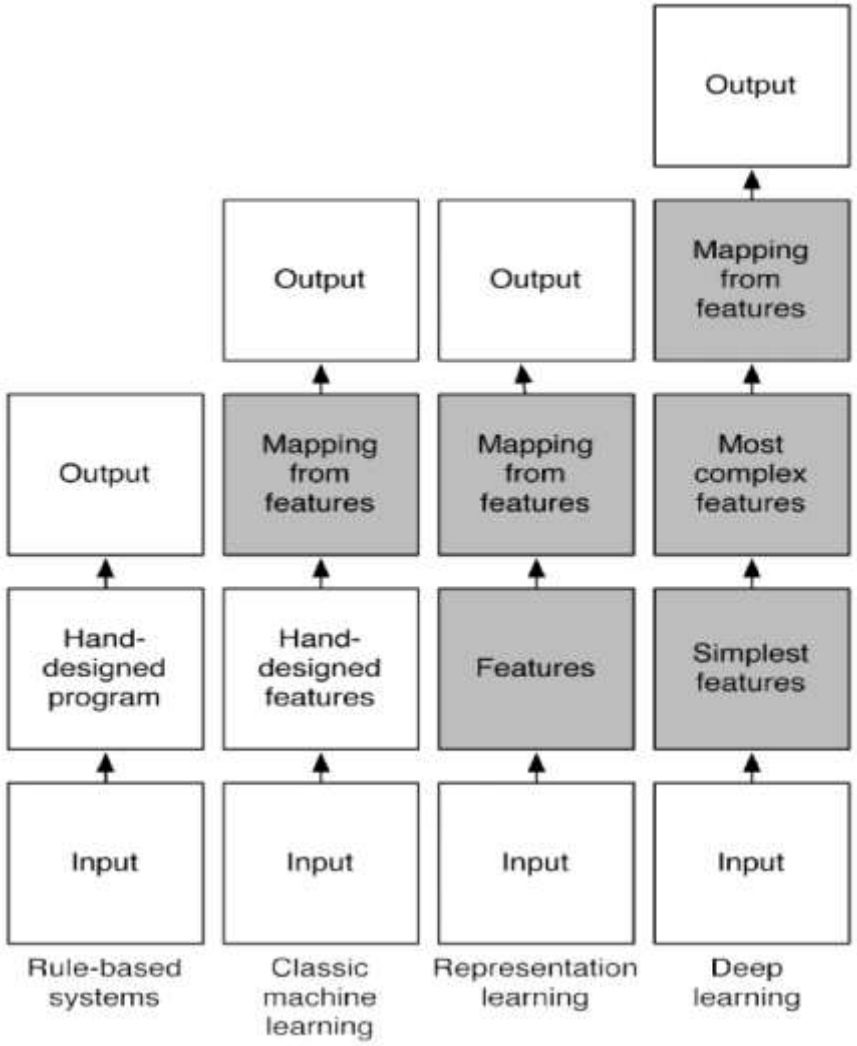
Test sample

Features

Learned model

Prediction

# What is deep learning?




Y. Bengio et al, ``Deep Learning", MIT Press, 2015

Hand-crafted features, Representative Learning and Deep Learning

- "**Hand Crafted" features** refer to properties derived using various algorithms using the information present in the data itself.
- **Representation learning**, also known as feature learning, is a process that allows a machine to identify the most useful features or representations from raw data automatically. This process is crucial in machine learning because it can significantly improve the performance of learning algorithms.
- **Deep learning** is a method in artificial intelligence (AI) that teaches computers to process data in a way that is inspired by the human brain. Deep learning models can recognize complex patterns in pictures, text, sounds, and other data to produce accurate insights and predictions.

# Presentation Outline

- Background
- **Basic concepts**
- Decision Tree Induction
- Bayes Classification Methods
- Lazy Learners (k-nearest neighbours)
- Linear Classifiers
- Model Evaluation and Selection
- Techniques to Improve Accuracy
- Summary

# Classification: Definition

- Given a collection of records (*training set* )
  - 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.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

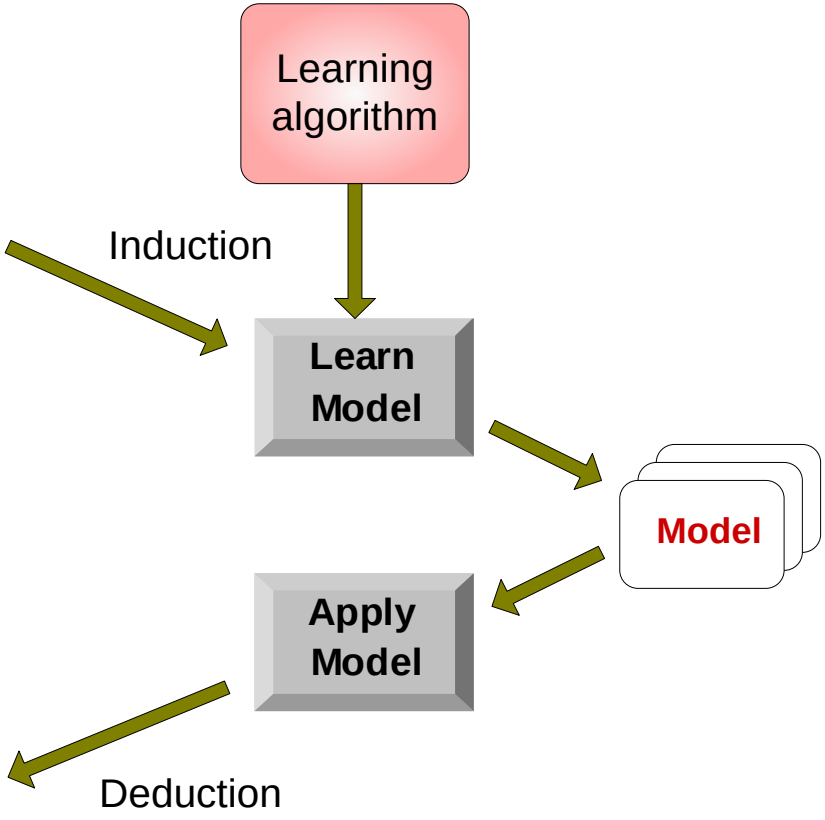
# Illustrating Classification Task

| Tid | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 1 | Yes | Large | 125K | No |
| 2 | No | Medium | 100K | No |
| 3 | No | Small | 70K | No |
| 4 | Yes | Medium | 120K | No |
| 5 | No | Large | 95K | Yes |
| 6 | No | Medium | 60K | No |
| 7 | Yes | Large | 220K | No |
| 8 | No | Small | 85K | Yes |
| 9 | No | Medium | 75K | No |
| 10 | No | Small | 90K | Yes |

Training Set

| Tid | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 11 | No | Small | 55K | ? |
| 12 | Yes | Medium | 80K | ? |
| 13 | Yes | Large | 110K | ? |
| 14 | No | Small | 95K | ? |
| 15 | No | Large | 67K | ? |

Test Set



# Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

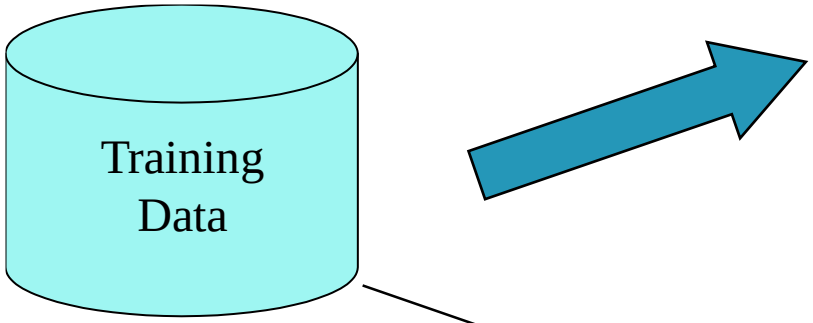
# Classification vs. Prediction

- 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
- Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

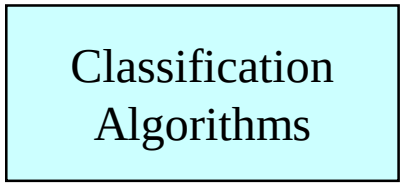
# Classification—A Two-Step Process

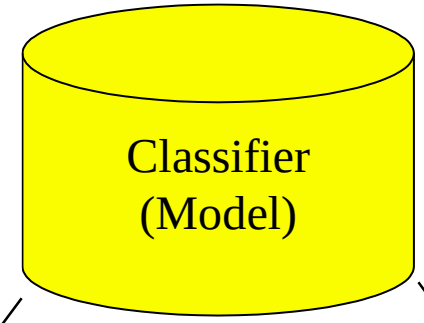
- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision 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 over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

# Process (1): Model Construction





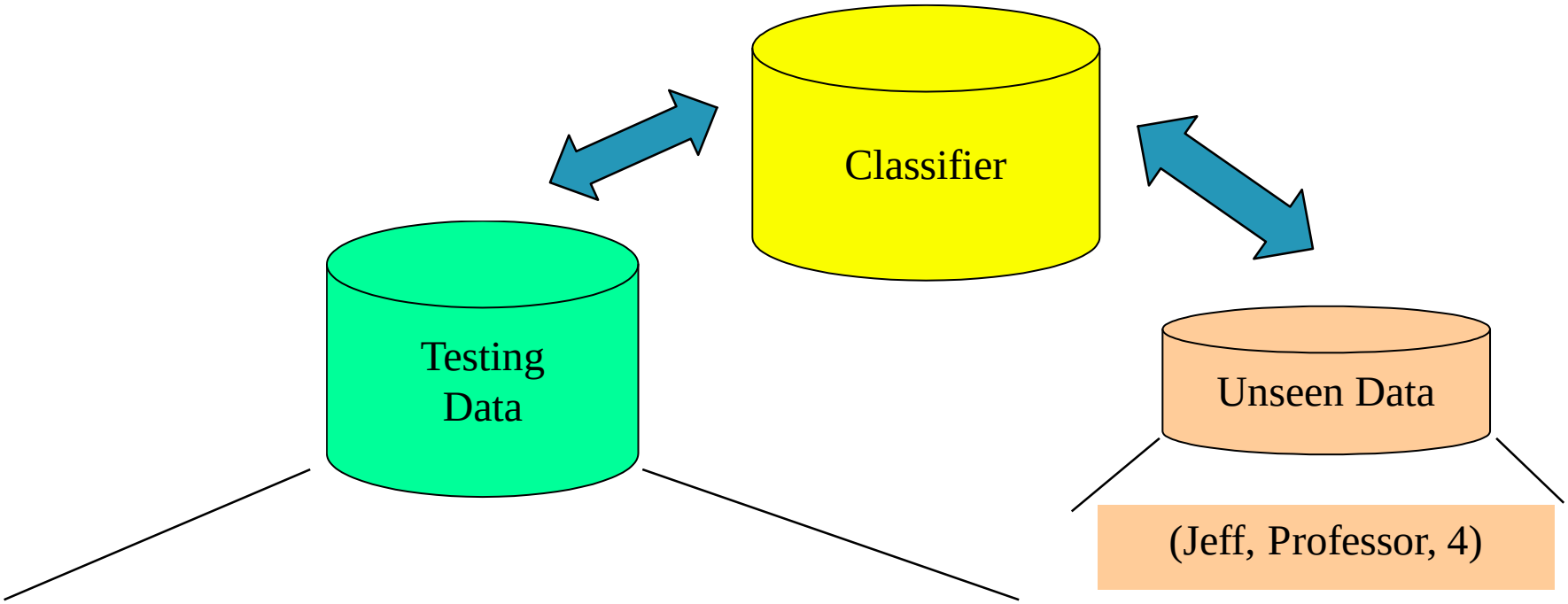




| NAME | RANK | YEARS | TENURED |
|---|---|---|---|
| Mike | Assistant Prof | 3 | no |
| Mary | Assistant Prof | 7 | yes |
| Bill | Professor | 2 | yes |
| Jim | Associate Prof | 7 | yes |
| Dave | Assistant Prof | 6 | no |
| Anne | Associate Prof | 3 | no |

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

# Process (2): Using the Model in Prediction




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

Tenured?

Yes

# Classification Issues: Data Preparation

- Data cleaning
  - 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

# Issues: Evaluating Classification Methods

- 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

# Presentation Outline

- Background
- Basic concepts
- **Decision Tree Induction**
- Bayes Classification Methods
- Rule-Based Classification
- Model Evaluation and Selection
- Techniques to Improve Accuracy
- Summary

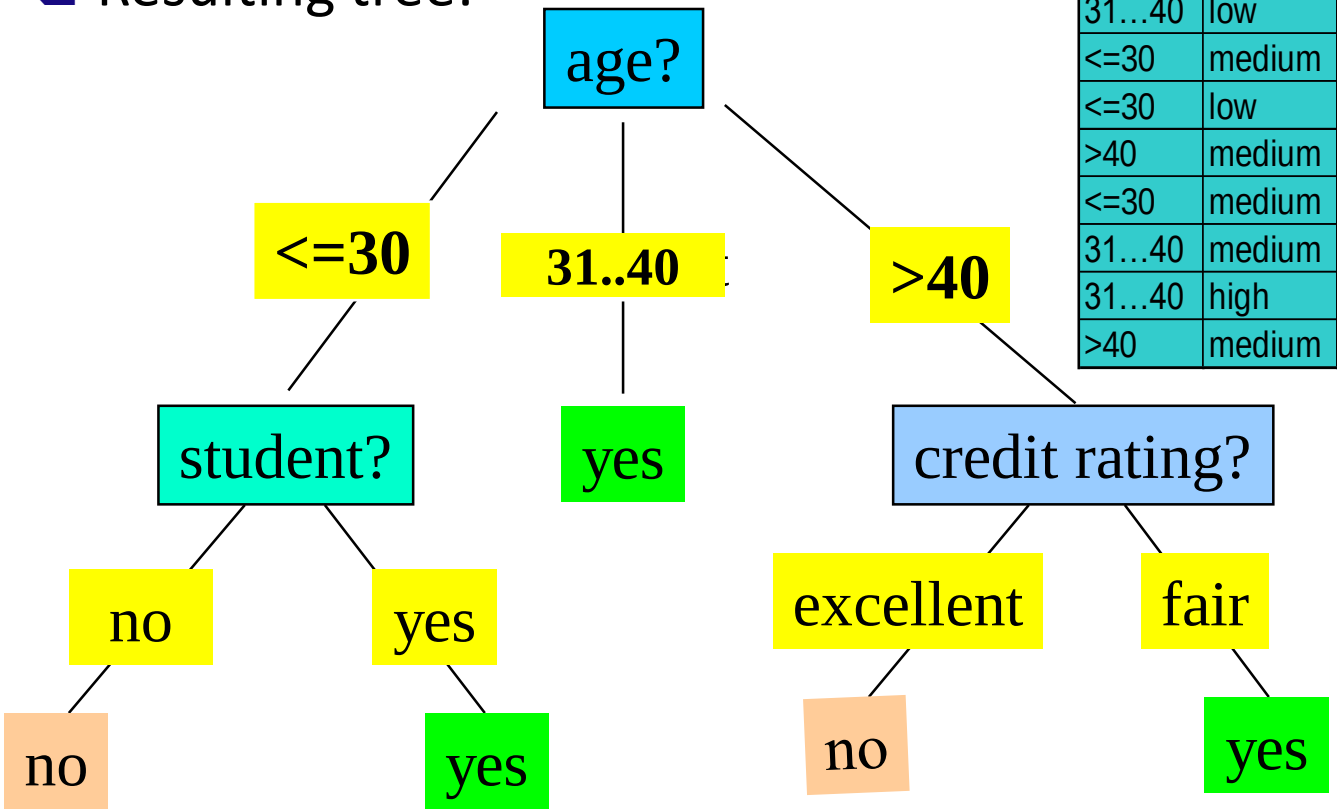
# Decision tree induction (Class #17)

- **Decision tree induction** is the learning of decision trees from class-labeled training tuples.
- A **decision tree** is a flowchart-like tree structure
  - Each **internal node** (nonleaf node) denotes a test on an attribute,
  - Each **branch** represents an outcome of the test
  - Each **leaf node** (or terminal node) holds a class label.
  - **Root node** is the topmost node.

# Decision Tree Induction: An Example

- Training data set: Buys_computer
- The data set follows an example of Quinlan's ID3
- Resulting tree:

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



# Use of decision tree

- For a tuple X, the attribute values are tested against the decision tree.
- A path is traced from the root to leaf.
- Reason for popularity
  - Easy to construct, simple and fast
    - No domain knowledge is required
  - Easy to understand; interpretable
  - They can handle multidimensional data
  - Good accuracy
  - Several applications

# History

- Introduced by J.Ross Quinlan
  - Developed ID3 algorithm (Iterative Dichotomiser)
- Quinlan also presented C4.5, which became a benchmark
- A CART algorithm is published by four researchers independently
  - IBM Intelligent Miner
- Three algorithms: ID3, C4.5, and CART adopt a greedy approach (non-backtracking)
  - Top-down, divide and conquer

# Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

- 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

- Inputs: D, attribute list, and attribute selection method
- D is the complete set of training tuples and associated class lablels
- Attribute list: list of attributes describing the tuples

- Attribute selection method specifies the method to select the attributes.

# Splitting possibilities

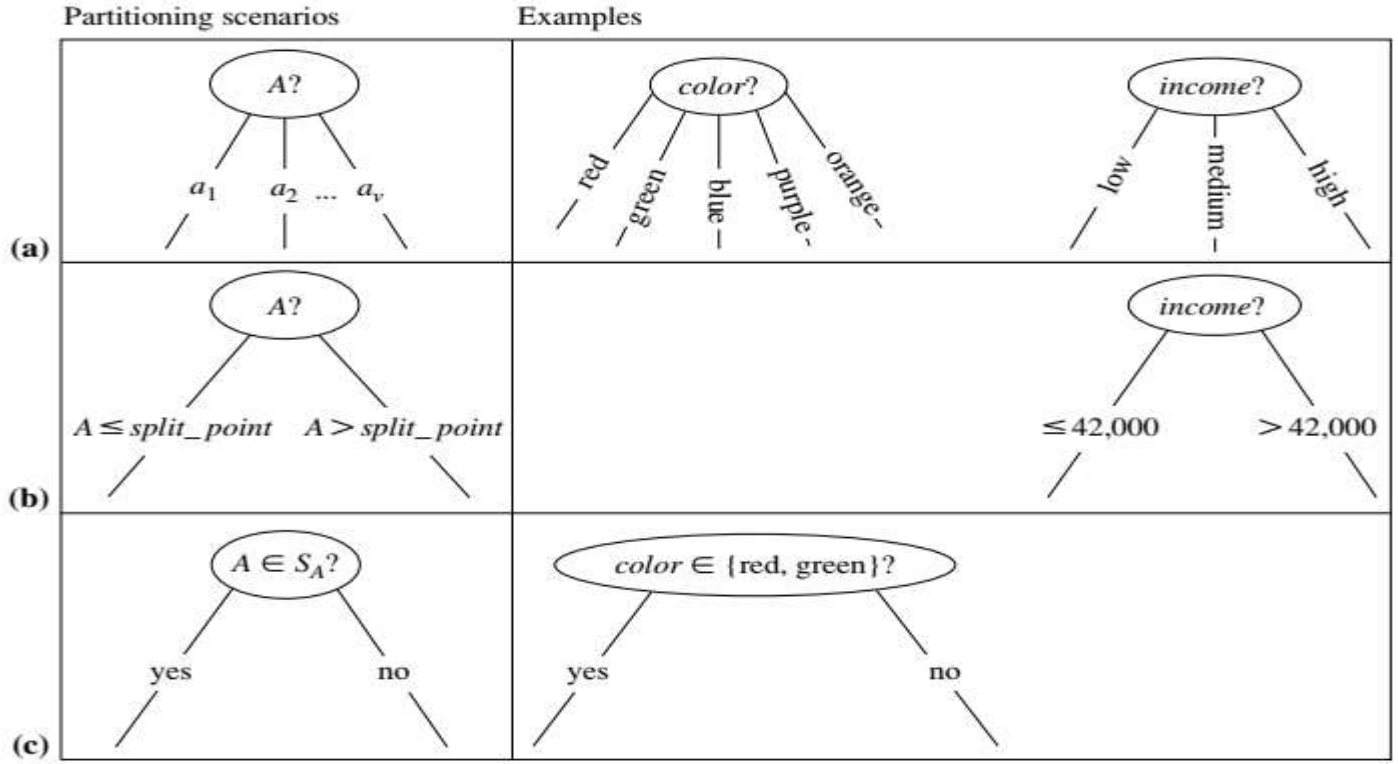
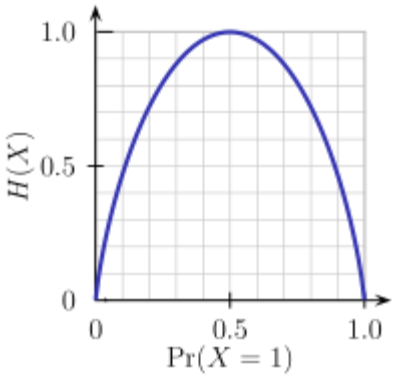



**Figure 8.4** This figure shows three possibilities for partitioning tuples based on the splitting criterion, each with examples. Let $A$ be the splitting attribute. (a) If $A$ is discrete-valued, then one branch is grown for each known value of $A$. (b) If $A$ is continuous-valued, then two branches are grown, corresponding to $A \leq split\_point$ and $A > split\_point$. (c) If $A$ is discrete-valued and a binary tree must be produced, then the test is of the form $A \in S_A$, where $S_A$ is the splitting subset for $A$.

# Brief Review of Entropy

- Entropy (Information Theory)
  - A measure of uncertainty associated with a random variable
  - Calculation: For a discrete random variable $Y$ taking $m$ distinct values $\{y_1, \ldots, y_m\}$,
    - $H(Y) = -\sum_{i=1}^{m} p_i \log(p_i)$ , where $p_i = P(Y = y_i)$
  - Interpretation:
    - Higher entropy => higher uncertainty
    - Lower entropy => lower uncertainty
- Conditional Entropy
  - $H(Y|X) = \sum_x p(x) H(Y|X = x)$




**m = 2**

# About Shannon's entropy

- Consider transmission of a1, a2, a3, a4
  - Option 1: a1= 00, a2=01, a3=10, a4= 11
  - Option 2: a1=0, a2=10, a3=110, a4=111

(i) Option 1: If p(a1)=p(a2)=p(a3)=p(a4)=1/4

Expected code length= 2 bits.

Option 2: Expected code length=2.25 bits

(ii) If p(a1)= 1/2; p(a2)=1/4; p(a3)=1/8; p(a4)=1/8

Option 1: 2 bits (if we do not consider probabilities)

Option 2= 1.75 bits =(-1/2*log(1/2) -1/4*log(1/4)-1/8*log(1/8)-1/8*log(1/8)=1/2+2/4+ 3/8+3/8=1.75

(Equal to Shannon's entropy)

The idea is that *frequent letters should be coded with smaller lengths*.

