# Data Mining
# 資料探勘

## 關連分析
## (Association Analysis)

1012DM02
MI4
Thu 9, 10 (16:10-18:00) B216


Min-Yuh Day
戴敏育
Assistant Professor
專任助理教授
Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系

http://mail. tku.edu.tw/myday/


2013-03-07

# 課程大綱 (Syllabus)

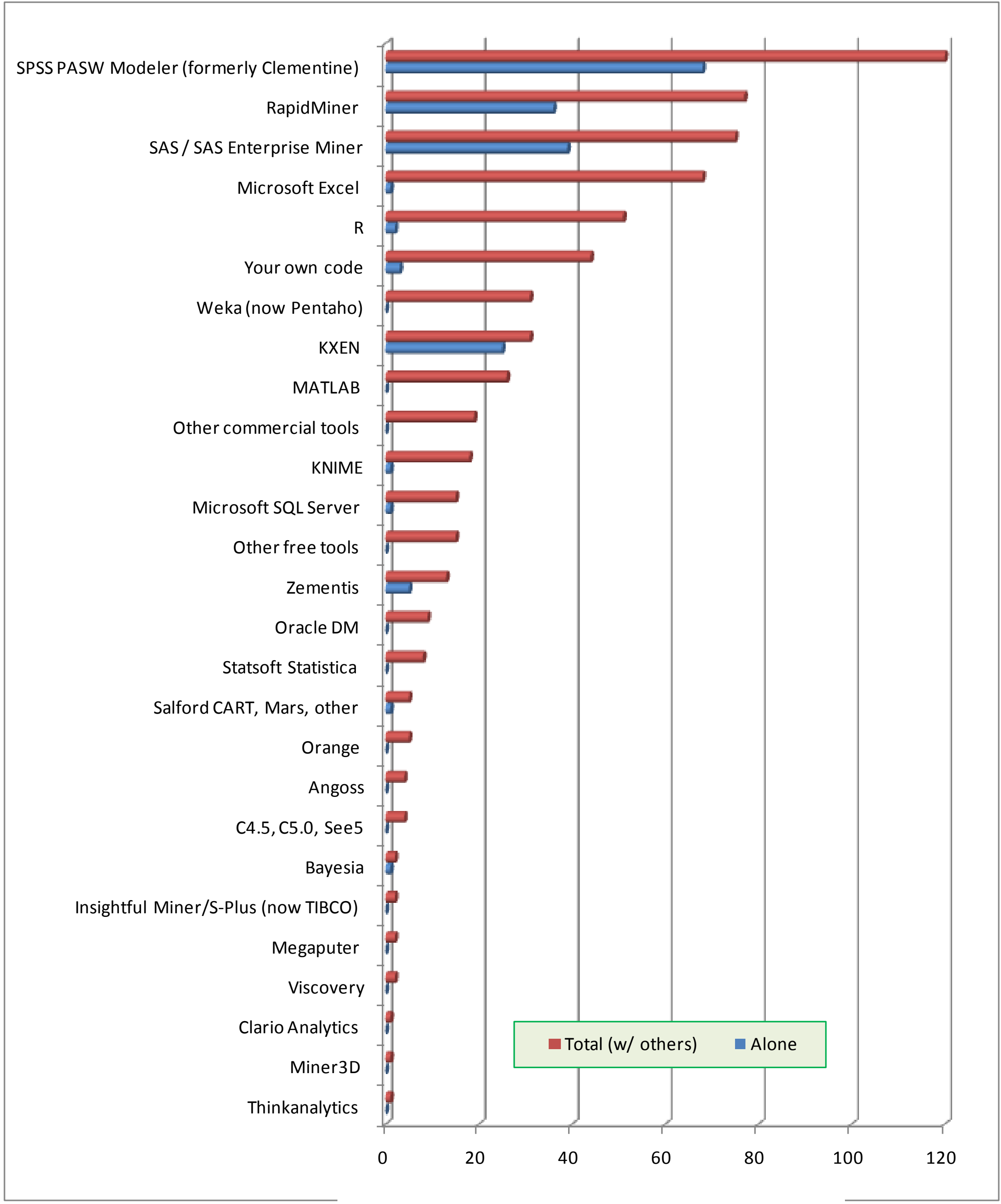
| 週次 | 日期 | 內容 (Subject/Topics) |
| --- | --- | --- |
| 1 | 102/02/21 | 資料探勘導論 (Introduction to Data Mining) |
| 2 | 102/02/28 | 和平紀念日 (放假一天)<br>(Peace Memorial Day) (No Classes) |
| 3 | 102/03/07 | 關連分析 (Association Analysis) |
| 4 | 102/03/14 | 分類與預測 (Classification and Prediction) |
| 5 | 102/03/21 | 分群分析 (Cluster Analysis) |
| 6 | 102/03/28 | SAS企業資料採礦實務<br>(Data Mining Using SAS Enterprise Miner) |
| 7 | 102/04/04 | 清明節、兒童節(放假一天)<br>(Children's Day, Tomb Sweeping Day)(No Classes) |
| 8 | 102/04/11 | 個案分析與實作一 (SAS EM 分群分析)：<br>Banking Segmentation (Cluster Analysis – K-Means using SAS EM) |