*The* **Shannon entropy** *of* $A$ *is*

$$H(A) = -\sum_{i=1}^{n} p_i \log_2 p_i.$$

pi= probability of event X

log(pi) is the amount of information of event x

Information content of event x with probability p=log(1/p(x))=-log(p(x))

# Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- 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

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

# Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

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

| age | $p_i$ | $n_i$ | $I(p_i, n_i)$ |
|---|---|---|---|
| <=30 | 2 | 3 | 0.971 |
| 31…40 | 4 | 0 | 0 |
| >40 | 3 | 2 | 0.971 |

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

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

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

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

Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

# 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
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

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

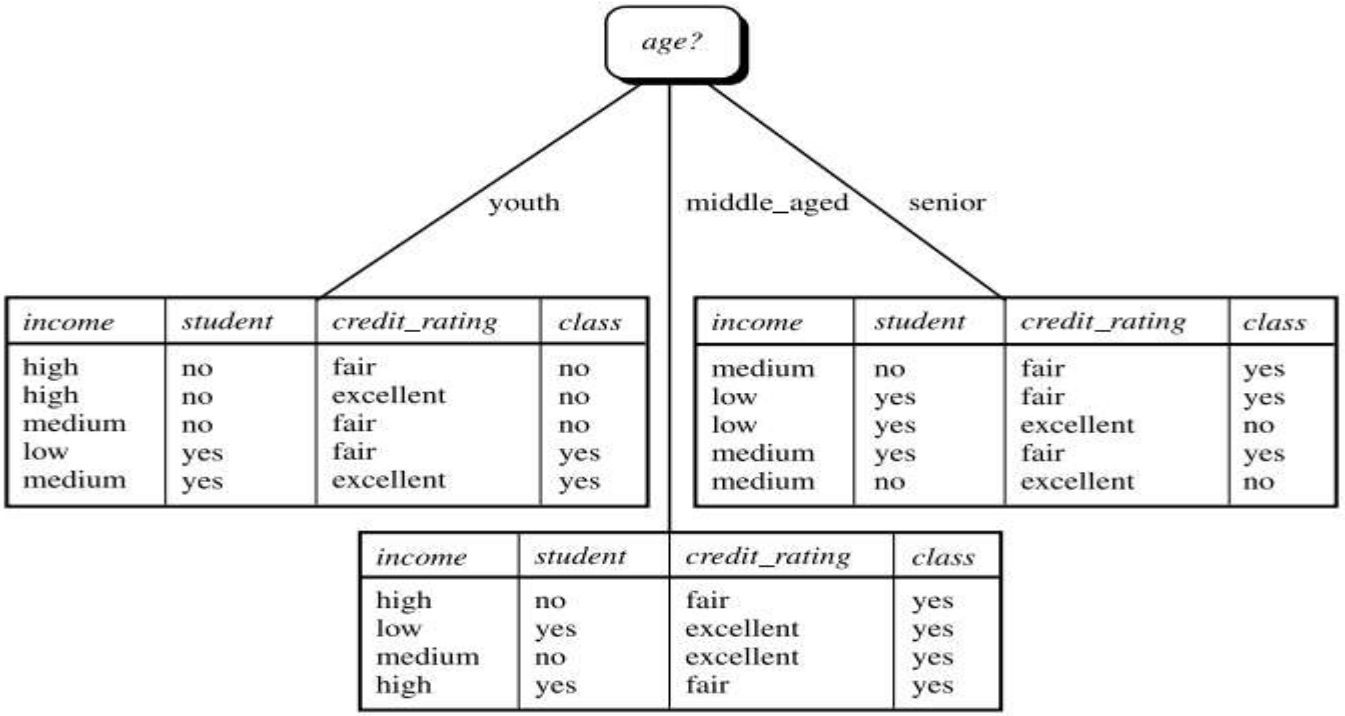
  - GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex.

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

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

**Table 8.1** Class-Labeled Training Tuples from the *AllElectronics* Customer Database

| RID | age | income | student | credit_rating | Class: buys_computer |
|---|---|---|---|---|---|
| 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 |



# Gini Index (CART, IBM IntelligentMiner)

- If a data set $D$ contains examples from $n$ classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where $p_j$ is the relative frequency of class $j$ in $D$

- If a data set $D$ is split on A into two subsets $D_1$ and $D_2$, the *gini* index $gini(D)$ is defined as

- Reduction in Impurity:

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

# Computation of Gini Index

- Ex. 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

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

# 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
    - 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 $\chi^2$ test for independence
- G-statistic: has a close approximation to $\chi^2$ 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

# Enhancements to Basic Decision Tree Induction

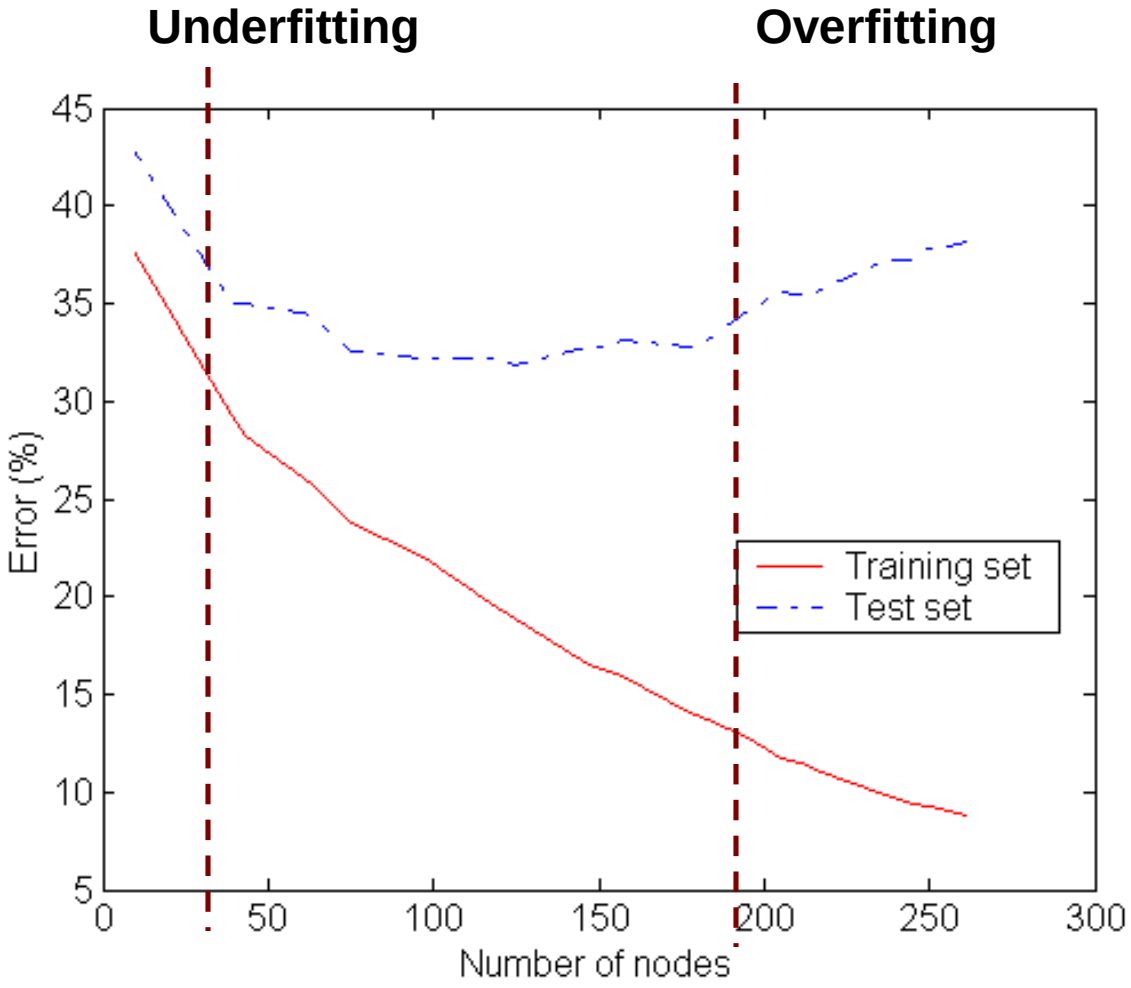
- Allow for **continuous-valued attributes**
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle **missing attribute values**
  - Assign the most common value of the attribute
  - Assign probability to each of the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

# Underfitting and Overfitting

- Underfitting and overfitting are two problems that will adversely affect the accuracy and usefulness of the model.

- Classification requires three data sets
  - Training data set
    - This is the data that are analyzed to first produce a model.
  - Test data set
    - Once a model is developed it is then tested on the evaluation data set. The second data set is an evaluation data set important for refining the model if necessary, and to get a sense of how well the model will perform on subsequent data that it will be applied on.
  - Data on which classification is applied

# Underfitting and Overfitting




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

**Overfitting:** once the tree becomes too large, its test error rates begins to increase even though its training error rate continues to decrease.

# Underfitting

- **Underfitting** is a situation where training and test error rates of the model are large when the size of the tree is very small.
- Underfitting occurs because the model has yet to learn the true structure of the data
- It performs poorly on both training and test sets
- As number of nodes in the decision tree increases the tree will have a fewer training and test errors.
- However, once the tree becomes too large, its test error rates begins to increase even though its training error rate continues to decrease. It is called **overfitting.**

# Underfitting

- Underfitting refers to a model that is too general and fails to find interesting patterns in the data.
  - This can result from not including important variables as inputs during the model building process.
  - In terms of a loan application scenario, the analyst or model-builder may include annual salary as an important factor, but may exclude information about the applicant's job, which may turn out to be important.
  - For example, jobs that are seasonal (swimming pool maintenance, landscaping, etc.) may affect the person's ability to submit monthly payments during the times when work is slower -- information that is not reflected by their annual salary.
  - This might suggest that as much information as possible should be included as inputs during the model building process to avoid underfitting.

# More on Underfitting

- Training error can be reduced by increasing the model complexity.
  - Leaf-nodes can be expanded until it perfectly fits the training data.
  - As a result the test error may be large because the tree may fit some of the noise points in the data.
  - Such nodes degrade the performance as they do not generalize well to the test examples.

# Overfitting

- Overfitting: once the tree becomes too large, its test error rates begins to increase even though its training error rate continues to decrease.
- Including as much information as possible to develop a model can lead to the problem of overfitting.
- Overfitting refers to models that are too specific, or too sensitive to the particulars of the data (the training set) used to build the model.
- This can be due to having too many variables used as inputs and/or a non-representative training set.
  - In the loan application scenario, if in the training set, many people that defaulted on their loans happened to be named "Smith" (a popular name), then the model (e.g. a decision tree) may decide that if the applicant's last name is "Smith", then deny the loan.
  - In refining the model, perhaps last name should not serve as an input. More importantly, the characteristics of the data used to build the model have to be representative of the data at large -- it's unlikely that people named Smith have a disproportionately high rate of defaulting on loans.

# Overfitting

- Overfitting can be corrected using the evaluation data set. If accurate performance on the training set is due to particular characteristics in that data (e.g. person's last name), then performance will be poor on the evaluation set as long as the evaluation set does not share these idiosyncrasies.
- Refining the model (e.g. by pruning the decision tree) involves setting performance to be generally equivalent on both the training and evaluation data sets.
- This will generally give the analyst the idea of how well the model may perform on "real" data.

# Overfitting Examples

- Overfitting can still occur despite the use of a training and evaluation data set.

- This can be the result of poorly creating the training and evaluation sets -- so that neither is **representative of subsequent data that the model will be applied to.**

- For example, of predicting stock market performance.
- One year's worth of data were partitioned into equal sized training and evaluation sets. A quantitative model was developed based on the training set -- and its predictive power on this dataset was impressive. It was equally impressive on the evaluation dataset, apparently requiring no refinement. But when applied to the next year's data it performed miserably despite the fact that there were not significant events related to the stock market. The reason for this is had to do with how the training and evaluation data sets were created. The training and evaluation sets were based on the daily closing values of alternate days. Day 1's close was assigned to the training set, day 2 to the evaluation set, day 3 to the training set, day 4 to the evaluation set, and so forth. As a result the overfitting that occurred when the model picked up factors tied to the temporal fluctuations in the stock market's closing values were carried over into the evaluation set as well.

# Overfitting and Tree Pruning

- 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
- 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"

# Classification in Large Databases

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why is decision tree induction popular?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods
- RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - Builds an AVC-list (attribute, value, class label)

# Scalability Framework for RainForest

- Separates the scalability aspects from the criteria that determine the quality of the tree
- Builds an AVC-list**: AVC (Attribute, Value, Class_label)**
- **AVC-set** (of an attribute *X* )
  - Projection of training dataset onto the attribute *X* and class label where counts of individual class label are aggregated
- **AVC-group** (of a node *n* )
  - Set of AVC-sets of all predictor attributes at the node *n*

# Rainforest: Training Set and Its AVC Sets

## Training Examples

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

AVC-set on *Age*

| Age | Buy_Computer | |
|---|---|---|
| | yes | no |
| <=30 | 2 | 3 |
| 31..40 | 4 | 0 |
| >40 | 3 | 2 |

AVC-set on *income*

| income | Buy_Computer | |
|---|---|---|
| | yes | no |
| high | 2 | 2 |
| medium | 4 | 2 |
| low | 3 | 1 |

AVC-set on *Student*

| student | Buy_Computer | |
|---|---|---|
| | yes | no |
| yes | 6 | 1 |
| no | 3 | 4 |

AVC-set on *credit_rating*

| Credit rating | Buy_Computer | |
|---|---|---|
| | yes | no |
| fair | 6 | 2 |
| excellent | 3 | 3 |

# BOAT (Bootstrapped Optimistic Algorithm for Tree Construction)

- Use a statistical technique called *bootstrapping* to create several smaller samples (subsets), each fits in memory
- Each subset is used to create a tree, resulting in several trees
- These trees are examined and used to construct a new tree *T'*
  - It turns out that *T'* is very close to the tree that would be generated using the whole data set together
- Adv: requires only two scans of DB, an incremental alg.

# Presentation Outline (Class #18)

- Background
- Basic concepts
- Decision Tree Induction
- **Bayes Classification Methods**
- Rule-Based Classification
- Model Evaluation and Selection
- Techniques to Improve Accuracy
- Summary

# 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

# Derivation of Bays' theorem

- Definition: The conditional probability that A is true, given that B, with the notion P(A/B) (read "probability of A given B). Here, P(A$\wedge$B) is probability of the joint intersection of events *A* and *B*
  - P(A/B)= P(A $\wedge$ B)/P(B)
  - Or P(A $\wedge$ B)= P(A/B) * P(B)
- Bays theorem
  - P(A $\wedge$ B)= P(B $\wedge$ A)
  - P(B$\wedge$ A)=P(B/A) * P(A)
- So, we have P(A/B)=P(B/A)*P(A) / P(B)

- Let H be some hypothesis, such as the data sample X belongs to a specified class C. For this we want to determine P(H/X), the probability that the hypothesis H holds given the observed sample X.
  - P(H/X)=P(X/H)*P(H)/P(X)

# Examples of conditional probability

- Example 1: Suppose you are drawing three marbles—red, blue, and green—from a bag. Each marble has an equal chance of being drawn. What is the conditional probability of drawing the red marble after already drawing the blue one?
  - First, the probability of drawing a blue marble is about 33% because it is one possible outcome out of three. Assuming this first event occurs, there will be two marbles remaining, with each having a 50% chance of being drawn. So the chance of drawing a blue marble after already drawing a red marble would be about 16.5% (33% x 50%).
- Example 2: Consider that a fair die has been rolled and you are asked to give the probability that it was a five. There are six equally likely outcomes, so your answer is 1/6.
  - But imagine if before you answer, you get extra information that the number rolled was odd. Since there are only three odd numbers that are possible, one of which is five, you would certainly revise your estimate for the likelihood that a five was rolled from 1/6 to 1/3.
- Example 3: Suppose a student is applying for admission to a university and hopes to receive an academic scholarship. The school to which they are applying accepts 100 of every 1,000 applicants (10%) and awards academic scholarships to 10 of every 500 students who are accepted (2%). Of the scholarship recipients, 50% of them also receive university stipends for books, meals, and housing.
  - For the students, the chance of them being accepted and then receiving a scholarship is .2% (.1 x .02). The chance of them being accepted, receiving the scholarship, then also receiving a stipend for books, etc. is .1% (.1 x .02 x .5).

# Bayesian Classification: Simple Overview

- "The essence of the Bayesian approach is to provide a mathematical rule explaining how you should change your existing beliefs in the light of new evidence.
- In other words, it allows scientists to combine new data with their existing knowledge or expertise.

- The canonical example is to imagine that a precocious newborn observes his first sunset, and wonders whether the sun will rise again or not. He assigns equal prior probabilities to both possible outcomes, and represents this by placing one white and one black marble into a bag. The following day, when the sun rises, the child places another white marble in the bag. The probability that a marble plucked randomly from the bag will be white (ie, the child's degree of belief in future sunrises) has thus gone from a half to two-thirds. After sunrise the next day, the child adds another white marble, and the probability (and thus the degree of belief) goes from two-thirds to three-quarters. And so on. Gradually, the initial belief that the sun is just as likely as not to rise each morning is modified to become a near-certainty that the sun will always rise."

# Bayesian Classification: Simple overview

- Suppose your data consist of fruits, described by their color and shape. Bayesian classifiers operate by saying "If you see a fruit that is red and round, which type of fruit is it most likely to be, based on the observed data sample? In future, classify red and round fruit as that type of fruit.“
- A difficulty arises when you have more than a few variables and classes -- you would require an enormous number of observations (records) to estimate these probabilities.
- Naive Baye’s classification gets around this problem by not requiring that you have lots of observations for each possible combination of the variables. Rather, the variables are assumed to be independent of one another and, therefore the probability that a fruit that is red, round, firm, 3" in diameter, etc. will be an apple can be calculated from the independent probabilities that a fruit is red, that it is round, that it is firm, that is 3" in diameter, etc.
- In other words, Naive Bayes classifiers assume that the effect of an variable value on a given class is independent of the values of other variable. This assumption is called class conditional independence. It is made to simplify the computation and in this sense considered to be naïve.

# Bayesian Classification: Simple overview…

- This assumption is a fairly strong assumption and is often not applicable. However, bias in estimating probabilities often may not make a difference in practice -- it is the order of the probabilities, not their exact values, that determine the classifications.
- Studies comparing classification algorithms have found the Naïve Bayesian classifier to be comparable in performance with classification trees and with neural network classifiers. They have also exhibited high accuracy and speed when applied to large databases.

# Bays Theorem

Let X be the data record (case) whose class label is unknown. Let H be some hypothesis, such as "data record X belongs to a specified class C.“

For classification, we want to determine P (H|X) – the probability that the hypothesis H holds, given the observed data record X.

P (H|X) is the posterior probability of H conditioned on X. For example, the probability that a fruit is an apple, given the condition that it is red and round.

In contrast, P(H) is the prior probability, or a priori probability, of H. In this example P(H) is the probability that any given data record is an apple, regardless of how the data record looks.

The posterior probability, P (H|X), is based on more information (such as background knowledge) than the prior probability, P(H), which is independent of X.

# Bayesian Classification: Simple introduction…

Similarly, P (X|H) is posterior probability of X conditioned on H. That is, it is the probability that X is red and round given that we know that it is true that X is an apple.

P(X) is the prior probability of X, i.e., it is the probability that a data record from our set of fruits is red and round.

Bayes theorem is useful in that it provides a way of calculating the posterior probability, P(H|X), from P(H), P(X), and P(X|H). Bayes theorem is

$$P(H|X) = P(X|H)\, P(H) / P(X)$$

# Bayes Classifier

- A probabilistic framework for solving classification problems
- Conditional Probability:

$$P(C \mid A) = \frac{P(A,C)}{P(A)}$$

$$P(A \mid C) = \frac{P(A,C)}{P(C)}$$

- Bayes theorem:

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

# Example of Bayes Theorem

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20

- If a patient has a stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

# Bayesian Classifiers

- Consider each attribute and class label as random variables

- Given a record with attributes ($A_1$, $A_2$,…,$A_n$)
  - Goal is to predict class C
  - Specifically, we want to find the value of C that maximizes P(C| $A_1$, $A_2$,…,$A_n$ )

- Can we estimate P(C| $A_1$, $A_2$,…,$A_n$ ) directly from data?

# Bayesian Classifiers

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes theorem

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

  - Choose value of C that maximizes

    $P(C \mid A_1, A_2, \ldots, A_n)$

  - Equivalent to choosing value of C that maximizes

    $P(A_1, A_2, \ldots, A_n \mid C)\ P(C)$

- How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$ ?

# Naïve Bayes Classifier

- Assume independence among attributes $A_i$ when class is given:
  - $P(A_1, A_2, \ldots, A_n \mid C) = P(A_1 \mid C_j)\, P(A_2 \mid C_j) \ldots P(A_n \mid C_j)$
  - Can estimate $P(A_i \mid C_j)$ for all $A_i$ and $C_j$.
  - New point is classified to $C_j$ if $P(C_j) \prod P(A_i \mid C_j)$ is maximal.

# How to Estimate Probabilities from Data?

| Tid | Refund | Marital Status | Taxable Income | Evade |
|---|---|---|---|---|
| 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 |

- Class: $P(C) = N_c/N$
  - e.g., P(No) = 7/10,
    P(Yes) = 3/10

- For discrete attributes:

$$P(A_i \mid C_k) = |A_{ik}| / N_c$$

  - where $|A_{ik}|$ is number of instances having attribute $A_i$ and belongs to class $C_k$
  - Examples:

    P(Status=Married|No) = 4/7
    P(Refund=Yes|Yes)=0

# How to Estimate Probabilities from Data?

- For continuous attributes:
  - Discretize the range into bins
    - one ordinal attribute per bin
    - violates independence assumption
  - Two-way split: $(A < v)$ or $(A > v)$
    - choose only one of the two splits as new attribute
  - Probability density estimation:
    - Assume attribute follows a normal distribution
    - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|c)$

# How to Estimate Probabilities from Data?

| Tid | Refund | Marital Status | Taxable Income | Evade |
|---|---|---|---|---|
| 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 |

- Normal distribution:

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each $(A_i, c_i)$ pair

- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

# Example of Naïve Bayes Classifier

**Given a Test Record:**

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No)
  × P(Married| Class=No)
  × P(Income=120K| Class=No)
  = 4/7 × 4/7 × 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes)
  × P(Married| Class=Yes)
  × P(Income=120K| Class=Yes)
  = 1 × 0 × 1.2 × $10^{-9}$ = 0

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)

=> Class = No

# Naïve Bayes Classifier

- If one of the conditional probability is zero, then the entire expression becomes zero
- Probability estimation:

$$\text{Original} : P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace} : P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m-estimate} : P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

c: number of classes

p: prior probability

m: parameter

# Naïve Bayes Classifier: Example 1

| Name | Give Birth | Can Fly | Live in Water | Have Legs | Class |
|---|---|---|---|---|---|
| human | yes | no | no | yes | mammals |
| python | no | no | no | no | non-mammals |
| salmon | no | no | yes | no | non-mammals |
| whale | yes | no | yes | no | mammals |
| frog | no | no | sometimes | yes | non-mammals |
| komodo | no | no | no | yes | non-mammals |
| bat | yes | yes | no | yes | mammals |
| pigeon | no | yes | no | yes | non-mammals |
| cat | yes | no | no | yes | mammals |
| leopard shark | yes | no | yes | no | non-mammals |
| turtle | no | no | sometimes | yes | non-mammals |
| penguin | no | no | sometimes | yes | non-mammals |
| porcupine | yes | no | no | yes | mammals |
| eel | no | no | yes | no | non-mammals |
| salamander | no | no | sometimes | yes | non-mammals |
| gila monster | no | no | no | yes | non-mammals |
| platypus | no | no | no | yes | mammals |
| owl | no | yes | no | yes | non-mammals |
| dolphin | yes | no | yes | no | mammals |
| eagle | no | yes | no | yes | non-mammals |

**A: attributes**

**M: mammals**

**N: non-mammals**

$$P(A \mid M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A \mid N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A \mid M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A \mid N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

| Give Birth | Can Fly | Live in Water | Have Legs | Class |
|---|---|---|---|---|
| yes | no | yes | no | ? |

**P(A|M)P(M) > P(A|N)P(N)**

**=> Mammals**

# Naïve Bayes Classifier: Example 2: 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 |

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

| |
|---|
| **P(p) = 9/14** |
| **P(n) = 5/14** |

# Example 2….

- Classify an unseen sample X = <rain, hot, high, false>

- P(X|p)·P(p) =
  P(rain|p)·P(hot|p)·P(high|p)·P(false|p)·P(p) =
  3/9·2/9·3/9·6/9·9/14 = 0.010582
- P(X|n)·P(n) =
  P(rain|n)·P(hot|n)·P(high|n)·P(false|n)·P(n) =
  2/5·2/5·4/5·2/5·5/14 = 0.018286

- Sample X is classified in class n (don't play)

# Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
  - … makes computation possible
  - … 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
    - Decision trees, that reason on one attribute at the time, considering most important attributes first

# Presentation Outline

- Background
- Basic concepts
- Decision Tree Induction
- Bayes Classification Methods
- **Lazy Learners (k-nearest neighbours)**
- Linear Classifiers
- Model Evaluation and Selection
- Techniques to Improve Accuracy
- Summary

# Lazy vs. Eager Learning

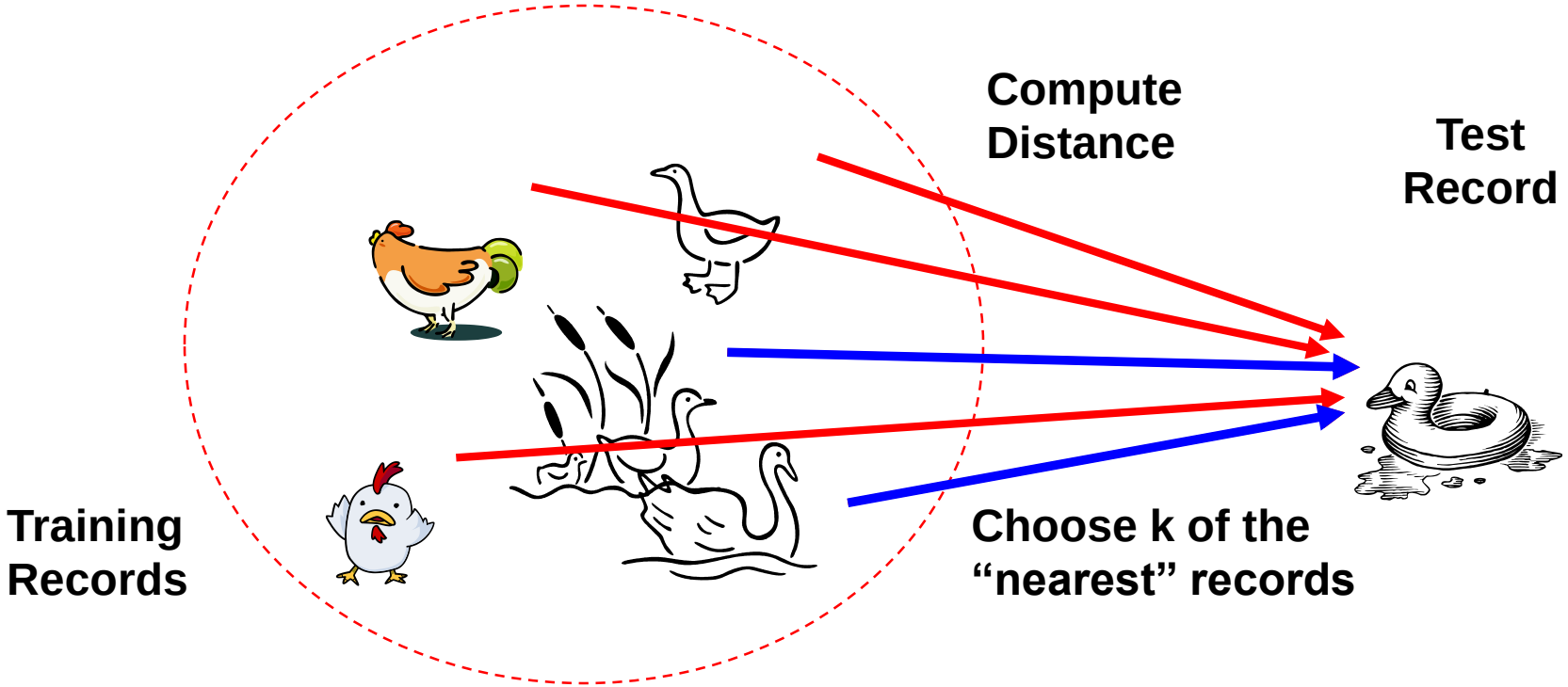
- Lazy vs. eager learning
  - **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - **Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form an implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

# Lazy Learner: Instance-Based Methods

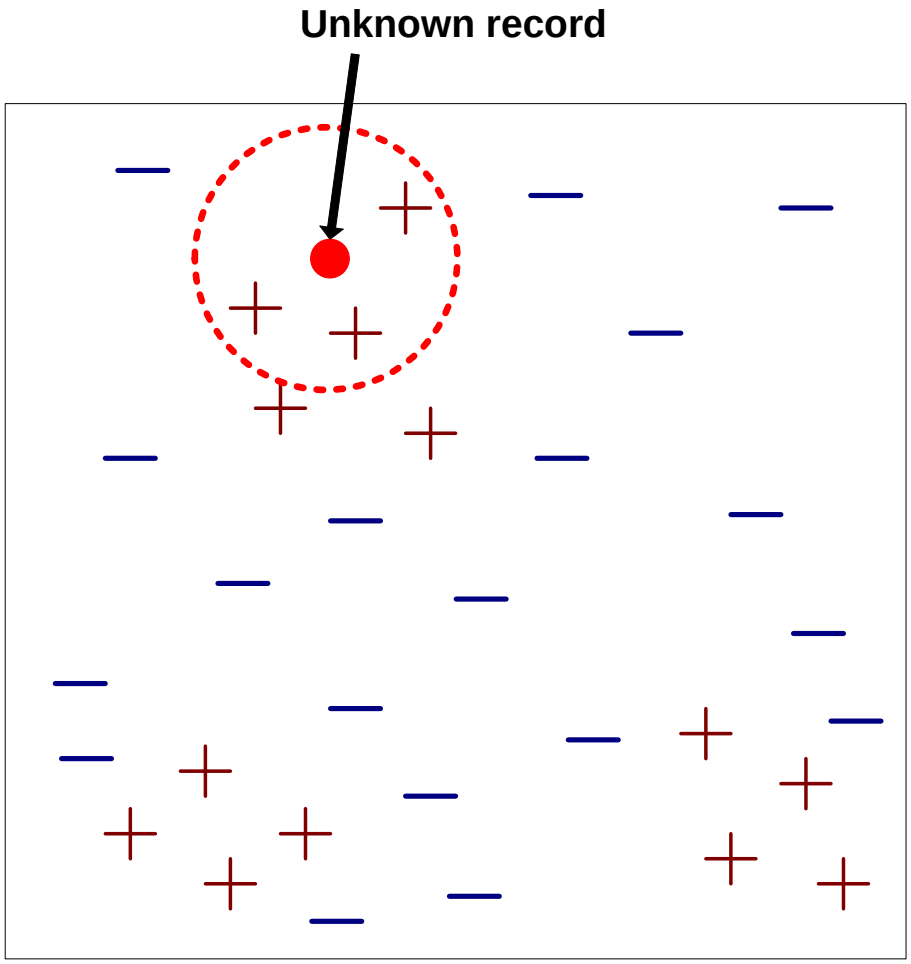
- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference

# k-Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck

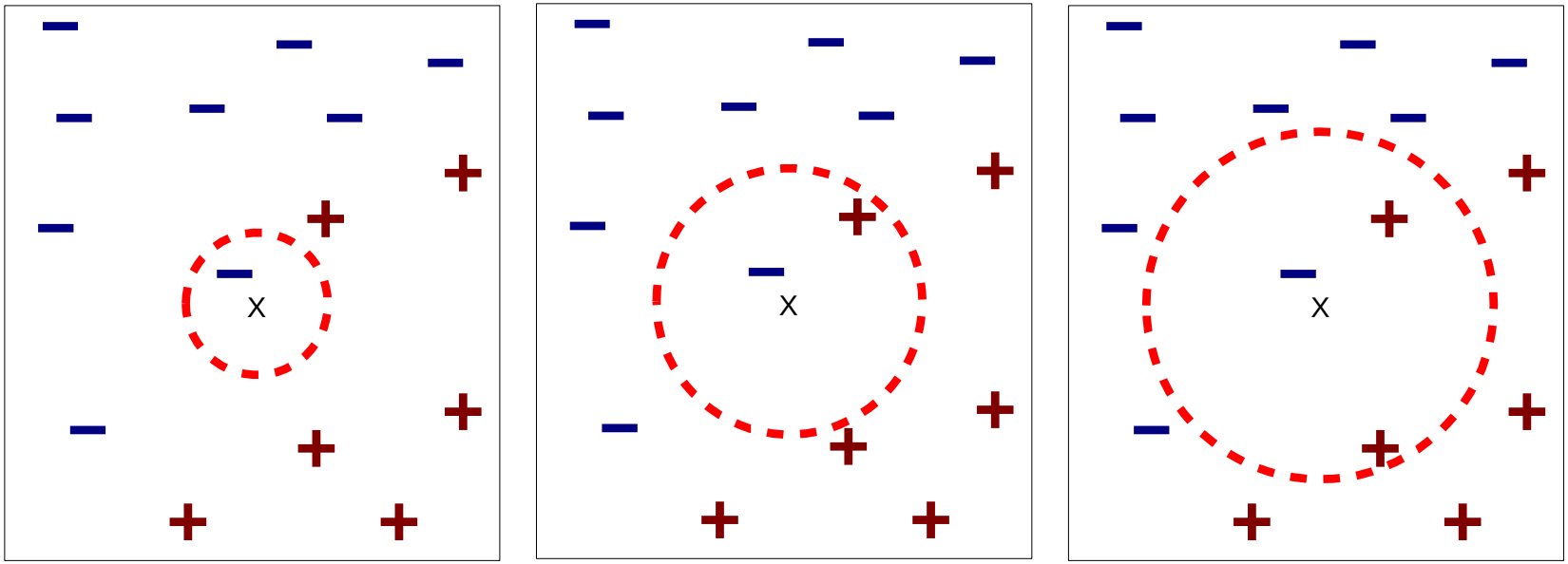


# k-Nearest-Neighbor Classifiers




- 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

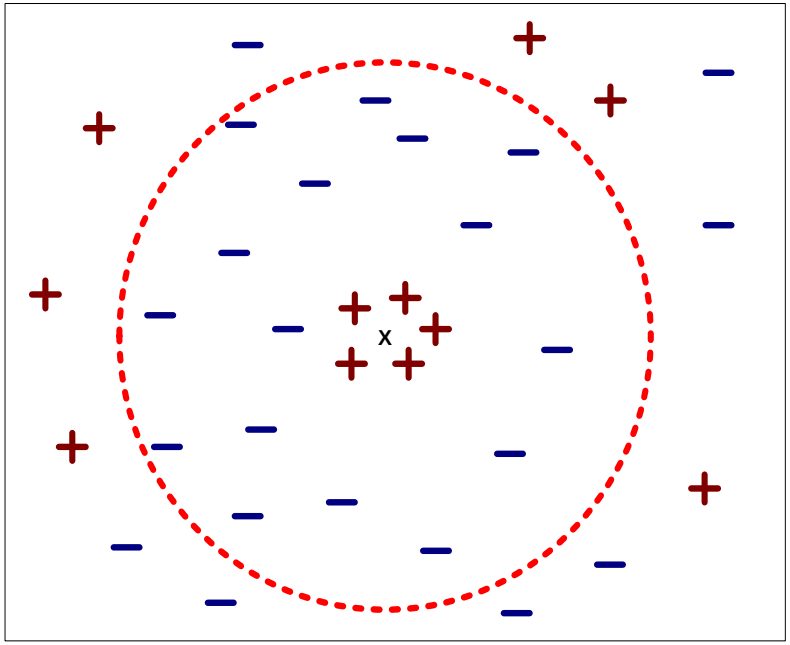
- Compute distance between two points:
  - Euclidean distance

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

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

# Nearest Neighbor Classification…

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



# Nearest Neighbor Classification…

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb
    - income of a person may vary from $10K to $1M

# Nearest Neighbor Classification…

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results

| | | | |
|---|---|---|---|
| 1 1 1 1 1 1 1 1 1 1 1 0 | | 1 0 0 0 0 0 0 0 0 0 0 0 | |
| 0 1 1 1 1 1 1 1 1 1 1 1 | vs | 0 0 0 0 0 0 0 0 0 0 0 1 | |
| **d = 1.4142** | | **d = 1.4142** | |

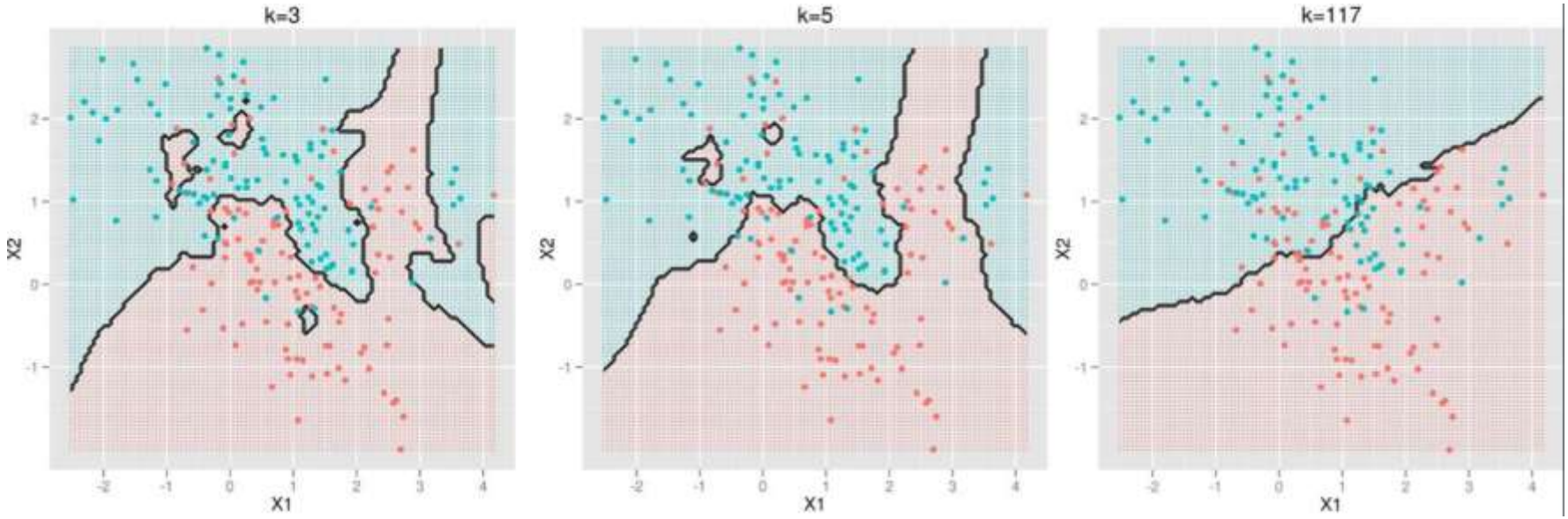
- Solution: Normalize the vectors to unit length

# Nearest neighbor Classification…

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive

# Selection of k for kNN

- The number of neighbors k
  - Small k: overfitting (high var., low bias)
  - Big k: bringing too many irrelevant points (high bias, low var.)




http://scott.fortmann-roe.com/docs/BiasVariance.html

# Case-Based Reasoning (CBR)

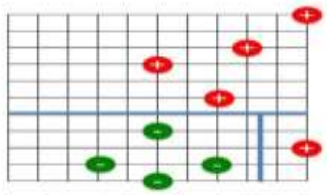
- **CBR**: Uses a database of problem solutions to solve new problems
- Store symbolic description (tuples or cases)—not points in a Euclidean space
- Applications: Customer-service (product-related diagnosis), legal ruling
- Methodology
  - Instances represented by rich symbolic descriptions (e.g., function graphs)
  - Search for similar cases, multiple retrieved cases may be combined
  - Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- Challenges
  - Find a good similarity metric
  - Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases

# Presentation Outline (Class #19)

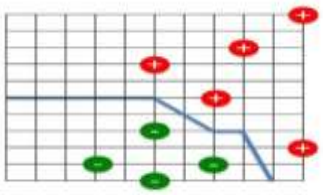
- Background
- Basic concepts
- Decision Tree Induction
- Bayes Classification Methods
- Lazy Learners (k-nearest neighbours)
- **Linear Classifiers**
- Model Evaluation and Selection
- Techniques to Improve Accuracy
- Summary

# About Linear classifiers

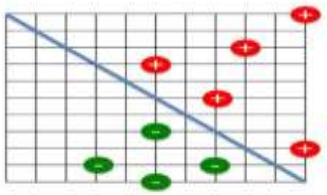
- Classifies generate complex decision boundaries.
- Decision tree might generate hyperrectangular shape boundary
- 1-nearest neighbour classifier may generate hyper polygonal boundary
- Linear classifier may general linear boundary
  - Better generalization performance, better interpretability
- We discuss three classifiers: Linear regression, perceptron, logistic regression (widely used)



(a) Decision Tree Classifier



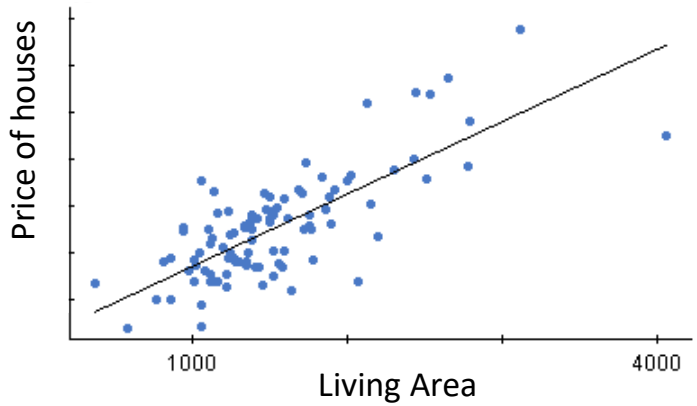
(b) 1—Nearest-Neighbor Classifier

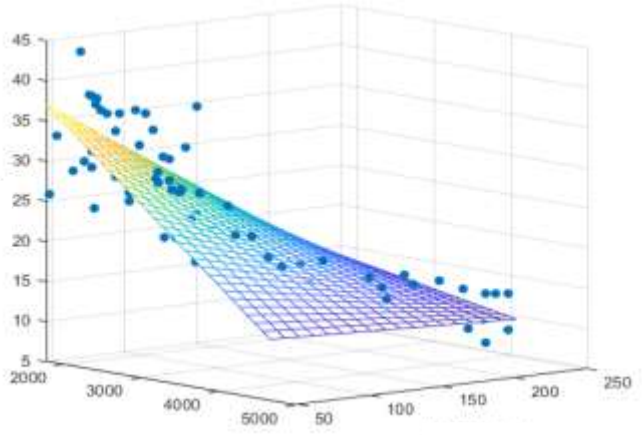


(c) Linear Classifier

# Linear Regression Problem: Example

- Mapping from independent attributes to **continuous value**: x => y
- {living area} => Price of the house
- {college; major; GPA} => Future Income



# Linear Regression Problem: Model

- Linear regression
  - Data: n independent objects
    - Observed Value: $y_i, i = 1,2,3, \cdots, n$
    - p-dimensional attributes: $x_i = \left(x_{i1}, x_{i2}, \cdots, x_{ip}\right)^T$, $i = 1,2,3 \cdots, n$
  - Model:
    - Weight vector: $w = \left(w_1, w_2, \cdots, w_p\right)$
    - $y_i = w^T x_i + b$
  - The weight vector w and bias b are the model parameter learnt by data

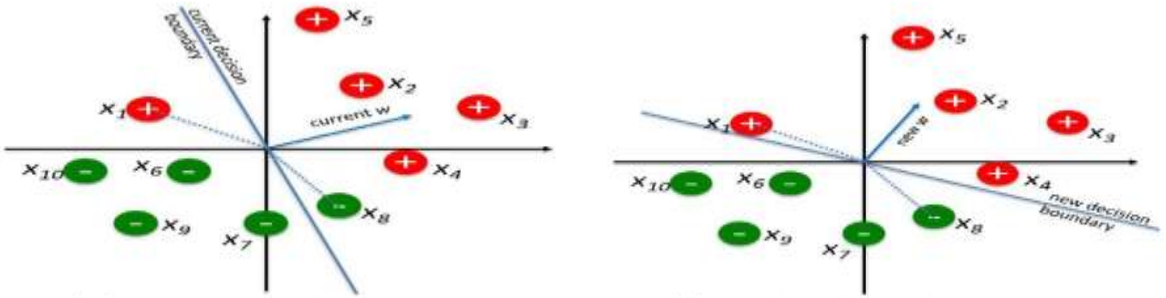
# Linear Regression Model: Solution

- Least Square Method
  - Cost / Loss Function: $\mathrm{L}(w,b) = \Sigma_{i=1}^{n}(y_i - wx_i - b)^2$
  - Optimization Goal: argmin $\mathrm{L}(w,b) = \Sigma_{i=1}^{n}(y_i - wx_i - b)^2$
  - Closed-form solution:
    - $w = \frac{\Sigma_{i=1}^{n} x_i(y_i - \bar{y})}{\Sigma_{i=1}^{n} x_i^2 - n\left(\Sigma_{i=1}^{n} x_i\right)^2}$ $\quad b = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - wx_i)$

- For multiple attributes, multiple linear regression methods are applied.

# Perceptron

- Consider a binary classification task
- The output value *yi* for a given tuple is a binary variable: *y*i = +1 indicates the *i*th tuple is a positive tuple (e.g., *buy computer*) and *yi* = 0 indicates the *i*th tuple is a negative one (e.g., *not buy computer*).
- Output of regression function is the sign of regression function
- Sign=+1 if yi is the predicted class label, sign=0, otherwise
- If we know the weight vector W, we can predict the class label.
- W is interactively learned from the training set.
- If the training tuples are linearly separable, the perceptron algorithm is guaranteed to find a weight vector (i.e., a hyperplane decision boundary)



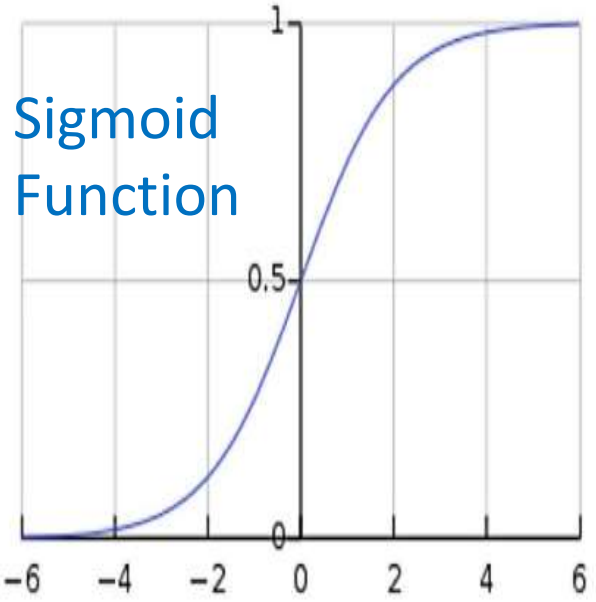

(a) current weight vector $w$ (b) updated weight vector $w$

# Logistic Regression

- Perceptron predicts the binary class label of a given tuple. However, can we also tell how confident such a prediction is?
- Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables.
- To convert the output to 0 to 1, a sigmoid function is introduced.
  - Sigmoid function (differentiable function) :
  - $\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1}$
  - Projects $(-\infty, +\infty)$ to $[0, 1]$
  - Not only LR uses this function,

    but also neural network, deep learning

- To determine the optimal W vector, the maximum likelihood estimation method is employed.
- It aims to solve the following optimization problem: "choosing the the best weight vector *w* that maximizes the likelihood of the training set.