# 課程大綱 (Syllabus)

| 週次 | 日期 | 內容 (Subject/Topics) |
|---|---|---|
| 9 | 102/04/18 | 期中報告 (Midterm Presentation) |
| 10 | 102/04/25 | 期中考試週 |
| 11 | 102/05/02 | 個案分析與實作二 (SAS EM 關連分析)：<br>Web Site Usage Associations ( Association Analysis using SAS EM) |
| 12 | 102/05/09 | 個案分析與實作三 (SAS EM 決策樹、模型評估)：<br>Enrollment Management Case Study<br>(Decision Tree, Model Evaluation using SAS EM) |
| 13 | 102/05/16 | 個案分析與實作四 (SAS EM 迴歸分析、類神經網路)：<br>Credit Risk Case Study<br>(Regression Analysis, Artificial Neural Network using SAS EM) |
| 14 | 102/05/23 | 期末專題報告 (Term Project Presentation) |
| 15 | 102/05/30 | 畢業考試週 |

# Data Mining Software

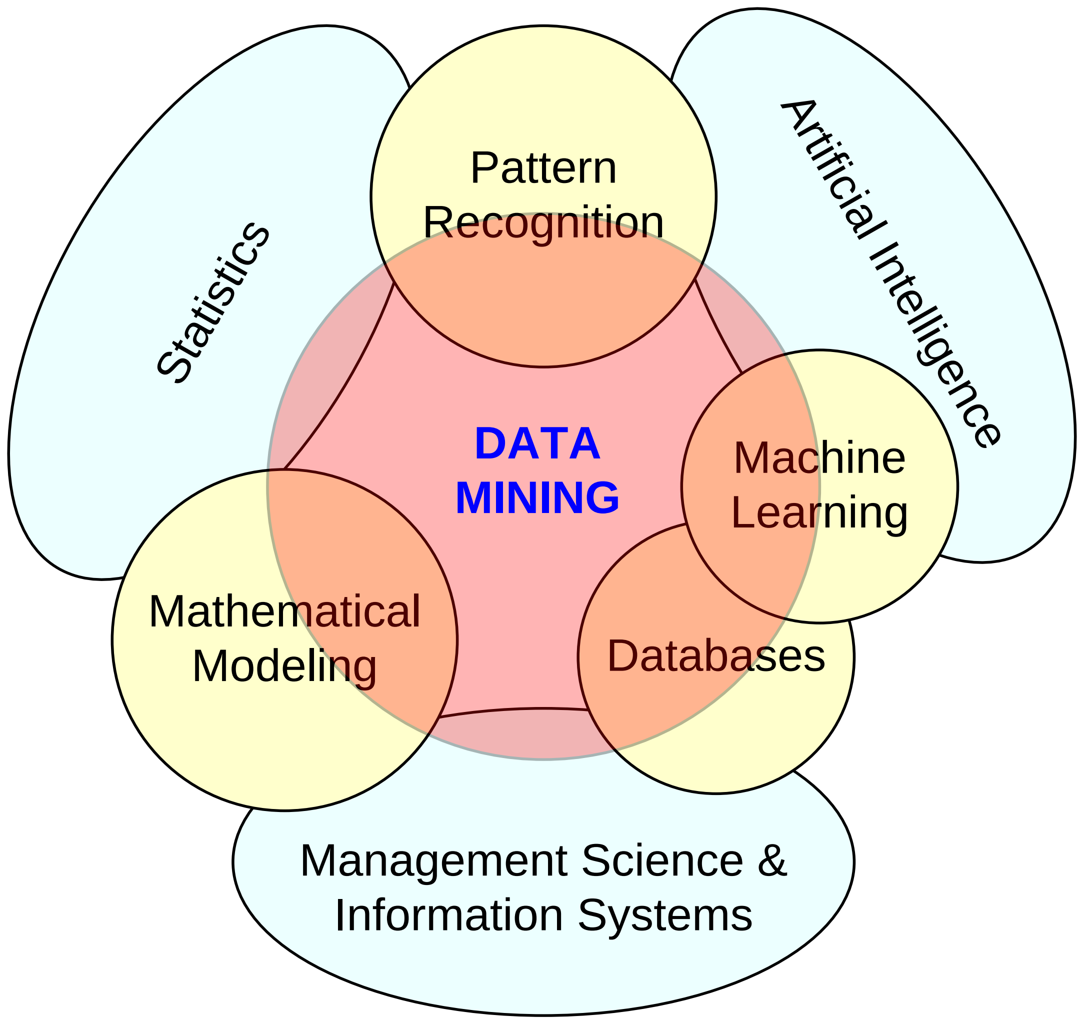
- Commercial
  - SPSS - PASW (formerly Clementine)
  - SAS - Enterprise Miner
  - IBM - Intelligent Miner
  - StatSoft – Statistical Data Miner
  - … many more
- Free and/or Open Source
  - Weka
  - RapidMiner…




*Source: KDNuggets.com, May 2009*

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# Data Mining at the Intersection of Many Disciplines




Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# A Taxonomy for Data Mining Tasks

| Data Mining | Learning Method | Popular Algorithms |
|---|---|---|
| Prediction | Supervised | Classification and Regression Trees, ANN, SVM, Genetic Algorithms |
| — Classification | Supervised | Decision trees, ANN/MLP, SVM, Rough sets, Genetic Algorithms |
| — Regression | Supervised | Linear/Nonlinear Regression, Regression trees, ANN/MLP, SVM |
| Association | Unsupervised | Apriory, OneR, ZeroR, Eclat |
| — Link analysis | Unsupervised | Expectation Maximization, Apriory Algorithm, Graph-based Matching |
| — Sequence analysis | Unsupervised | Apriory Algorithm, FP-Growth technique |
| Clustering | Unsupervised | K-means, ANN/SOM |
| — Outlier analysis | Unsupervised | K-means, Expectation Maximization (EM) |

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# Why Data Mining?