# Presentation Outline

- Background
- Basic concepts
- Decision Tree Induction
- Bayes Classification Methods
- Lazy Learners
  - K-nearest neighbors
  - Case based reasoning
- Linear Classifiers
  - Linear regression
  - Perceptron: turning linear regression to classification
  - Logistic regression
- **Model Evaluation and Selection**
- Techniques to Improve Accuracy
- Summary

# Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use **validation test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap
- Comparing classifiers:
  - Confidence intervals
  - Cost-benefit analysis and ROC Curves

# Confusion Matrix

- **True positives** (TP): These refer to the positive tuples that were correctly labeled by the classifier.
- **True negatives** (TN): These are the negative tuples that were correctly labeled by the classifier.
- **False positives** (FP): These are the negative tuples that were incorrectly labeled as positive (e.g., tuples of class buys_computer = no for which the classifier predicted buys_computer = yes).
- **False negatives (FN):** These are the positive tuples that were mislabeled as negative (e.g., tuples of class buys_computer = yes for which the classifier predicted buys_computer = no).

# Classifier Evaluation Metrics: Confusion Matrix

## Confusion Matrix:

| Actual class\Predicted class | $C_1$ | ¬ $C_1$ |
|---|---|---|
| $C_1$ | **True Positives (TP)** | **False Negatives (FN)** |
| ¬ $C_1$ | **False Positives (FP)** | **True Negatives (TN)** |

## Example of Confusion Matrix:

| Actual class\Predicted class | buy_computer = yes | buy_computer = no | Total |
|---|---|---|---|
| buy_computer = yes | **6954** | **46** | 7000 |
| buy_computer = no | **412** | **2588** | 3000 |
| Total | 7366 | 2634 | 10000 |

- Given *m* classes, an entry, $\boldsymbol{CM_{i,j}}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals

# Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

| A\P | C | ¬C | |
|---|---|---|---|
| C | **TP** | **FN** | **P** |
| ¬C | **FP** | **TN** | **N** |
| | **P'** | **N'** | **All** |

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified

  **Accuracy = (TP + TN)/P+N**

- **Error rate:** *1 – accuracy*, or

  **Error rate = (FP + FN)/P+N**

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - **Sensitivity**: True Positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/N**

# Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?

$$recall = \frac{TP}{TP + FN}$$

- Perfect score is 1.0
- ***F* measure (*$F_1$* or *F*-score)**: harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- ***$F_\beta$*:** weighted measure of precision and recall
  - assigns ß times as much weight to recall as to precision

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

# More about Precision and Recall

- Precession (Exactness): Suppose you have done duty for 10 hours, e.g., digging a well. How many hours out of total duty have you spent doing useful work? Ideally, you are supposed to spend 10 hours doing useful work. Total duty time (TP+FP) = time spent on useful work (TP)+ time spent on other work/mistakes (FP). Precision=TP/TP+FP
- Recall (completeness): Suppose the objective is to dig the well for 10 meters depth. You have spent some total time. Recall is about how much of the well is being dug. Total time = time spent digging work (TP) plus other work/mistakes (FN). Recall= TP/TP+FN.
- For a classifier, it is possible to have 100% precision and low recall. For example, you have spent 10 hours (the entire time) digging well. But, only a small portion of the well is being dug (recall is low).
- For a classifier, it is possible to have 100% recall and low precision. For example, you have dug the 10 meters well as required but have spent several days making mistakes (several wrong results).
- The objective is to get good precision and recall, which is a good F1 metric.

# Classifier Evaluation Metrics: Example

| Actual Class\Predicted class | cancer = yes | cancer = no | Total |
|---|---|---|---|
| cancer = yes | **90** | **210** | 300 |
| cancer = no | **140** | **9560** | 9700 |
| Total | 230 | 9770 | 10000 |

| A\P | C | ¬C | |
|---|---|---|---|
| C | **TP** | **FN** | **P** |
| ¬C | **FP** | **TN** | **N** |
| | **P'** | **N'** | **All** |

- Calculate the measure just introduced
  - Sensitivity = TP/P = 90/300 = 30%
  - Specificity = TN/N = 9560/9700 = 98.56%
  - Accuracy = (TP + TN)/All = (90+9560)/10000 = 96.50%
  - Error rate = (FP + FN)/All = (140 + 210)/10000 = 3.50%
  - Precision = TP/(TP + FP) = 90/(90 + 140) = 90/230 = 39.13%
  - Recall = TP/ (TP + FN) = 90/(90 + 210) = 90/300 = 30.00%
  - F1 = 2 P × R /(P + R) = 2 × 39.13% × 30.00%/(39.13% + 30%) = 33.96%

# Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- **Cross-validation** (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use $D_i$ as test set and others as training set
  - Leave-one-out: *k* folds where *k* = # of tuples, for small sized data
  - ***Stratified cross-validation*: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data (popular method)**

# Evaluating Classifier Accuracy: Bootstrap

- **Bootstrap**
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, and a common one is **.632 boostrap**
  - A data set with *d* tuples is sampled *d* times, with replacement, resulting in a training set of *d* samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedure *k* times, overall accuracy of the model:

$$Acc(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set})$$

# Estimating Confidence Intervals: Classifier Models $M_1$ vs. $M_2$

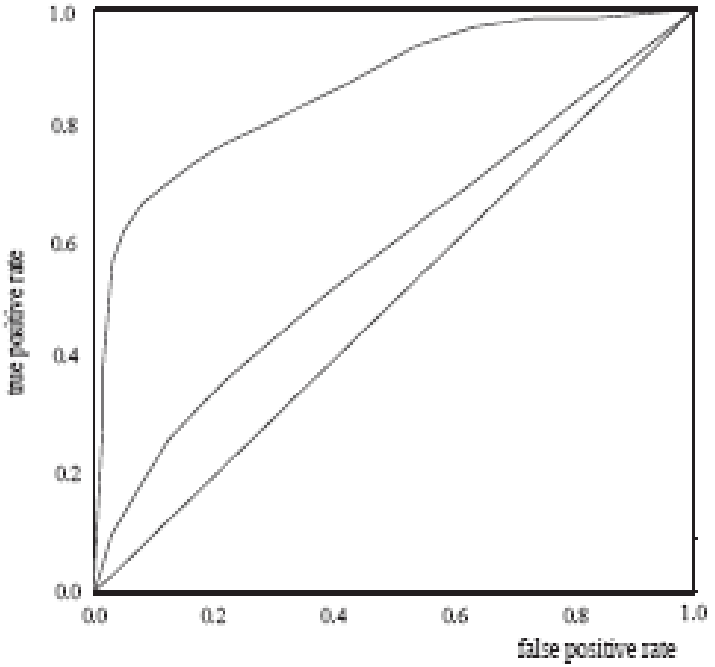
- Suppose we have 2 classifiers, $M_1$ and $M_2$, which one is better?
- Use 10-fold cross-validation to obtain $\overline{err}(M_1)$ and $\overline{err}(M_2)$
- These mean error rates are just *estimates* of error on the true population of *future* data cases
- What if the difference between the error rates is just attributed to *chance*?
  - Use a **test of statistical significance**
  - Obtain **confidence limits** for our error estimates

# Estimating Confidence Intervals: Null Hypothesis

- Perform 10-fold cross-validation
- Assume samples follow a **t distribution** with *k–1* **degrees of freedom**
- Use **t-test** (or **Student's t-test**)
- **Null Hypothesis**: $M_1$ & $M_2$ are the same
- If we can **reject** null hypothesis, then
  - we conclude that the difference between $M_1$ & $M_2$ is **statistically significant**
  - Chose model with lower error rate

# Model Selection: ROC Curves

- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Shows the trade-off between the true positive rate (TPR) and the false positive rate (FPR)
- *TPR* is the proportion of positive (or "yes") tuples that **are** correctly labeled by the model;
- FPR is the proportion of negative (or "no") tuples that are mislabeled as positive. Recall that T P, FP, P, and N are the number of true positive, false positive, positive, and negative tuples, respectively.
- Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model




- Vertical axis represents TPR
- Horizontal axis rep. the FPR
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

| Tuple # | Class | Prob. | TP | FP | TN | FN | TPR | FPR |
|---|---|---|---|---|---|---|---|---|
| 1 | P | 0.90 | 1 | 0 | 5 | 4 | 0.2 | 0 |
| 2 | P | 0.80 | 2 | 0 | 5 | 3 | 0.4 | 0 |
| 3 | N | 0.70 | 2 | 1 | 4 | 3 | 0.4 | 0.2 |
| 4 | P | 0.60 | 3 | 1 | 4 | 2 | 0.6 | 0.2 |
| 5 | P | 0.55 | 4 | 1 | 4 | 1 | 0.8 | 0.2 |
| 6 | N | 0.54 | 4 | 2 | 3 | 1 | 0.8 | 0.4 |
| 7 | N | 0.53 | 4 | 3 | 2 | 1 | 0.8 | 0.6 |
| 8 | N | 0.51 | 4 | 4 | 1 | 1 | 0.8 | 0.8 |
| 9 | P | 0.50 | 5 | 4 | 1 | 0 | 1.0 | 0.8 |
| 10 | N | 0.40 | 5 | 5 | 0 | 0 | 1.0 | 1.0 |

**FIGURE 6.20**

Tuples sorted by decreasing score, where the score is the value returned by a probabilistic classifier.

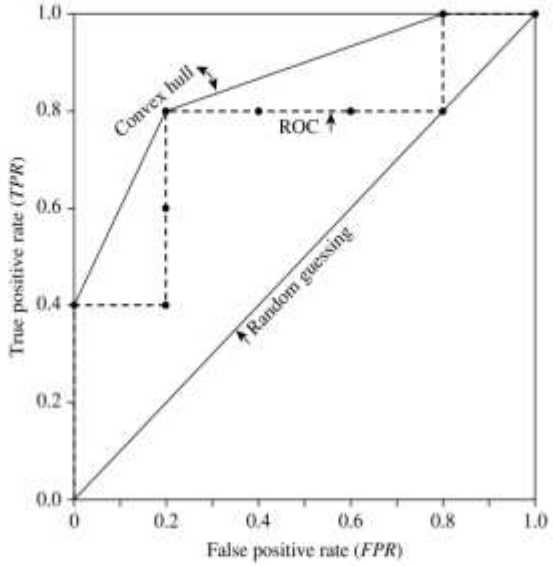



**FIGURE 6.21**

ROC curve for the data in Figure 6.20.

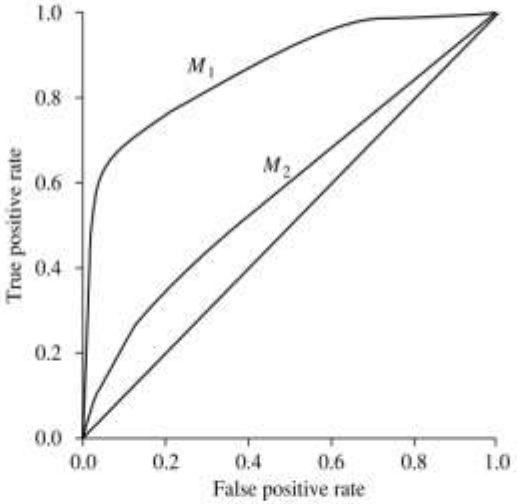



**FIGURE 6.22**

ROC curves of two classification models, $M_1$ and $M_2$. The diagonal shows where, for every true positive, we are equally likely to encounter a false positive. The closer a ROC curve is to the diagonal line, the less accurate the model is. Thus $M1$ is more accurate here.

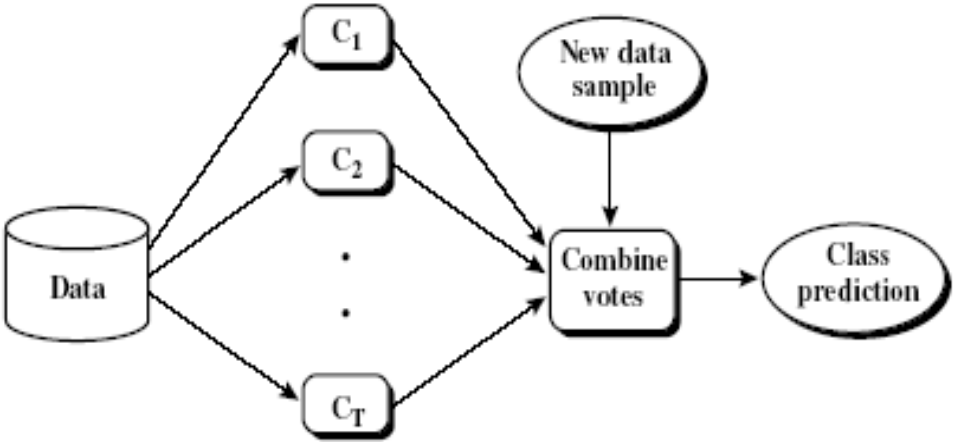
# Issues Affecting Model Selection

- **Accuracy**
  - classifier accuracy: predicting class label
- **Speed**
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- **Robustness**: handling noise and missing values
- **Scalability**: efficiency in disk-resident databases
- **Interpretability**
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

# Presentation Outline

- Background
- Basic concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Model Evaluation and Selection
- **Techniques to Improve Accuracy**
- Summary

# 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*

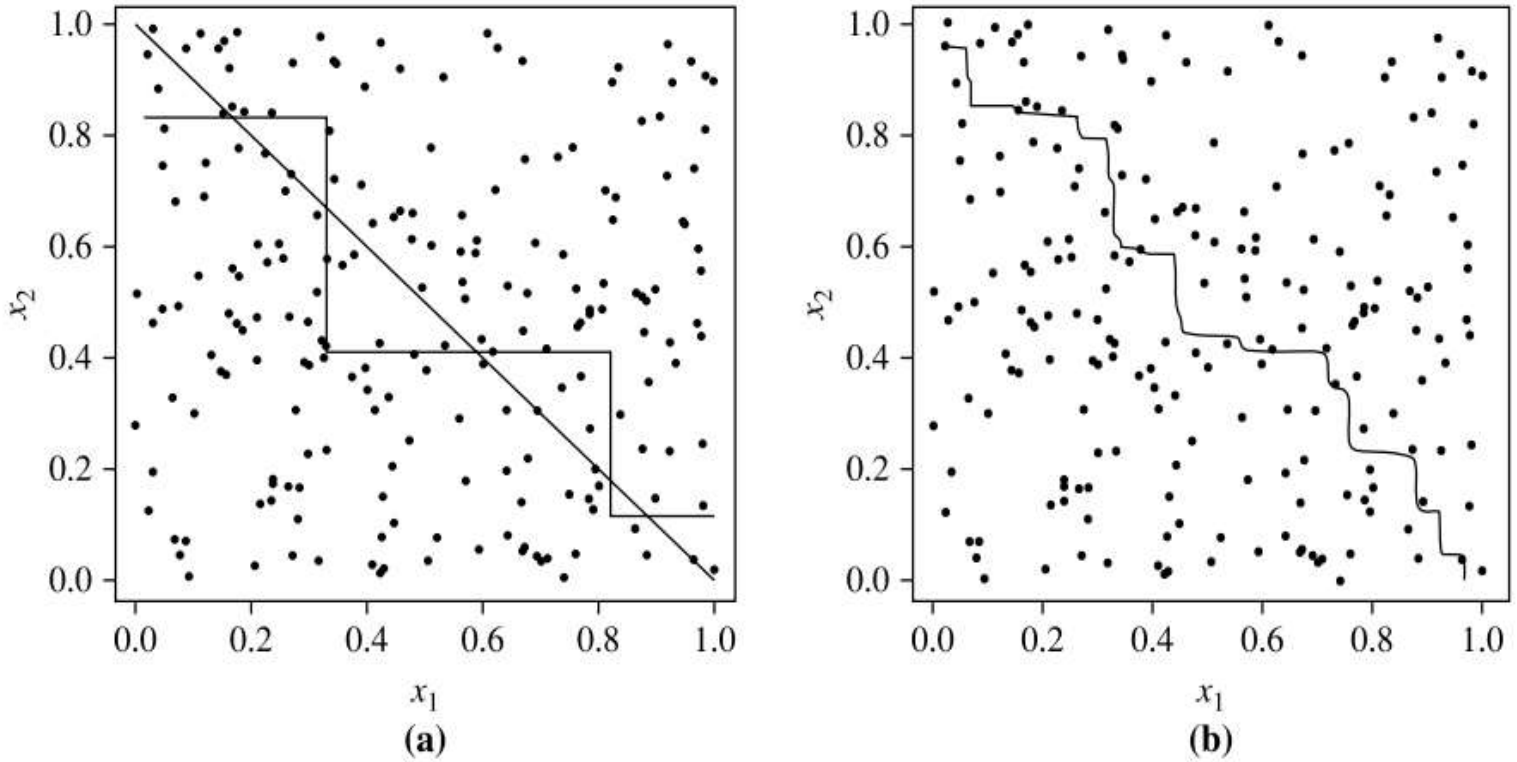
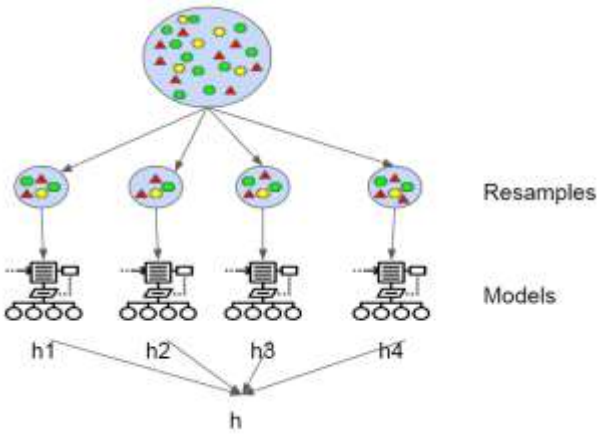



**FIGURE 6.24**

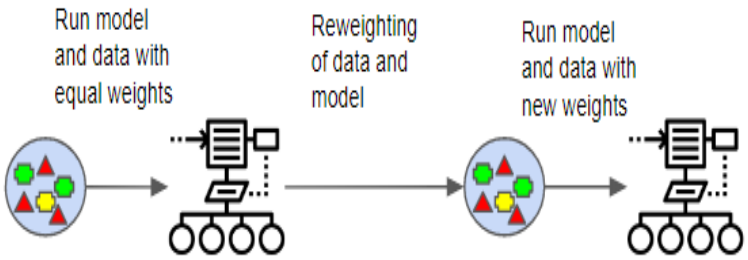
Decision boundary by (a) a single decision tree and (b) an ensemble of decision trees for a linearly separable problem (i.e., where the actual decision boundary is a straight line). The decision tree struggles with approximating a linear boundary. The decision boundary of the ensemble is closer to the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

# Ensemble Methods: Increasing the Accuracy

- Popular ensemble methods
  - Bagging: Trains each model using a subset of the training set, and models learned in parallel
  - Boosting: Trains each new model instance to emphasize the training instances that previous models mis-classified, and models learned in order




Bagging




Boosting

# Bagging: Bootstrap 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

**Algorithm: Bagging.** The bagging algorithm—create an ensemble of classification models for a learning scheme where each model gives an equally weighted prediction.

**Input:**

- $D$, a set of $d$ training tuples;
- $k$, the number of models in the ensemble;
- a classification learning scheme (e.g., decision tree algorithm, naïve Bayesian, etc.).

**Output:** The ensemble—a composite model, $M^*$.

**Method:**

(1) **for** $i = 1$ to $k$ **do** // create $k$ models:
(2) create bootstrap sample, $D_i$, by sampling $D$ with replacement;
(3) use $D_i$ and the learning scheme to derive a model, $M_i$.
(4) **endfor**

**To use the ensemble to classify a tuple, $X$:**

let each of the $k$ models classify $X$ and return the majority vote;

**FIGURE 6.25**

Bagging.

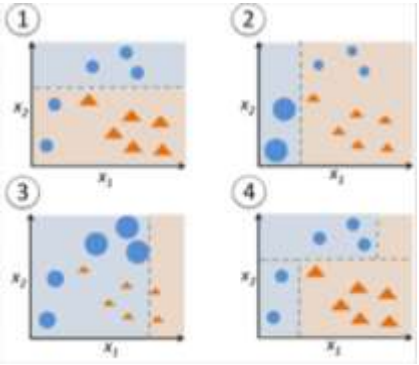
# Boosting

- 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

# Adaboost (Freund and Schapire, 1997)

1. Assign initial weights to each training tuple

3. Update weights based on current model




$\{w_n^{(1)}\}$ $\{w_n^{(2)}\}$ ... $\{w_n^{(k)}\}$

$M_1$ $M_2$ ... $M_k$

2. Train base classifier on weighted dataset

Two 'weighting' strategy:

1. Assign weights to each training example
2. Sample dataset based on weight distribution

$$M^*(x) = \text{sign}\left(\sum_{i=1}^{k} \alpha_i M_i(x)\right)$$

4. After base classifiers are trained, they are combined to give the final classifier

# Adaboost (Freund and Schapire, 1997)
# Adaptive boosting

- Given a set of *d* class-labeled tuples, ($\mathbf{X_1}$, $y_1$), …, ($\mathbf{X_d}$, $y_d$)
- 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:

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

- The weight of classifier $M_i$'s vote is $\log \frac{1-error(M_i)}{error(M_i)}$

**Algorithm: AdaBoost.** A boosting algorithm—create an ensemble of classifiers. Each one gives a weighted vote.

**Input:**

- $D$, a set of $d$ class-labeled training tuples;
- $k$, the number of rounds (one classifier is generated per round);
- a classification learning scheme.

**Output:** A composite model.

**Method:**

(1) initialize the weight of each tuple in $D$ to $1/d$;
(2) **for** $i = 1$ to $k$ **do** // for each round:
(3) sample $D$ with replacement according to the tuple weights to obtain $D_i$;
(4) use training set $D_i$ to derive a model, $M_i$;
(5) compute $error(M_i)$, the error rate of $M_i$ (Eq. (6.34))
(6) **if** $error(M_i) > 0.5$ **then**
(7) abort the loop;
(8) **endif**
(9) **for** each tuple in $D$ that was correctly classified **do**
(10) multiply the weight of the tuple by $error(M_i)/(1 - error(M_i))$; // update weights
(11) normalize the weight of each tuple.
(12) **endfor**

**To use the ensemble to classify tuple, $X$:**

(1) initialize weight of each class to 0;
(2) **for** $i = 1$ to $k$ **do** // for each classifier:
(3) $w_i = log \frac{1-error(M_i)}{error(M_i)}$; // weight of the classifier's vote
(4) $c = M_i(X)$; // get class prediction for $X$ from $M_i$
(5) add $w_i$ to the weight for class $c$
(6) **endfor**
(7) return the class with the largest weight.

**FIGURE 6.26**

AdaBoost, a boosting algorithm.

# Gradient Boosting and XGboost

- **Gradient boosting** is another powerful boosting technique, which can be used for classification, regression, and ranking.
  - If we use a tree (e.g., decision tree for classification, regression tree for regression) as the base model (i.e., the weak learner), it is called **gradient tree boosting**, or **gradient boosted tree**
- A highly scalable end-to-end gradient tree boosting system is called **XGBoost**, which is capable to handle a billion-scale training set.
- XGBoost has made a number of innovations for training gradient tree boosting, including a new tree construction algorithm designed for sparse data, feature subsampling (as opposed to training tuple subsampling in stochastic gradient boosting)
- A highly efficient cacheaware block structure. XGBoost has been successfully used by data scientists in many data mining challenges, often leading to top competitive results.

# Random Forest (Breiman 2001)

- 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

# Ensemble Methods Recap

- Random forest and XGBoost are the most commonly used algorithms for tabular data
- Pros
  - Good performance for tabular data, requires no data scaling
  - Can scale to large datasets
  - Can handle missing data to some extent
- Cons
  - Can overfit to training data if not tuned properly
  - Lack of interpretability (compared to decision trees)

# 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 so that the training set contains equal number of positive and negative samples.
  - **Under-sampling**: randomly eliminate tuples from negative class so that the training set contains equal number of positive and negative samples.
  - **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

# Presentation Outline

- Background
- Basic concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Model Evaluation and Selection
- Techniques to Improve Accuracy
- **Summary**

# Summary (I)

- Classification is a form of data analysis that extracts models describing important data classes.
- Effective and scalable methods have been developed for decision tree induction, Naive Bayesian classification, rule-based classification, and many other classification methods.
- Evaluation metrics include: accuracy, sensitivity, specificity, precision, recall, *F* measure, and $F_{\beta}$ measure.
- Stratified k-fold cross-validation is recommended for accuracy estimation. Bagging and boosting can be used to increase overall accuracy by learning and combining a series of individual models.

# Summary (II)

- Significance tests and ROC curves are useful for model selection.
- There have been numerous comparisons of the different classification methods; the matter remains a research topic
- No single method has been found to be superior over all others for all data sets
- Issues such as accuracy, training time, robustness, scalability, and interpretability must be considered and can involve trade-offs, further complicating the quest for an overall superior method

# Classification: Overview of Advanced Methods (Class #20)

P. Krishna Reddy, IIIT Hyderabad

# Topics

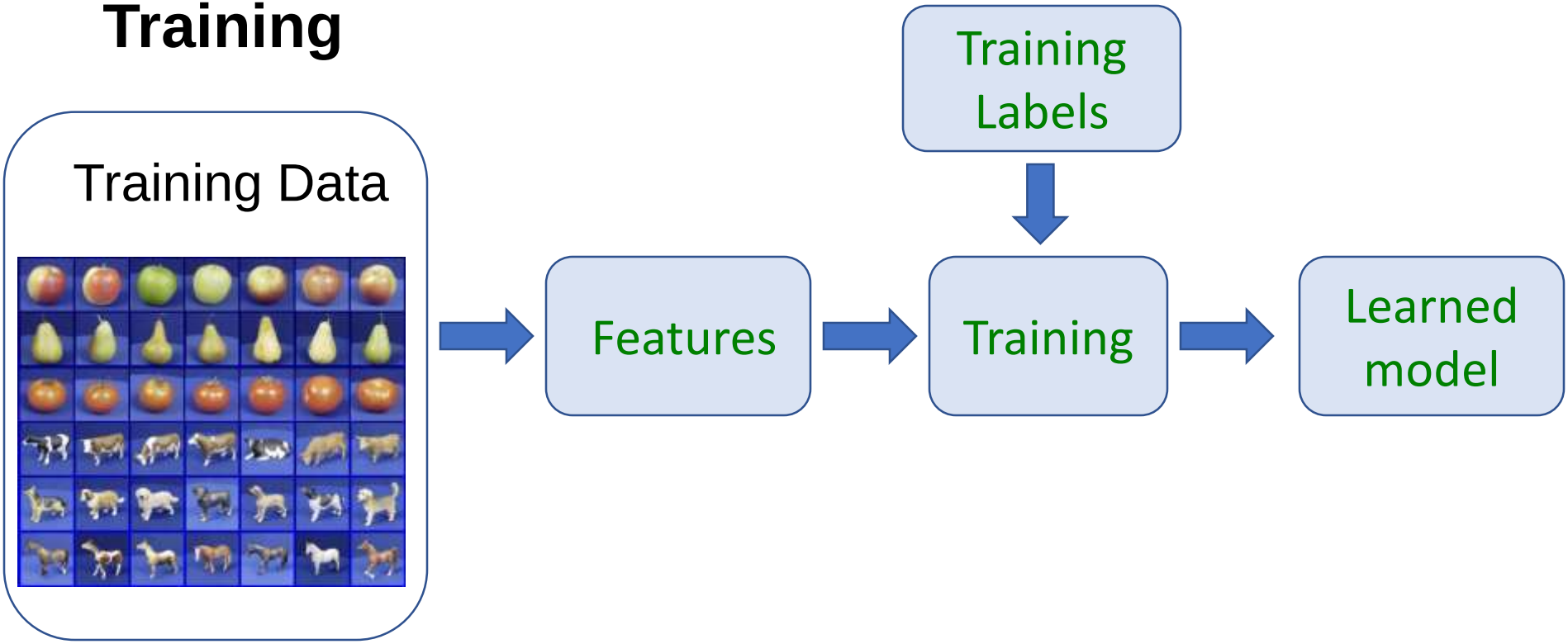
- Introduction (1.5 hour): Definition, KDD framework, Issues in data mining.
- Data summarization (7.5 hrs): Data Types, Preprocessing, Characterization, Discrimination, data warehousing techniques (Multidimensional data model, Data warehousing architecture, Data cube computation methods and OLAP technology)
- Concepts and algorithms for mining patterns and associations (9 hours) (Frequent item-set generation, A priori and FP-growth algorithm, Evaluation of Association patterns) and preprocessing, Overview of advanced mehods
- **Concepts and algorithms related to classification and regression (9hrs) (Overview, Decision tree induction, Over-fitting and under-fitting, Scalable decision tree algorithms, Bayesian Classification, Regression-based Prediction methods, Overview of Advanced Methods (9 hours)**
- Concepts and algorithms for clustering the data (9 hours) (Overview, Types of Data, K-means, Aglomerative clustering, Clustering algorithms (DBSCAN, BIRCH, CURE, ROCK, CHAMELEON)).
- Outlier analysis and future trends (graph mining, spatio-temporal mining). (3 hours)

# Outline

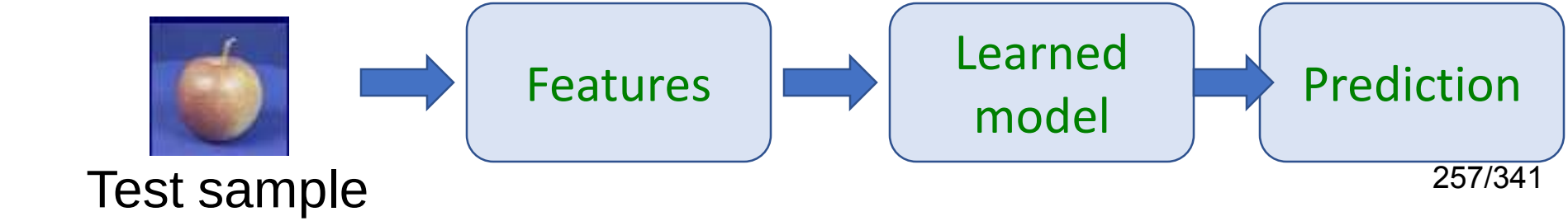
- **Feature Selection and Engineering**
- Bayesian Belief Networks
- Support Vector Machines
- Rule based and pattern based classification
- Classification by weak supervision
- Classification with rich data type
- Other Related  Methods
- Summary

# Steps

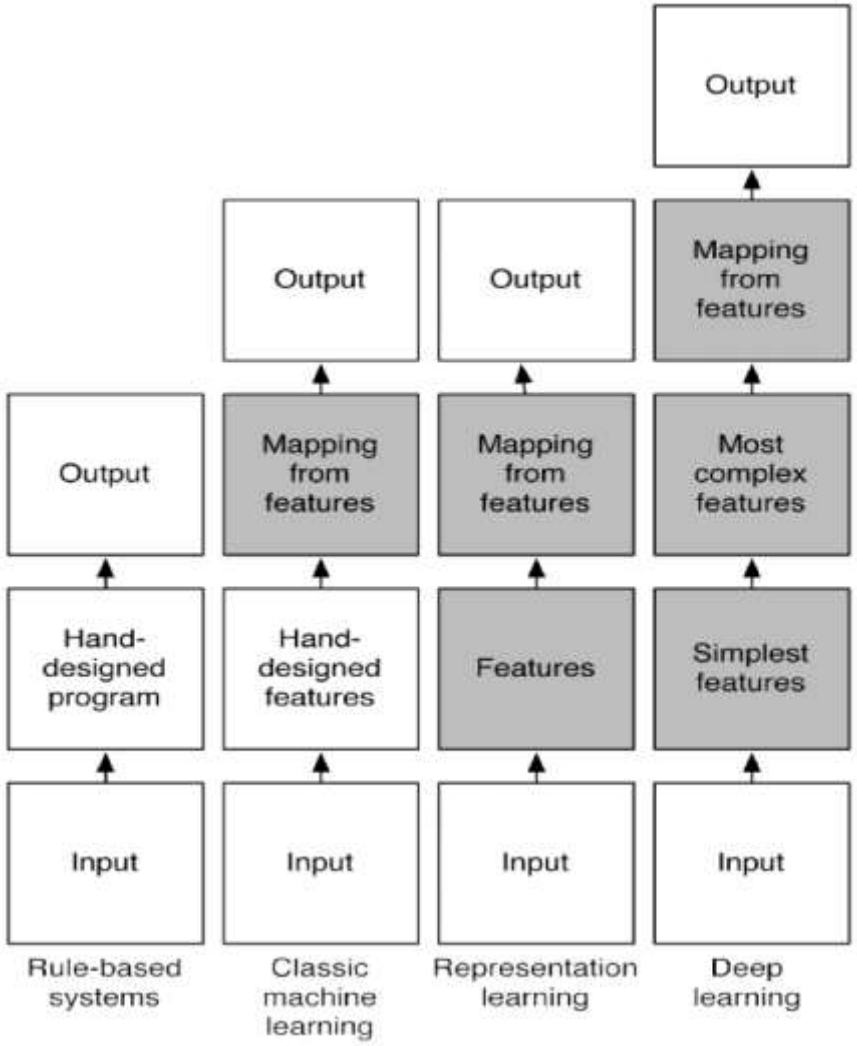
## Training




## Testing



# What is deep learning?




Y. Bengio et al, ``Deep Learning", MIT Press, 2015

# Tabular Data

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

# Issues

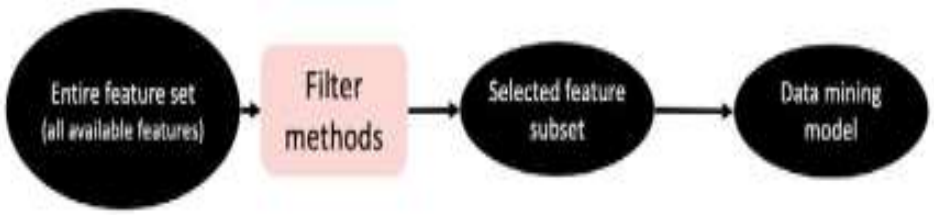
- Assumption: attributes are known
  - Realty: Attributes are unknown
- Assumption: Training set is given
  - Realty:  Training set is not given

# About feature selection and Engineering

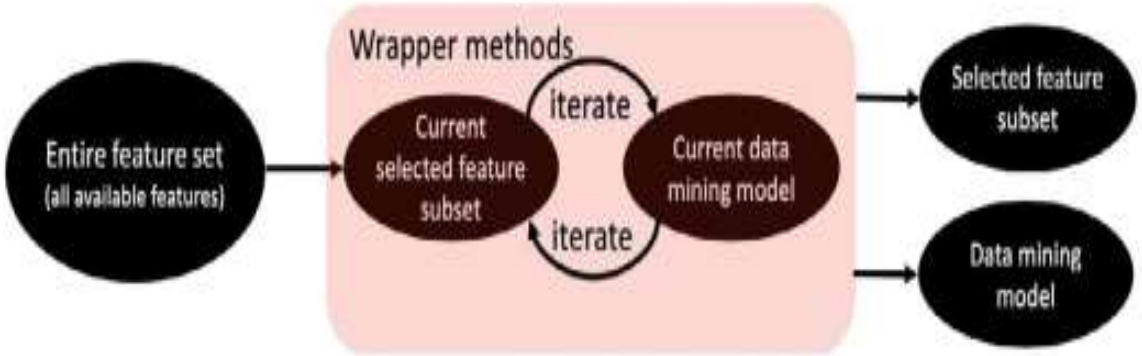
- So far, we assumed
  - A training set of with n tuples for p attributes (features) is available.
- Question, where do these features have come?
- **Feature selection:**
  - Large number of features are collected (thousands or more)
  - Question: how to select a subset of features from initial set of p features to improve generalization (classification) performance?
  - Example:
    - Student ID is irrelevant to predict whether a student will drop the course.
    - Monthly income and yearly income of redundant features to predict "Buy Computer"
  - Irrelevant attributes make classifier sensitive to noise.
- **Feature Engineering:** How can I construct *p* features so that all of them are critical for the classification task I have?
- Given set of p features, can I transform them to p' attributes so that these transformed features will be more effective for classification?
- Example: Health surveillance data which includes number of daily positive cases, number of daily tests, number of daily hospitalization.
  - It has turned out that weekly positive rate is powerful indicator for disease outbreak.
  - It can be derived from the daily data

# Feature selection methods

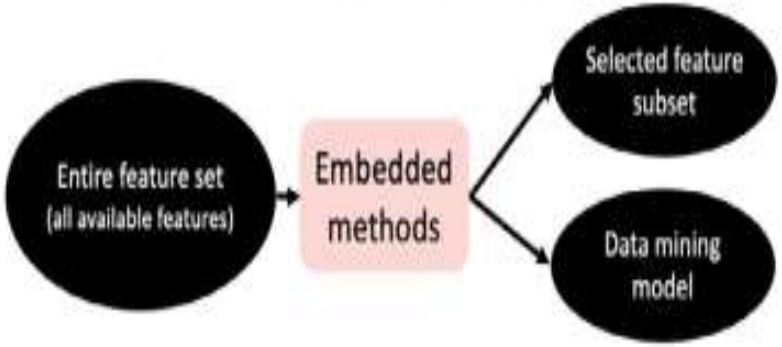
- Filter methods
  - Independent of classification model
- Wrapper methods
  - Combines feature selection and classifier model
- Embedded methods
  - Embeds the feature selection model during the classification step




(a) Filter methods




(b) Wrapper methods




(c) Embedded methods

# Filter Method

- Selects features based on certain goodness measure of the input features
- Have a goodness score for each feature and select features with highest goodness score.
- Method 1: A feature is good if it highly correlated with the class label we want to predict.
  - Higher chi-square value indicates stronger corelation.
  - Select k features with highest chi-square values.
  - For continuous attribute
    - Option 1: discretize and compute chi-square
    - Option 2: Compute Fisher score (computes correlation between continuous value and a categorical attribute
- Method 2: Information theoretic measures
  - Information gain
- Limitations
  - Independent of classification model
  - Does not consider the interaction between the attributes.

# Wrapper method

- It is an iterative process
  - In each iteration
    - Build a classifier based on currently selected feature set and build a classifier
    - Update the feature set (add, remove, swap)
- Straightforward method
  - Try all subsets of p features = 2^p-1 → exponential!
- Stepwise forward selection method
  - Start with empty set.
  - Iteratively, include additional feature if it improves accuracy.
- Stepwise backward selection
  - Start with initial P features
  - Iteratively eliminate a feature, if it improves accuracy
- We can combine both
- Other techniques
  - Genetic algorithms
  - Simulated annealing: probabilistic algorithm designed for nonconvex optimization problem

# Embedded methods

- Combines advantages of filter methods and wrapper methods
- Embedded method performs feature selection and classification model construction simultaneously.
- Example: decision tree
  - After termination, all features at non-leaf nodes are selected features.
- Sparse Learning
  - We build data mining models by minimizing some objective function.
  - Example:
    - Least square linear regression: we want to find a weight vector w by minimizing the sum of the squared difference between the predicted output and the actual output.
    - Logistic regression: find the optimal vector w by minimizing the negative log likelihood.
- By making the corresponding weight of the variable zero, we can remove the feature and calculate the loss function.

# Outline

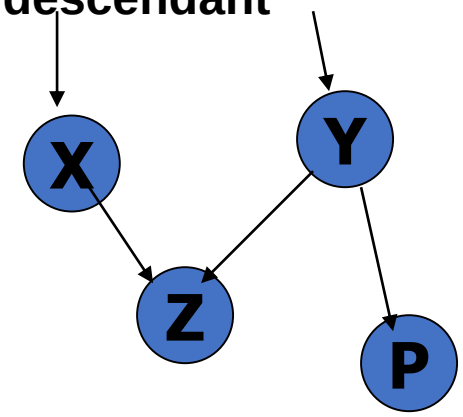
- Feature Selection and Engineering
- **Bayesian Belief Networks**
- Support Vector Machines
- Rule based and pattern based classification
- Classification by weak supervision
- Classification with rich data type
- Other Related  Methods
- Summary

# Bayesian belief networks

- Assumption of Bayesian classifier is class conditional independence
  - Given the class label of a tuple, the values of the attributes are assumed to be conditionally independent of one another. This simplifies computation.
- When the assumption holds true, then the naïve Bayesian classifier is the most accurate in comparison with all other classifiers.
- In practice, however, dependencies can exist between variables.
- **Bayesian belief networks** specify joint conditional probability distributions.
  - allow class conditional independencies to be defined between subsets of variables.
  - Provide a graphical model of causal relationships, on which learning can be performed.
- Trained Bayesian belief networks can be used for classification.
- Other names are **belief networks**, **Bayesian networks**, and **probabilistic networks**.
- We will refer to them as belief networks

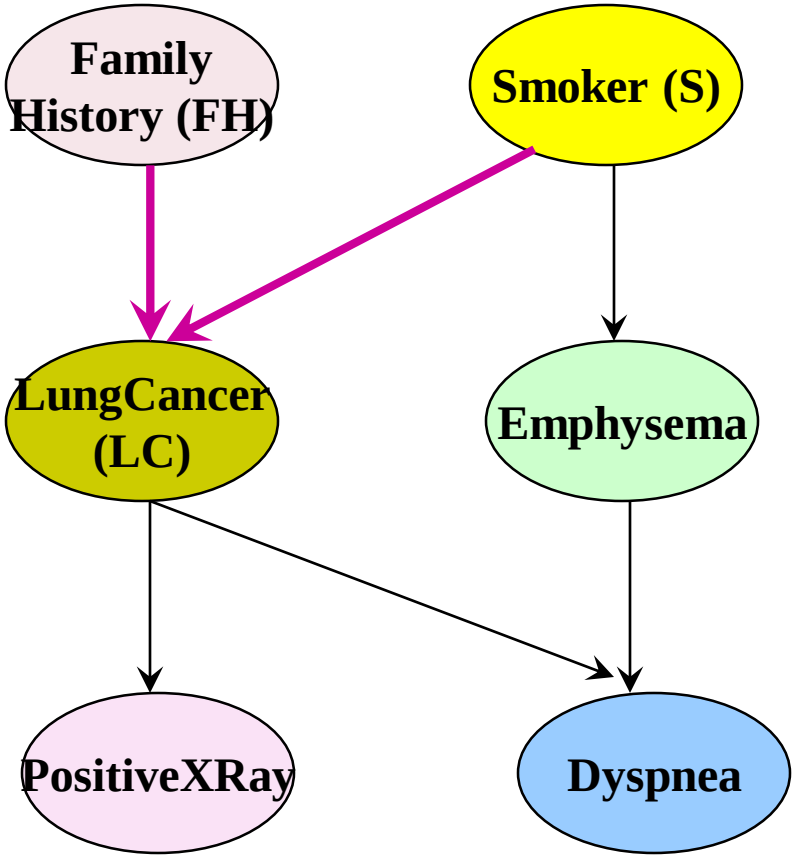
# Bayesian Belief Networks

- A belief network is defined by two components—
  - a directed acyclic graph and a set of conditional probability tables.
- Each node in the directed acyclic graph represents a random variable.
  - The variables may be discrete- or continuous-valued.
  - They may correspond to actual attributes given in the data or to "hidden variables" believed to form a relationship
  - About hidden variables
    - In the case of medical data, a hidden variable may indicate a syndrome, representing a number of symptoms that, together, characterize a specific disease.
- Each arc represents a probabilistic dependence. If an arc is drawn from a node *Y* to a node *Z*, then *Y* is a **parent** or **immediate predecessor** of *Z*, and *Z* is a **descendant**




- Nodes: random variables
- Links: dependency
- X and Y are the parents of Z, and Y is the parent of P
- No dependency between Z and P
- Has no loops/cycles

# Bayesian Belief Network: An Example




**Bayesian Belief Network**

**CPT**: **Conditional Probability Table** for variable LungCancer:

| | (FH, S) | (FH, ~S) | (~FH, S) | (~FH, ~S) |
|---|---|---|---|---|
| LC | 0.8 | 0.5 | 0.7 | 0.1 |
| ~LC | 0.2 | 0.5 | 0.3 | 0.9 |

shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of **X**, from CPT:

$$P(x_1,\ldots,x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(Y_i))$$

# About Example

- The arcs allow the representation of casual knowledge
  - For example, having lung cancer is influenced by a person's family history of lung cancer, as well as whether or not the person is a smoker.
  - Note that the variable PositiveXRay is independent of whether the patient has a family history of lung cancer or is a smoker, given that we know the patient has lung cancer.
- In other words, once we know the outcome of the variable LungCancer, then the variables FamilyHistory and Smoker do not provide any additional information regarding PositiveXRay.
- The arcs also show that the variable LungCancer is conditionally independent of Emphysema, given its parents, FamilyHistory and Smoker

# About Conditional Probability Table (CPT)

- The CPT for a variable *Y* specifies the conditional distribution *P(Y/Parents(Y))*, where *Parents(Y)* are the parents of *Y*.
- in CPT, the conditional probability for each known value of *LungCancer* is given for each possible combination of the values of its parents.
- Example

*P(LungCancer =yes| FamilyHistory =yes, Smoker =yes)*= 0.8
*P(LungCancer =no| FamilyHistory =no, Smoker =no)*= 0.9.

Let $\boldsymbol{X} = (x_1, \ldots, x_n)$ be a data tuple described by the variables or attributes $Y_1, \ldots, Y_n$, respectively. Recall that each variable is conditionally independent of its nondescendants in the network graph, given its parents. This allows the network to provide a complete representation of the existing joint probability distribution with the following equation:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | Parents(Y_i)), \qquad (9.1)$$

where $P(x_1, \ldots, x_n)$ is the probability of a particular combination of values of $\boldsymbol{X}$, and the values for $P(x_i|Parents(Y_i))$ correspond to the entries in the CPT for $Y_i$.

# Applications

- Genetic linkage analysis (e.g., the mapping of genes onto a chromosomes.
- Computer vision (e.g., image restoration and stereo vision)
- document and text analysis
- decision support systems
- sensitivity analysis.

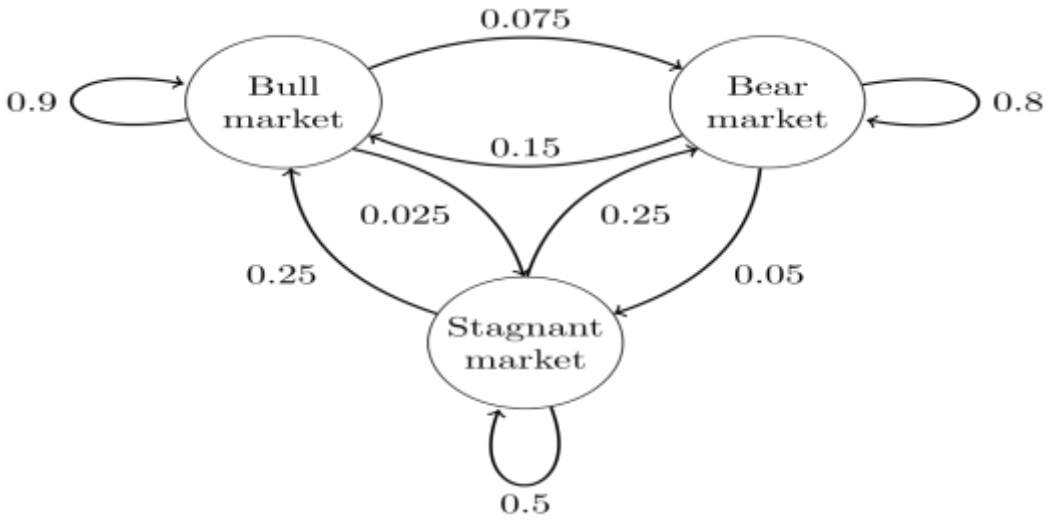
# Training Bayesian Networks: Several Scenarios

- **Scenario 1: Given both the network structure and all variables observable: *compute only the CPT entries***
  - **Similar to naïve Bayesian classification**
- Scenario 2: Network structure known, some variables hidden: *gradient descent* (greedy hill-climbing) method, i.e., search for a solution along the steepest descent of a criterion function
  - Weights are initialized to random probability values
  - At each iteration, it moves towards what appears to be the best solution at the moment, w.o. backtracking
  - Weights are updated at each iteration & converge to local optimum
- Scenario 3: Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
- Scenario 4: Unknown structure, all hidden variables: No good algorithms known for this purpose
- D. Heckerman. A Tutorial on Learning with Bayesian Networks. In *Learning in Graphical Models*, M. Jordan, ed.. MIT Press, 1999.

# Markov Chains

- Undirected graph
- A Markov chain is **a stochastic model that uses mathematics to predict the probability of a sequence of events occurring based on the most recent event**.
- Memory less
- Example 1: A common example of a Markov chain in action is the way Google predicts the next word in your sentence based on your previous entry within Gmail

- Example 2:The states represent whether a hypothetical stock market is exhibiting a bull market, bear market, or stagnant market trend during a given week. According to the figure, a bull week is followed by another bull week 90% of the time, a bear week 7.5% of the time, and a stagnant week the other 2.5% of the time.



# Outline

- Feature Selection and Engineering
- Bayesian Belief Networks
- **Support Vector Machines**
- Rule based and pattern based classification
- Classification by weak supervision
- Classification with rich data type
- Other Related  Methods
- Summary

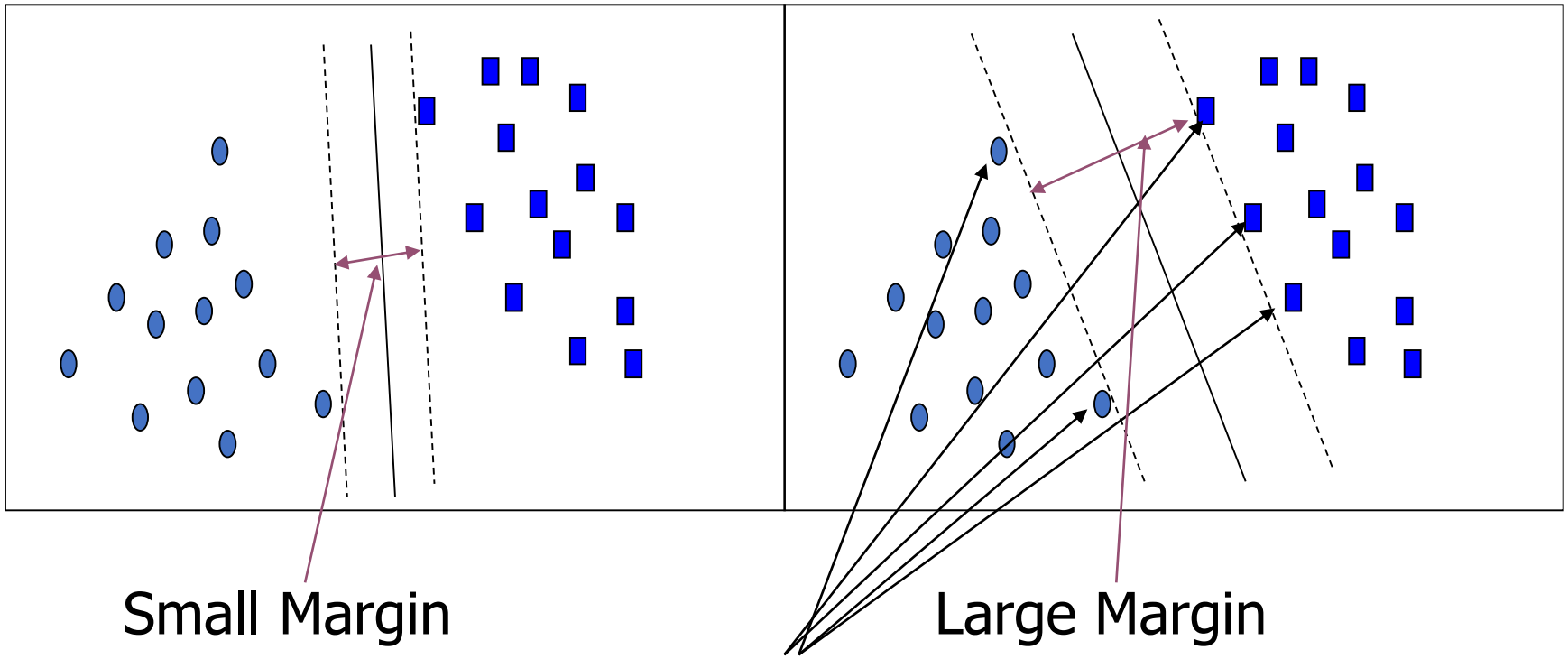
# SVM—Support Vector Machines

- A relatively new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating **hyperplane** (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using **support vectors** ("essential" training tuples) and **margins** (defined by the support vectors)

# SVM—History and Applications

- Vapnik and colleagues (1992)—groundwork from Vapnik & Chervonenkis' statistical learning theory in 1960s
- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- Used for: classification and numeric prediction
- Applications:
  - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

# SVM—General Philosophy



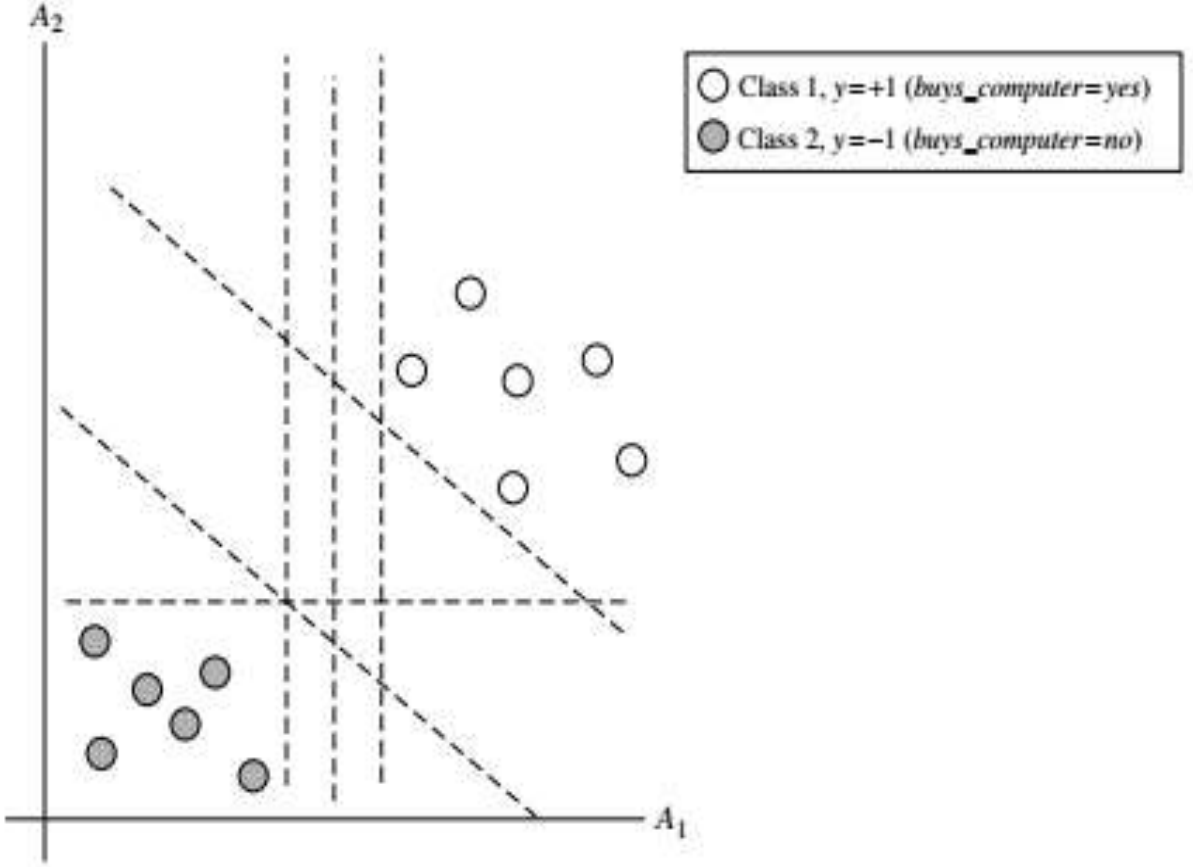



**Figure 9.7** The 2-D training data are linearly separable. There are an infinite number of possible separating hyperplanes or "decision boundaries," some of which are shown here as dashed lines. Which one is best?

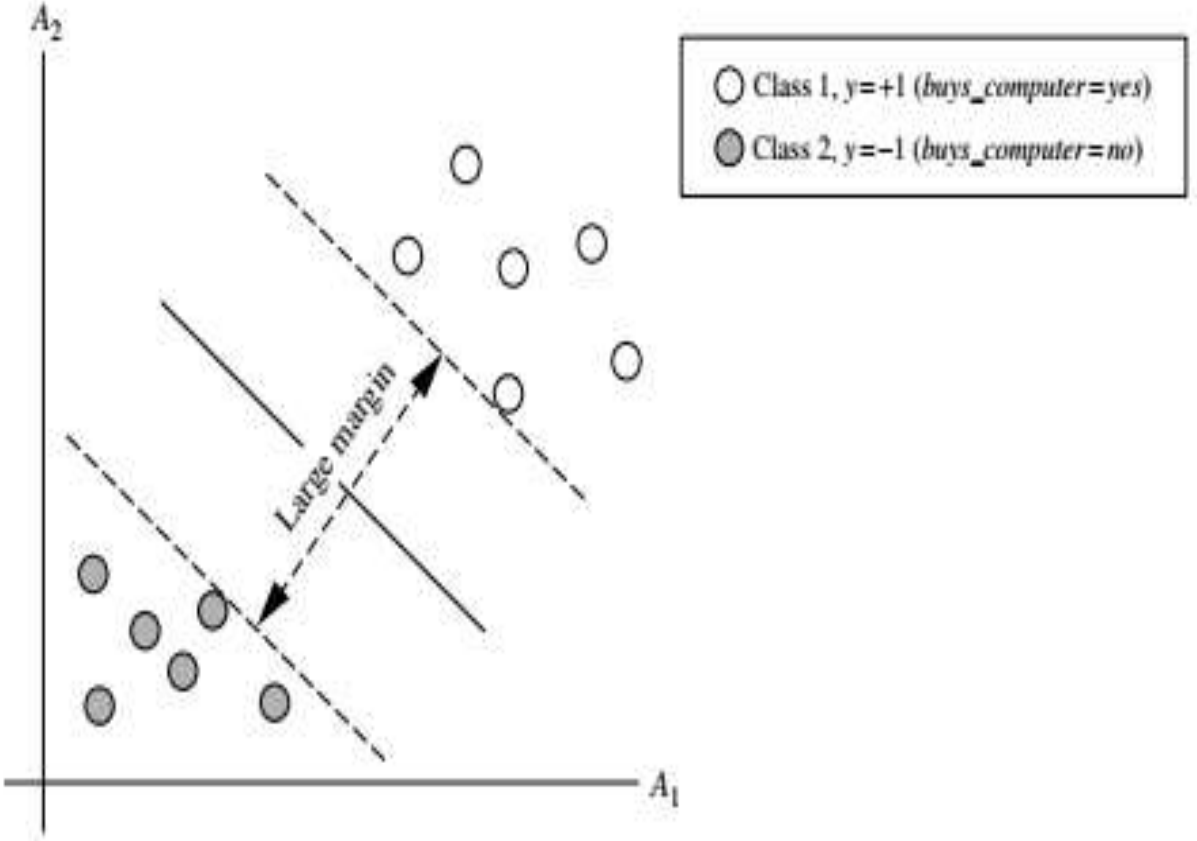
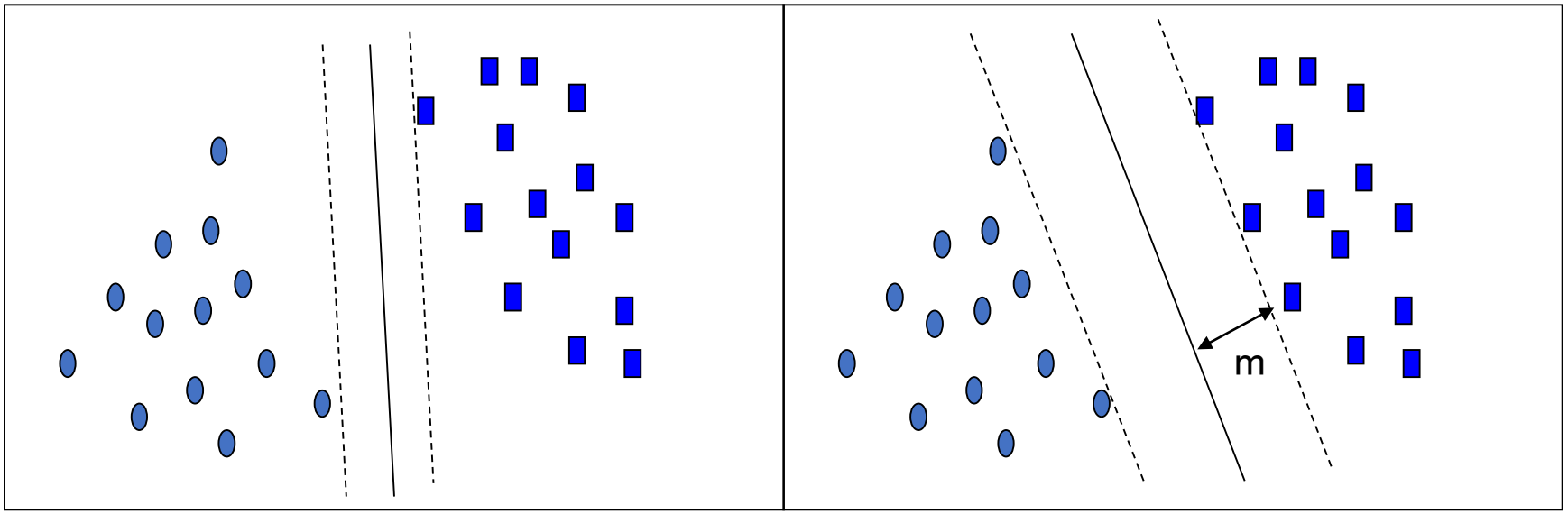



**Figure 9.9** Support vectors. The SVM finds the maximum separating hyperplane, that is, the one with maximum distance between the nearest training tuples. The support vectors are shown with a thicker border.

# SVM—When Data Is Linearly Separable

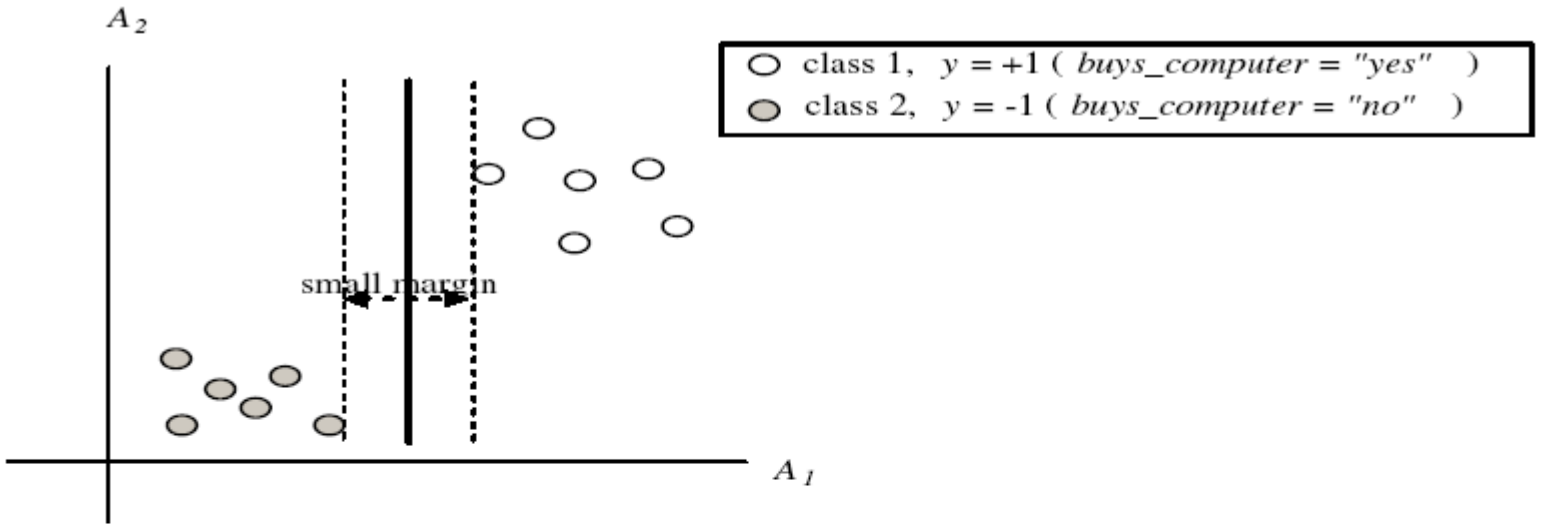


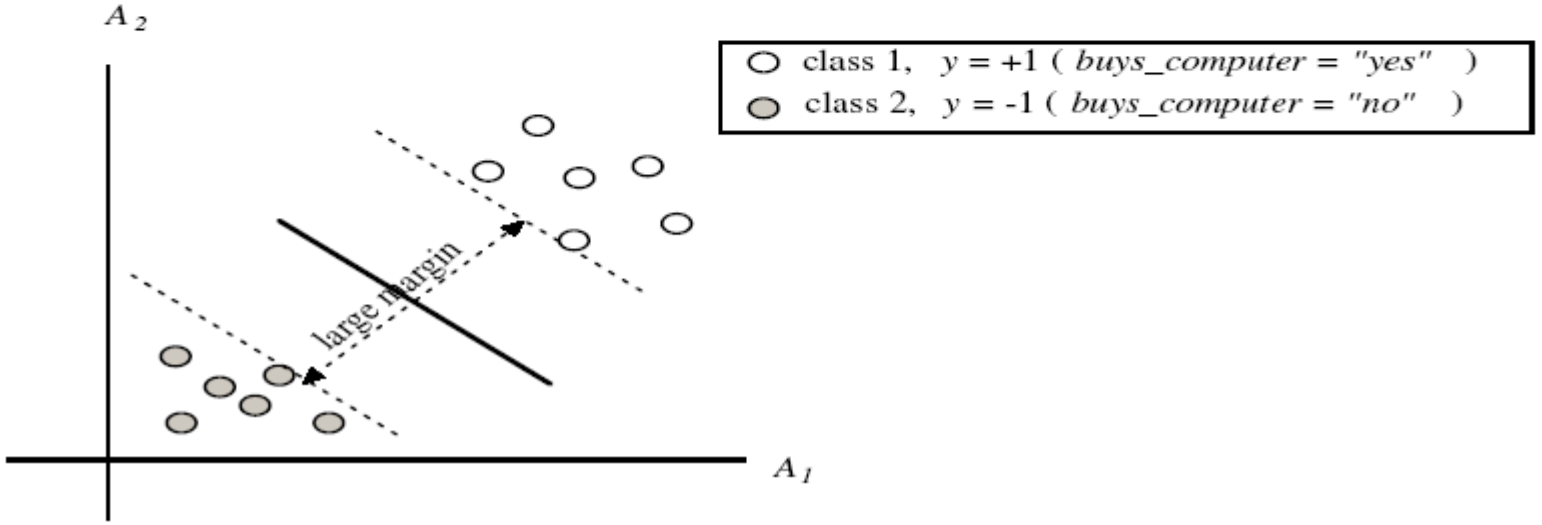

Let data D be $(\mathbf{X}_1, y_1), \ldots, (\mathbf{X}_{|D|}, y_{|D|})$, where $\mathbf{X}_i$ is the set of training tuples associated with the class labels $y_i$

There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data)

*SVM searches for the hyperplane with the largest margin*, i.e., **maximum marginal hyperplane** (MMH)

# SVM—Margins and Support Vectors



# SVM—Linearly Separable

- A separating hyperplane can be written as

  $\mathbf{W} \bullet \mathbf{X} + b = 0$

  where $\mathbf{W}=\{w_1, w_2, \ldots, w_n\}$ is a weight vector and b a scalar (bias)
- For 2-D it can be written as

  $w_0 + w_1 x_1 + w_2 x_2 = 0$
- The hyperplane defining the sides of the margin:

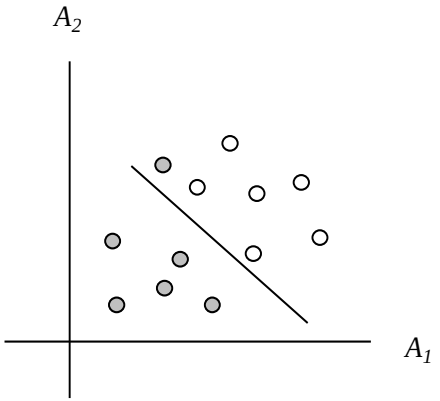
  $H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

  $H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$
- Any training tuples that fall on hyperplanes $H_1$ or $H_2$ (i.e., the sides defining the margin) are **support vectors**
- This becomes a **constrained (convex) quadratic optimization** problem: Quadratic objective function and linear constraints → *Quadratic Programming (QP)* → Lagrangian multipliers method

# Why Is SVM Effective on High Dimensional Data?

- The **complexity** of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The **support vectors** are the essential or critical training examples —they lie closest to the decision boundary, maximum marginal hyperplane (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

# SVM—Linearly Inseparable




- Transform the original input data into a higher dimensional space

Example 6.8 Nonlinear transformation of original input data into a higher dimensional space. Consider the following example. A 3D input vector $\mathbf{X} = (x_1, x_2, x_3)$ is mapped into a 6D space $Z$ using the mappings $\phi_1(X) = x_1, \phi_2(X) = x_2, \phi_3(X) = x_3, \phi_4(X) = (x_1)^2, \phi_5(X) = x_1x_2$, and $\phi_6(X) = x_1x_3$. A decision hyperplane in the new space is $d(\mathbf{Z}) = \mathbf{WZ} + b$, where $\mathbf{W}$ and $\mathbf{Z}$ are vectors. This is linear. We solve for $\mathbf{W}$ and $b$ and then substitute back so that we see that the linear decision hyperplane in the new ($\mathbf{Z}$) space corresponds to a nonlinear second order polynomial in the original 3-D input space,

$$\begin{aligned} d(Z) &= w_1x_1 + w_2x_2 + w_3x_3 + w_4(x_1)^2 + w_5x_1x_2 + w_6x_1x_3 + b \\ &= w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5 + w_6z_6 + b \end{aligned}$$

- Search for a linear separating hyperplane in the new space
- Computation is costly.

# Using SVM for classification

"*Once I've got a trained support vector machine, how do I use it to classify test (i.e., new) tuples?*" Based on the Lagrangian formulation mentioned before, the MMH (maximum marginal hyperplane ) can be rewritten as the decision bou

$$d(X^T) = \sum_{i=1}^{l} y_i \alpha_i X_i X^T + b_0, \tag{9.19}$$

where $y_i$ is the class label of support vector $X_i$; $X^T$ is a test tuple; $\alpha_i$ and $b_0$ are numeric parameters that were determined automatically by the optimization or SVM algorithm noted before; and $l$ is the number of support vectors.

Given a test tuple, $X^T$, we plug it into Eq. (9.19), and then check to see the sign of the result. This tells us on which side of the hyperplane the test tuple falls. If the sign is positive, then $X^T$ falls on or above the MMH, and so the SVM predicts that $X^T$ belongs to class +1 (representing *buys_computer* = *yes*, in our case). If the sign is negative, then $X^T$ falls on or below the MMH and the class prediction is −1 (representing *buys_computer* = *no*).

# SVM vs. Neural Network

- **SVM**
  - Deterministic algorithm
  - Nice generalization properties
  - Hard to learn – learned in batch mode using quadratic programming techniques
  - Using kernels can learn very complex functions

- **Neural Network**
  - Nondeterministic algorithm
  - Generalizes well but doesn't have strong mathematical foundation
  - Can easily be learned in incremental fashion
  - To learn complex functions—use multilayer perceptron (nontrivial)

# SVM Related Links

- SVM Website: http://www.kernel-machines.org/
- Representative implementations
  - **LIBSVM**: an efficient implementation of SVM, multi-class classifications, nu-SVM, one-class SVM, including also various interfaces with java, python, etc.
  - **SVM-light**: simpler but performance is not better than LIBSVM, support only binary classification and only in C
  - **SVM-torch**: another recent implementation also written in C

# Outline

- Feature Selection and Engineering
- Bayesian Belief Networks
- Support Vector Machines
- **Rule based and pattern based classification**
- Classification by weak supervision
- Classification with rich data type
- Other Related  Methods
- Summary

# Rule-based classification

- A classifier is a set of rules
- A rule based classifier uses a set of IF THEN rules for classification.
- An IF-THEN rule is an expression of the form

  IF condition THEN conclusion

- Example

R1: IF age=youth AND student=yes THEN buys_computer= yes

- The IF part of the rule is antecedent or precondition
- The THEN part of the rule is the rule consequent.

# Using IF-THEN Rules for Classification

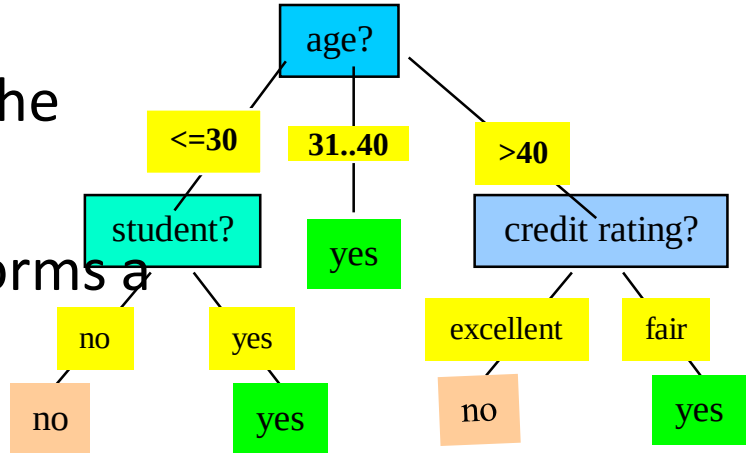
- Assessment of a rule: *coverage* and *accuracy*
  - $n_{covers}$ = # of tuples covered by R
  - $n_{correct}$ = # of tuples correctly classified by R

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

  accuracy(R) = $n_{correct}$ / $n_{covers}$
- 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




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

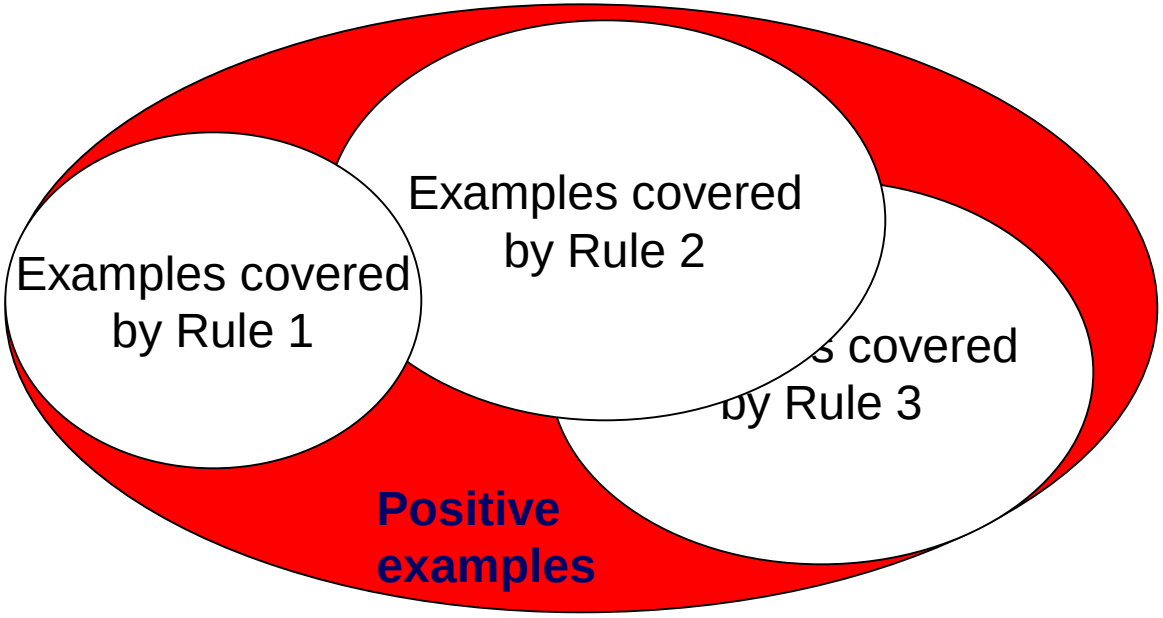
IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*
IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*
IF *age* = mid-age THEN *buys_computer* = *yes*
IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *no*
IF *age* = old AND *credit_rating* = *fair*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

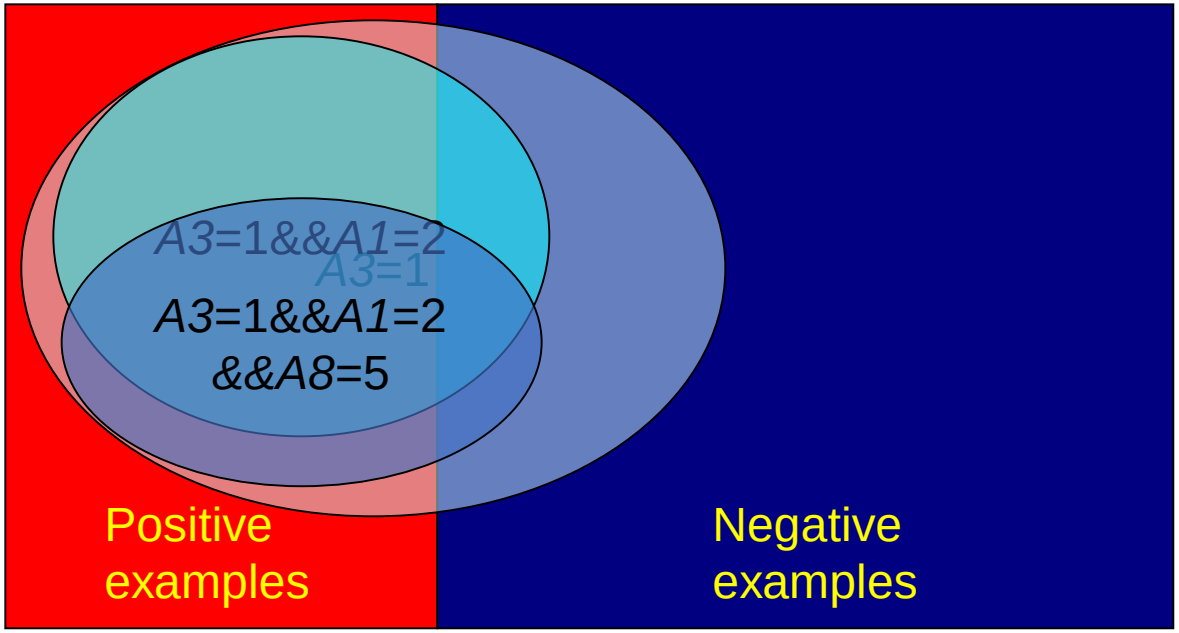
- To generate a rule

  **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
- FOIL- First Order Inductive Learner
- Rule-Quality measures: consider both coverage and accuracy
  - Let **pos** and **neg** be the number of positive and negative tuples learned by R and **pos'** and **neg'** be the number of positive and negative tuples learned by R'
  - Foil-gain (in FOIL & RIPPER): assesses info_gain by extending condition

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

    - favors rules that have high accuracy and cover many positive tuples
- Rule pruning based on an independent set of test tuples. Given R

$$FOIL\_Prune(R) = \frac{pos - neg}{pos + neg}$$

Pos/neg are # of positive/negative tuples covered by R.
If *FOIL_Prune* is higher for the pruned version of R, prune R

# Associative Classification

- Associative classification: Major steps
  - Mine data to find strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels
  - Association rules are generated in the form of

    $P_1 \wedge p_2 \ldots \wedge p_l \rightarrow$ "$A_{class} = C$" (conf, sup)
  - Organize the rules to form a rule-based classifier
- Why effective?
  - It explores highly confident associations among multiple attributes and may overcome some constraints introduced by decision-tree induction, which considers only one attribute at a time
  - Associative classification has been found to be often more accurate than some traditional classification methods, such as C4.5

# Typical Associative Classification Methods

- **CBA** (Classification Based on Associations: Liu, Hsu & Ma, KDD'98)
  - Mine possible association rules in the form of
    - Cond-set (a set of attribute-value pairs) → class label
  - Build classifier: Organize rules according to decreasing precedence based on confidence and then support
- **CMAR** (Classification based on Multiple Association Rules: Li, Han, Pei, ICDM'01)
  - Classification: Statistical analysis on multiple rules
- **CPAR** (Classification based on Predictive Association Rules: Yin & Han, SDM'03)
  - Generation of predictive rules (FOIL-like analysis) but allow covered rules to retain with reduced weight
  - Prediction using best k rules
  - High efficiency, accuracy similar to CMAR

# Quiz 2: Till the preceding slide

# Discriminative Frequent Pattern Classification

- This topic is not included.

# Outline

- Feature Selection and Engineering
- Bayesian Belief Networks
- Support Vector Machines
- Rule based and pattern based classification
- **Classification by weak supervision**
- Classification with rich data type
- Other Related  Methods
- Summary

# Weak Supervision

- SVM, Logistic regression, k-NN
  - Strong supervision: require large number of high quality training tuples annotated by domain experts.
- In real work, training set is not available.
  - Document classification, speech recognition, computer vision
  - One minute of speech requires 10 minutes to label
  - For annotating sounds, it takes 400 times more
- Classification approaches are required for limited number of training samples or no training samples.

# Semi-Supervised Classification

- Semi-supervised: Uses labeled and unlabeled data to build a classifier
- Self-training:
  - Build a classifier using the labeled data
  - Use it to label the unlabeled data, and those with the most confident label prediction are added to the set of labeled data
  - Repeat the above process
  - Adv: easy to understand; disadv: may reinforce errors
- Co-training: Use two or more classifiers to teach each other
  - Each learner uses a mutually independent set of features of each tuple to train a good classifier, say f1 and f2
  - Then f1 and f2 are used to predict the class label for unlabeled data X
  - **Teach each other: The tuple having the most confident prediction from $f_1$ is added to the set of labeled data for $f_2$, & vice versa**

**Self-training**

1. Select a learning method such as Bayesian classification. Build the classifier using the labeled data, $X_l$.
2. Use the classifier to label the unlabeled data, $X_u$.
3. Select the tuple $x \in X_u$ having the highest confidence (most confident prediction). Add it and its predicted label to $X_l$.
4. Repeat (i.e., retrain the classifier using the augmented set of labeled data).

**Cotraining**

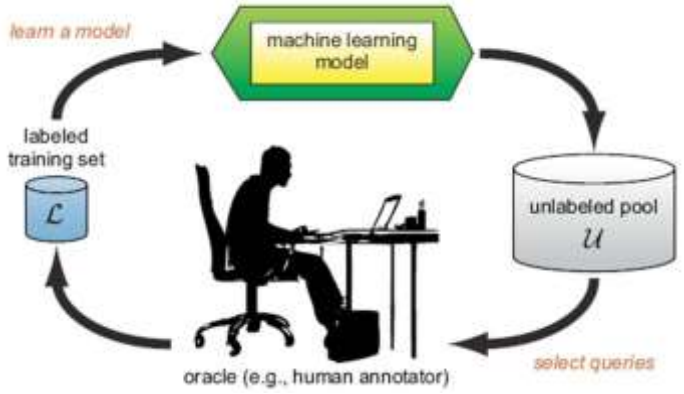
1. Define two separate nonoverlapping feature sets for the labeled data, $X_l$.
2. Train two classifiers, $f_1$ and $f_2$, on the labeled data, where $f_1$ is trained using one of the feature sets and $f_2$ is trained using the other.
3. Classify $X_u$ with $f_1$ and $f_2$ separately.
4. Add the most confident $(x, f_1(x))$ to the set of labeled data used by $f_2$, where $x \in X_u$. Similarly, add the most confident $(x, f_2(x))$ to the set of labeled data used by $f_1$.
5. Repeat.

**FIGURE 7.16**

Self-training and cotraining methods of semisupervised classification.

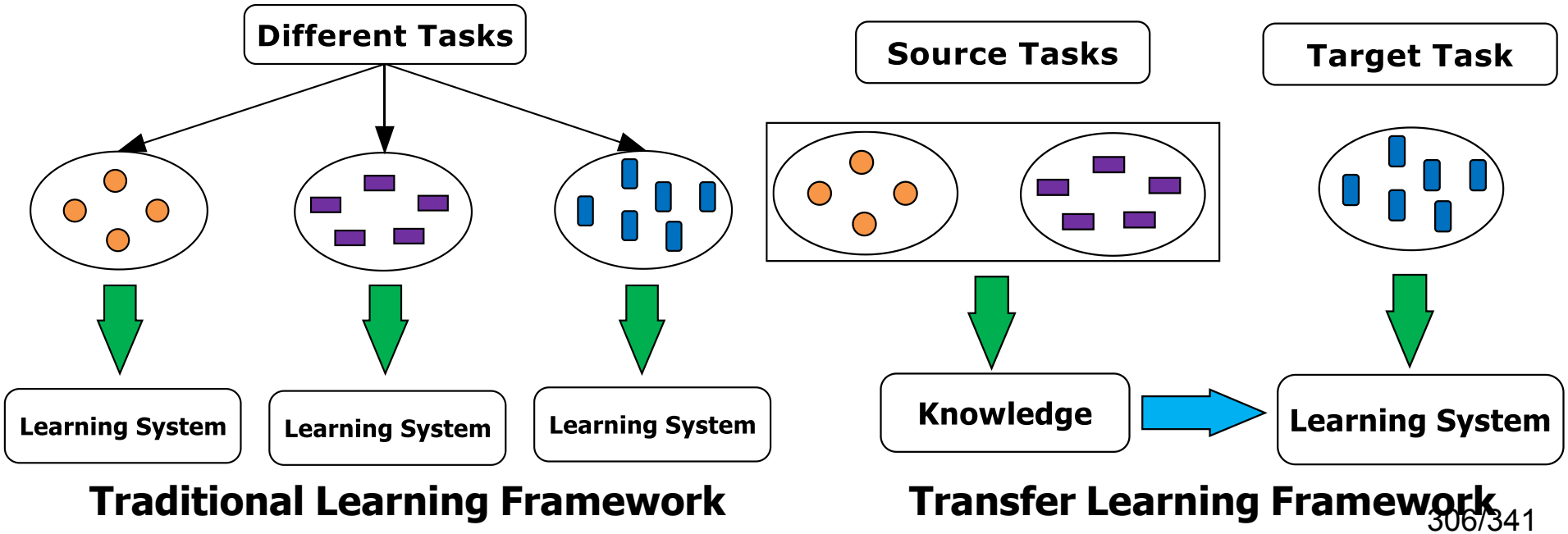
# Active Learning

- Class labels are expensive to obtain
- Active learner: query human (oracle) for labels
- Pool-based approach: Uses a pool of unlabeled data
  - L: a small subset of D is labeled, U: a pool of unlabeled data in D
  - Use a query function to carefully select one or more tuples from U and request labels from an oracle (a human annotator)
  - The newly labeled samples are added to L, and learn a model
  - Goal: Achieve high accuracy using as few labeled data as possible
- Evaluated using *learning curves*: Accuracy as a function of the number of instances queried (# of tuples to be queried should be small)
- Research issue: How to choose the data tuples to be queried?
  - Uncertainty sampling: choose the least certain ones
  - Reduce *version space*, the subset of hypotheses consistent w. the training data
  - Reduce expected entropy over U: Find the greatest reduction in the total number of incorrect predictions



# Transfer Learning: Conceptual Framework

- Transfer learning: Extract knowledge from one or more source tasks and apply the knowledge to a target task
  - Classification of camera reviews to classification of TV reviews
- Traditional learning: Build a new classifier for each new task
- Transfer learning: Build new classifier by applying existing knowledge learned from source tasks



# Transfer Learning: Methods and Applications

- Applications: Especially useful when data is outdated or distribution changes, e.g., Web document classification, e-mail spam filtering
- *Instance-based transfer learning*: Reweight some of the data from source tasks and use it to learn the target task
- TrAdaBoost (Transfer AdaBoost)
  - Assume source and target data each described by the same set of attributes (features) & class labels, but rather diff. distributions
  - Require only labeling a small amount of target data
  - Use source data in training: When a source tuple is misclassified, reduce the weight of such tupels so that they will have less effect on the subsequent classifier
- Research issues
  - Negative transfer: When it performs worse than no transfer at all
  - Heterogeneous transfer learning: Transfer knowledge from different feature space or multiple source domains
  - Large-scale transfer learning

# Distant Supervision

- Automatic generation of large number of labels.
- Exploits external knowledge base to automatically generate labels.
- Examples:
  - Classification of tweets
    - :)  sign means negative  :(  sign means positive
    - If a tweet contains a URL, one can consider URL category as a label
- Issues: it is noisy

# Zero-shot learning

- Example:
  - Suppose that we have a collection of animal images, each of which has a unique label, including “owl,” “dog,” or “fish.”
  - Using this training data set, we can build a classifier, say SVMs or logistic regression classifier.
  - We can use the trained classifier to predict its class label, that is, which one of the three possible animals (owl, dog, or fish).
- *But, what if the test image is actually about a cat?*
- In other words, the class label of the test data *never* appears in the training data.
- *zero-shot learning*
  - classifier needs to predict a test tuple whose class label was never observed during the training stage.
- Solution:
  - we might have some high-level description about the novel classes.
    - For example, for “cat,” we can learn from the Wikipedia that a cat has retractable claws and super night vision.
  - Zero-shot learning tries to leverage such external knowledge or side-information to build a classifier that can recognize such novel class labels

# Zero-shot learning: example

- n training images each of which is represented by a d-D feature vector and a 3-D label vector.
  - The label vector indicates which of the three known classes the training image belongs to. For example, for an image about a "dog," its label vector is [1,0,0].
- In addition, we have the external knowledge about the class label, where each class label (animal) can be described by four semantic attributes. including whether the animal "has four legs," "has wings," "has retractable claws," and "has super night vision."
  - For example, since a dog has four legs, but no wings or retractable claws or super night vision, the class label "dog" can be described by a 4-D semantic attribute vector [1,0,0,0].
- Likewise, the class label "cat" can be described by a 4-D semantic attribute vector [1,0,1,1], meaning that a cat has four legs, retractable claws and super night vision but no wings.
- Such external knowledge is available for both known class labels (e.g., "owl," "dog," and "fish") and novel class labels (e.g., "cat" and "rooster")
- We train a Semantic Attribute Classifier F, which predicts a 4-D semantic attribute vector for an input image represented by a d-dimensional feature vector.
- We can employ a two-level neural network
- Applications
  - Neural activity recognition

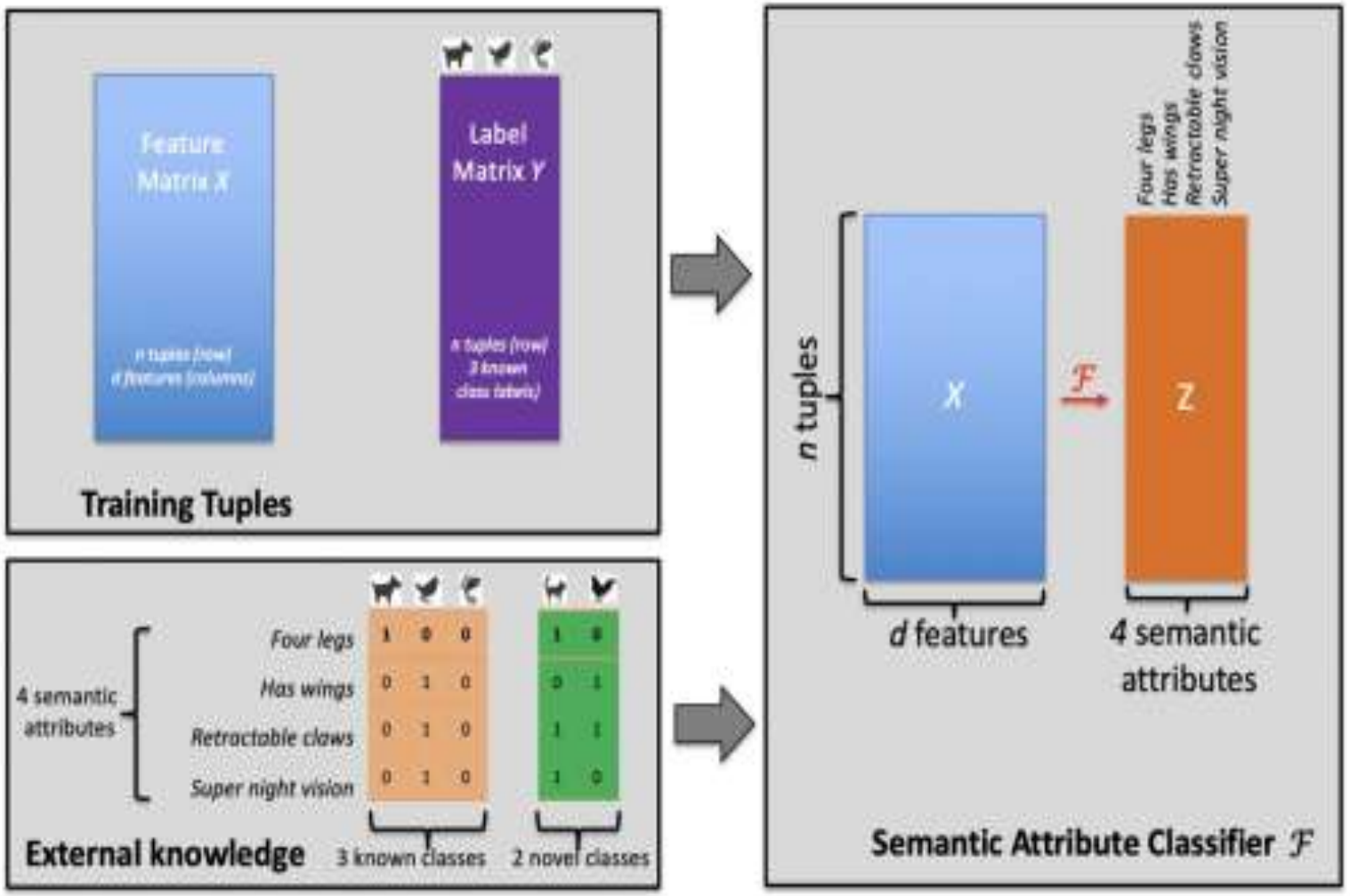



**FIGURE 7.19**

Top left: input $n$ training tuples in $d$-dimensional feature space, each of which is labeled by one of the three known classes (i.e., "dog," "owl," and "fish"). Bottom left: external knowledge where each known and novel class is described by four semantic attributes. Right: the trained semantic attribute classifier $\mathcal{F}$.

# Outline

- Feature Selection and Engineering
- Bayesian Belief Networks
- Support Vector Machines
- Rule based and pattern based classification
- Classification by weak supervision
- **Classification with rich data type**
- Other Related  Methods
- Summary

# Classification with rich data type

- Assumption so far
  - Each training tuple is represented by a feature vector and a class label. We will predict the class label
  - Each training tuple can be considered as a data point in the feature space.
- Other types of data
  - Stream data, sequence data, graph data, grid (image) data, spatial-temporal data (video)
  - Deep learning techniques are powerful tools to classify rich data types.

# Stream data classification

- Transactions arrive sequentially
  - Example bank transactions
  - A sequence of transactions may be fraud.
- Classifier has to classify the transactions and update the labels confirmed by bank expert.
- Challenges
  - Speed of data arrival
  - Impossible to store all data tuples/transactions
  - One-pass constraint
- Ensemble is effective method
- Applications
  - Marketing, network monitoring and sensor networks

**Ensemble method**

- Partition the data into equal sized chunks (i=0,1,…,k) where i=0 is the current i=k is the oldest chunk
- Each chunk i has a training set and labels for I
- We train each classifier (example: Naïve Bayes) which gives posterior probability of a tuple belongs to a particular class.
- The output is the weighted sum of all classifiers.
- When a new chunk comes, we train a new classifier.
- Use newly arrived chunk as a test set and evaluate the performance of k+1 classifiers.
  - We select k classifiers with lowest classification error.
  - Discard the classifier with highest error rate.

# Sequence Classification

- A sequence is an ordered list of values.
  - Example: sentence, DNA segment (value is one of the amino acid: A,C,G,T.
- Goal of sequence mining
  - Predict a label of a sequence
    - Sentiment of a sentence, gene coding are/noncoding area, high value/ordinary customer
- Approach:
  - Convert the input feature into a vector of features and fed into a conventional classifier.
- K-nearest neighbour classifier with commonly used distance measures gives competitive performance
- Selection of features is important
  - Recurrent neural networks is powerful technique
- Number of features may be huge
  - Feature engineering plays a major role.

# Sequence Classification

| | Unigrams | | | | Bigrams | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| Candidate features | A | C | G | T | AC | AG | AT | CG | CT | CA | GA | GC | GT | TA | TC | TG | AA | CC | GG | TT |
| Feature vector (binary) | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Feature vector (frequency) | 1 | 5 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 4 | 0 | 0 |

**FIGURE 7.21**

An example of using *n*-gram for sequence data feature engineering. Given a DNA segment "ACCCCCGT," we want to convert it to a feature vector, using *n*-grams. There are 20 candidate features in total, including 4 unigrams and 16 bigrams. For the binary feature vector (the third row), an element indicates whether the corresponding feature appears (1) or not (0) in the input sequence. For the weighted feature vector (the fourth row), an element indicates how many times (the frequency) that the corresponding feature appears in the input sequence.

# Graph data classification

- Graph is a ubiquitous data type
- Graph classification: predict the label of nodes (node-level classification) or entire graph (graph level classification).
- Node-level classification
  - Given a web graph (with pages and hyperlinks), predict a category of a page
- Graph-level classification
  - Given a collection f molecular graphs, predict whether a given molecule is toxic.
- Two approaches: feature engineering or proximity measures
- Feature Engineering
  - Extract a set of features for each node or a graph and feed into conventional classifier.
  - Nodel–level: number of neighbouring nodes linked to the given node, node importance measure, node clustering coefficient
  - Graph-level: size of the graph (nodes, edges, weights), diameter of the graph, tital number of triangles in the graph
    - Recent technique: Graph neural networks

# Outline

- Feature Selection and Engineering
- Bayesian Belief Networks
- Support Vector Machines
- Rule based and pattern based classification
- Classification by weak supervision
- Classification with rich data type
- **Other Related  Methods**
- Summary

# Other methods

- Multiclass classification
  - How to build a classifier for multiple classes?
- Distance metric learning
  - How to learn the distance automatically?
- How to render interpretability?
- Genetic algorithm
- Reinforcement learning
  - Learning receives evaluation feedback

# 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

# Error-Correcting Codes for Multiclass Classification

- Originally designed to correct errors during data transmission for communication tasks by exploring data redundancy
- Example
  - A 7-bit codeword associated with classes 1-4

| Class | Error-Corr. Codeword | | | | | | |
|---|---|---|---|---|---|---|---|
| $C_1$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $C_2$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| $C_3$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| $C_4$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

  - Given a unknown tuple **X**, the 7-trained classifiers output: 0001010
  - Hamming distance: # of different bits between two codewords
  - H(**X**, $C_1$) = 5, by checking # of bits between [1111111] & [0001010]
  - H(**X**, $C_2$) = 3, H(**X**, $C_3$) = 3, H(**X**, $C_4$) = 1, thus $C_4$ as the label for **X**
- Error-correcting codes can correct up to (h-1)/h 1-bit error, where h is the minimum Hamming distance between any two codewords
- If we use 1-bit per class, it is equiv. to one-vs.-all approach, the code are insufficient to self-correct
- When selecting error-correcting codes, there should be good row-wise and col.-wise separation between the codewords

# Interpretability

- Interpretability: The models' ability to explain the classification results or process in a user understandable way.
- Decision trees are interpretable
- Linear classifiers are interpretable due to the weight of the attribute.
- Approach: LIME: Local Interpretable Model-agnostic Explanation

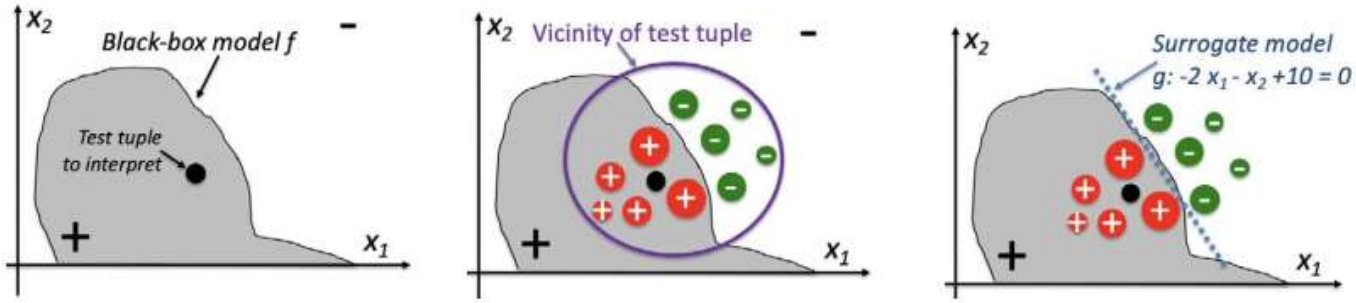



**FIGURE 7.28**

An example of LIME: Local Interpretable Model-agnostic Explanation.

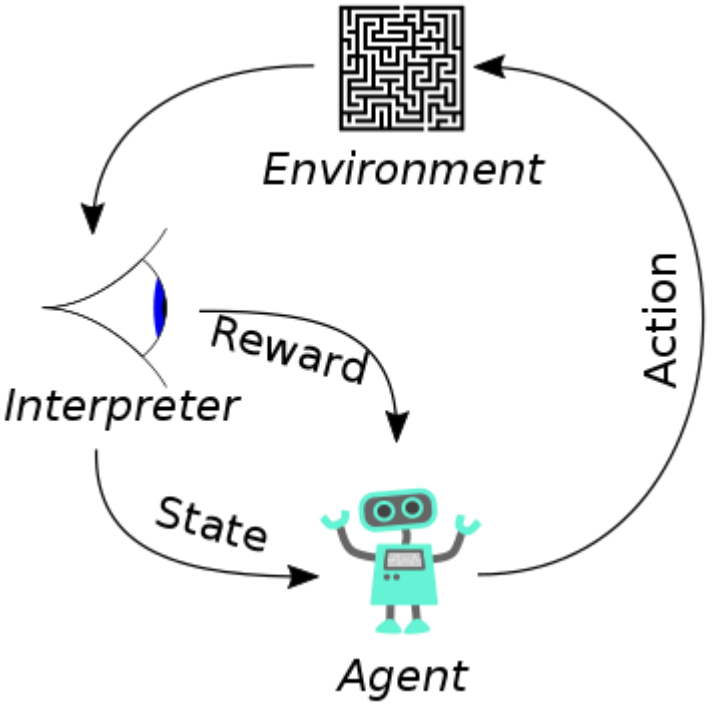
- Beyond classification, interpretability also plays an important role in other data mining tasks, including clustering, outlier detection, ranking, and recommendation. In addition to making the data mining models transparent so as to gain the users' trust of the model, interpretability is also intimately related to other important aspects of data mining, such as fairness, robustness, causality, privacy.

# Genetic Algorithms (GA)

- Genetic Algorithm: based on an analogy to biological evolution
- An initial **population** is created consisting of randomly generated rules
  - Each rule is represented by a string of bits
  - E.g., if $A_1$ and $\neg A_2$ then $C_2$ can be encoded as 100
  - If an attribute has k > 2 values, k bits can be used
- Based on the notion of survival of the **fittest**, a new population is formed to consist of the fittest rules and their offspring
- The ***fitness of a rule*** is represented by its classification accuracy on a set of training examples
- Offspring are generated by ***crossover*** and ***mutation***
- The process continues until a population P evolves *when each rule in P satisfies a prespecified threshold*
- Slow but easily parallelizable

# Reinforcement learning

- In reinforcement learning, the learning agent tries to figure out what to do (e.g., choose the best product to advertise in order to maximize the overall increased revenue) by interacting with the environment (e.g., trying a few different advertisements, observing their rewards, and adjusting the advertisement for next day accordingly).
- In classification, classifier receives the *instructive feedback* (i.e., the true labels for the training tuples) in order to construct the best classifier.
- In reinforcement learning, the learning agent receives *evaluative feedback*

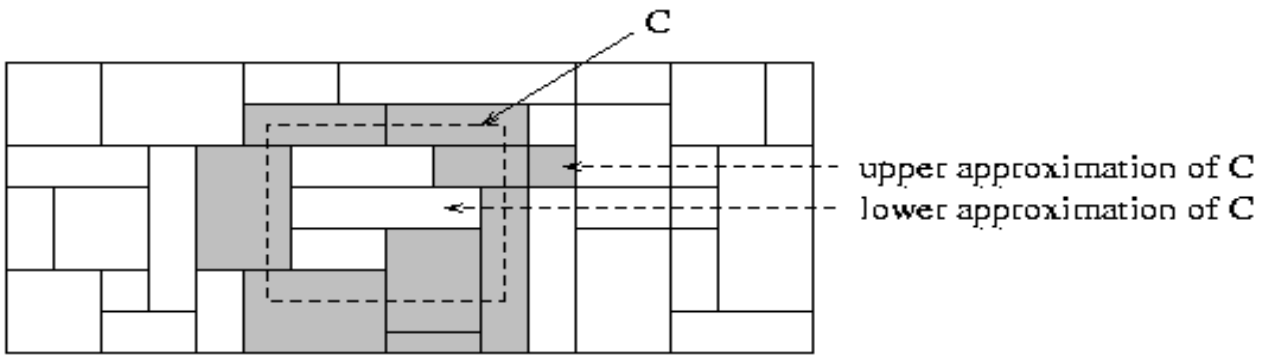


# Outline

- Feature Selection and Engineering
- Bayesian Belief Networks
- Support Vector Machines
- Rule based and pattern based classification
- Classification by weak supervision
- Classification with rich data type
- Other Related  Methods
- Summary

# Rough Set Approach

- Rough sets are used to **approximately or "roughly" define equivalent classes**
- A rough set for a given class C is approximated by two sets: a lower approximation (certain to be in C) and an upper approximation (cannot be described as not belonging to C)
- Finding the minimal subsets (**reducts**) of attributes for feature reduction is NP-hard but a **discernibility matrix** (which stores the differences between attribute values for each pair of data tuples) is used to reduce the computation intensity



# Fuzzy sets

- The word "fuzzy" means "vaguness (ambiguity)".
- Fuzziness occurs when the boundary of a piece of information is not clear-cut.
- Fuzzy sets - 1965 Lotfi Zadeh as an extension of classical notation set.
- Classical set theory allows the membership of the elements in the set in **binary terms**.
- Fuzzy set theory permits membership function valued in the interval [0,1].

# Fuzzy sets

**Example:**

Words like young, tall, good or high are fuzzy.

- There is no single quantitative value which defines the term young.
- For some people, age 25 is young, and for others, age 35 is young.
  The concept young has no clean boundary.
- Age 35 has some possibility of being young and usually
- depends on the context in which it is being considered.

Fuzzy set theory is an extension of classical set theory where elements have degree of membership.

# Introduction

- In real world, there exist much fuzzy knowledge (i.e. vague, uncertain inexact etc).
- Human **thinking** and **reasoning** (analysis, logic, interpretation) frequently involved **fuzzy** information.
- Human can give satisfactory answers, which are probably true.
- Our systems are unable to answer many question because the systems are designed based upon classical set theory (Unreliable and incomplete).

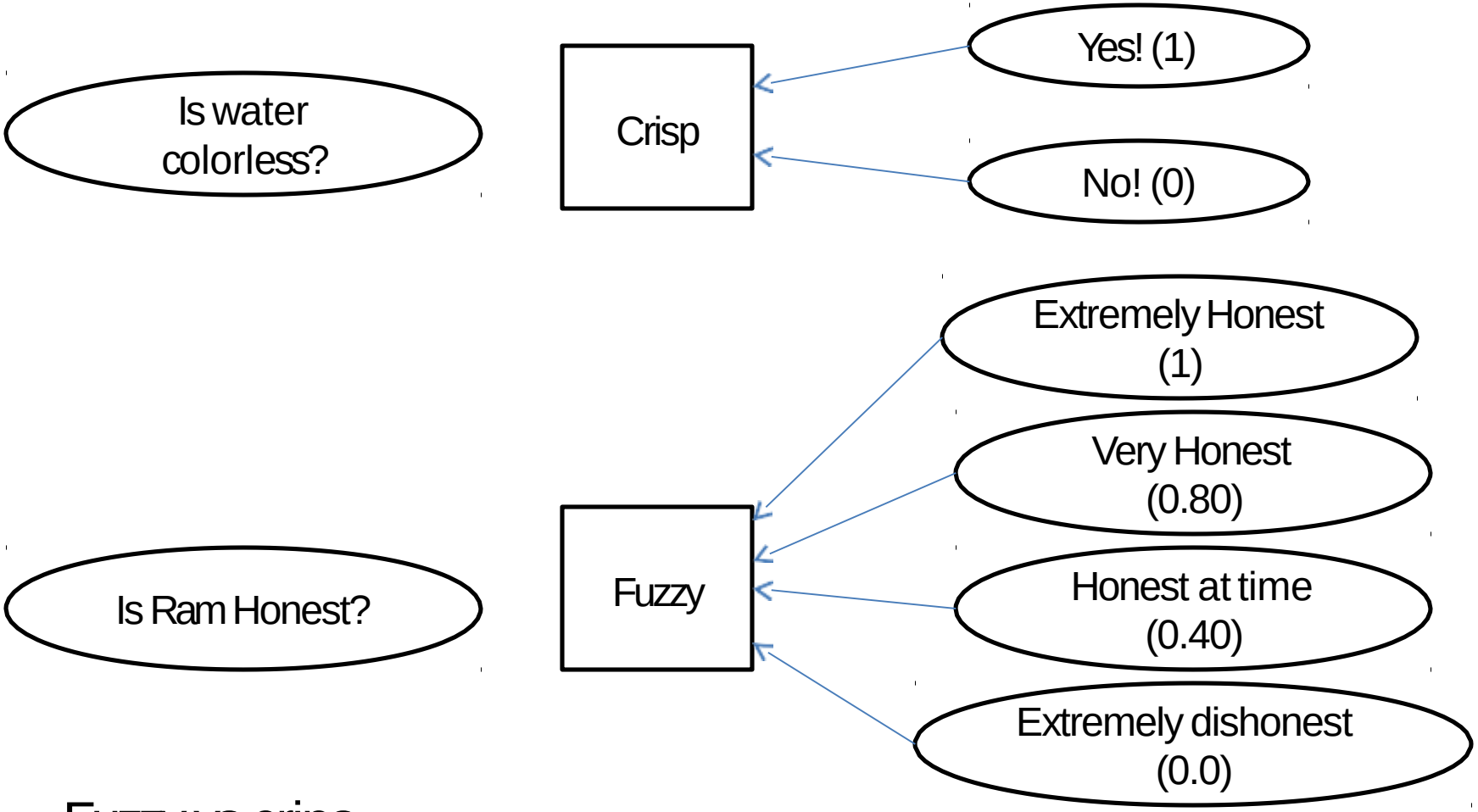
- We want, our system should be able to cope with unreliable and incomplete information.
- Fuzzy system have been provide solution.

# Introduction

| Classical set theory | Fuzzy set theory |
| --- | --- |
| • Classes of objects with sharp boundaries. | • Classes of objects with un- sharp boundaries. |
| • A classical set is defined by crisp(exact) boundaries, i.e., there is no uncertainty about the location of the set boundaries. | • A fuzzy set is defined by its ambiguous boundaries, i.e., there exists uncertainty about the location of the set boundaries. |
| • Widely used in digital system design | • Used in fuzzy controllers. |

# Introduction (Continue)

Example




Fuzzy vs crips

# Classical set theory

- A Set is any well defined collection of objects.
- An object in a set is called an element or member of that set.
  - Sets are defined by a simple statement,
  - Describing whether a particular element having a certain
  - propertybelongs to that particular set.

$$A = \{a_1, a_2, a_3, \ldots\ldots, a_n\}$$

- If the elements $a_i$ $(i = 1,2,3,\ldots,n)$ of a set A are subset of universal set X, then set A can be represented for all elements $x \epsilon X$ by its characteristics function

$$\mu_A(x) = 1 \text{ if } x \epsilon X \text{ otherwise } 0$$

# Operations on classical set theory

***Union:*** the union of two sets A and B is given as

$$A \cup B = \{ x \mid x \in A \text{ or } x \in B \}$$

***Intersection:*** the intersection of two sets A and B is given as

$$A \cap B = \{ x \mid x \in A \text{ and } x \in B \}$$

***Complement:*** It is denoted by $\tilde{A}$ and is defined as

$$\tilde{A} = \{ x \mid x \text{ does not belongs } A \text{ and } x \in X \}$$

# Fuzzy Sets

- Fuzzy sets theory is an extension of classical set theory.
- Elements have varying degree of membership. A logic based on two truth values,
- ***True*** and ***False*** is sometimes insufficient when describing human reasoning.
- Fuzzy Logic uses the whole interval between 0 (false) and 1 (true) to describe human reasoning.
- A Fuzzy Set is any set that allows its members to have different degree of membership, called **membership function**, having interval [0,1].

# Fuzzy Sets

- **Fuzzy Logic** is derived from fuzzy set theory
- Many degree of membership (between 0 to 1) are allowed.
- Thus a membership function $\mu_A(x)$ is associated with a fuzzy sets Ã such that the function maps every element of universe of discourse X to the interval [0,1].
- The mapping is written as: $\mu_{\tilde{A}}(x)$: X □ [0,1].

- Fuzzy Logic is capable of handing inherently imprecise (vague or inexact or rough or inaccurate) concepts

# Fuzzy Sets

- **Fuzzy set** is defined as follows:

- If X is an universe of discourse and x is a particular element of X, then a fuzzy set A defined on X and can be written as a collection of ordered pairs

$$A = \{(x, \mu_{\tilde{A}}(x)), x \epsilon X\}$$

# Fuzzy Sets (Continue)

Example

- Let $X = \{g_1, g_2, g_3, g_4, g_5\}$ be the reference set of students.
- Let $\tilde{A}$ be the fuzzy set of "smart" students, where "smart" is fuzzy term.

$$\tilde{A} = \{(g_1,0.4)(g_2,0.5)(g_3,1)(g_4,0.9)(g_5,0.8)\}$$

Here $\tilde{A}$ indicates that the smartness of $g_1$ is 0.4 and soon

# Fuzzy Sets (Continue)

**Membership Function**

- The membership function fully defines the fuzzyset
- A membership function provides a measure of *the degree of similarity* of an element to a fuzzyset

**Membership functions can**

- either be chosen by the user arbitrarily, based on the user's experience (MF chosen by two users could be different depending upon their experiences, perspectives, etc.)
- Or be designed using machine learning methods (e.g., artificial neural networks, genetic algorithms, etc.)

# Fuzzy Sets (Continue)

There are different shapes of membership functions;

- **Triangular,**
- **Trapezoidal,**
- **Gaussian, etc**

# Fuzzy Set Classification

- Fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership (such as in a *fuzzy membership graph*)
- Attribute values are converted to fuzzy values. Ex.:
  - Income, *x*, is assigned a fuzzy membership value to each of the discrete categories {low, medium, high}, e.g. $49K belongs to "medium income" with fuzzy value 0.15 but belongs to "high income" with fuzzy value 0.96
  - Fuzzy membership values do not have to sum to 1.
- Each applicable rule contributes a vote for membership in the categories
- Typically, the truth values for each predicted category are summed, and these sums are combined