- More intense competition at the global scale
- Recognition of the value in data sources
- Availability of quality data on customers, vendors, transactions, Web, etc.
- Consolidation and integration of data repositories into data warehouses
- The exponential increase in data processing and storage capabilities; and decrease in cost
- Movement toward conversion of information resources into nonphysical form

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# Definition of Data Mining

- The nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data stored in structured databases.
  *- Fayyad et al., (1996)*
- Keywords in this definition: Process, nontrivial, valid, novel, potentially useful, understandable.
- Data mining: a misnomer?
- Other names:
  – knowledge extraction, pattern analysis, knowledge discovery, information harvesting, pattern searching, data dredging,…

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

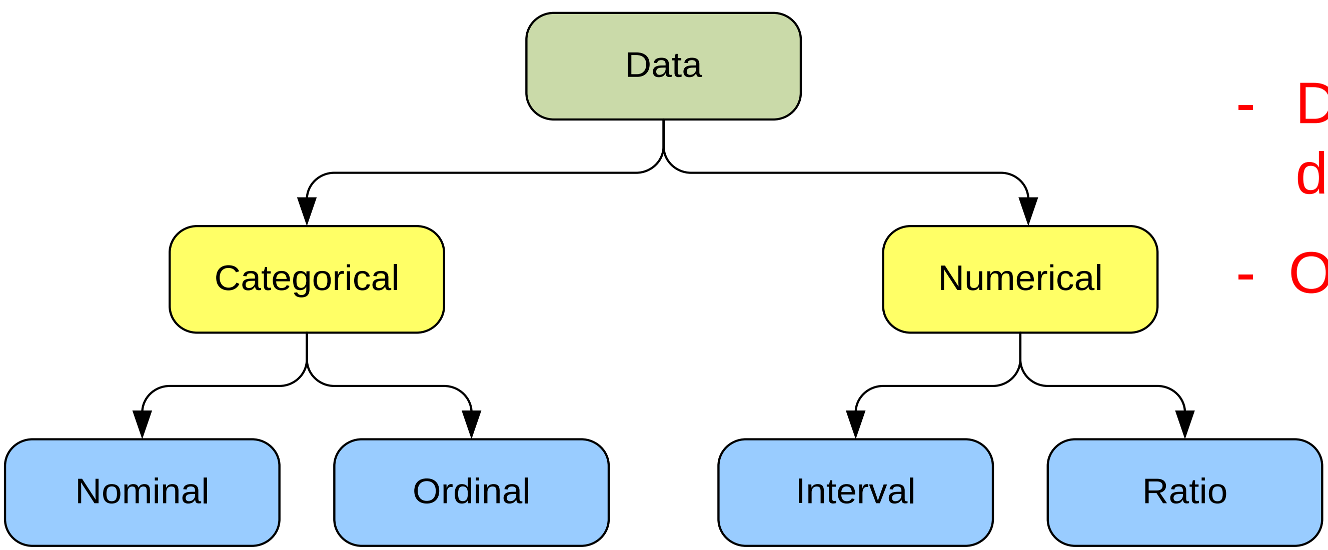
# Data Mining Characteristics/Objectives

- Source of data for DM is often a consolidated data warehouse (not always!)
- DM environment is usually a client-server or a Web-based information systems architecture
- Data is the most critical ingredient for DM which may include soft/unstructured data
- The miner is often an end user
- Striking it rich requires creative thinking
- Data mining tools' capabilities and ease of use are essential (Web, Parallel processing, etc.)

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# Data in Data Mining

- Data: a collection of facts usually obtained as the result of experiences, observations, or experiments
- Data may consist of numbers, words, images, …
- Data: lowest level of abstraction (from which information and knowledge are derived)




- DM with different data types?
- Other data types?

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# What Does DM Do?

- DM extract patterns from data
  - Pattern?
    A mathematical (numeric and/or symbolic) relationship among data items

- Types of patterns
  - Association
  - Prediction
  - Cluster (segmentation)
  - Sequential (or time series) relationships

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# Data Mining Applications

- Customer Relationship Management
  - Maximize return on marketing campaigns
  - Improve customer retention (churn analysis)
  - Maximize customer value (cross-, up-selling)
  - Identify and treat most valued customers
- Banking and Other Financial
  - Automate the loan application process
  - Detecting fraudulent transactions
  - Optimizing cash reserves with forecasting

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# Data Mining Applications (cont.)

- Retailing and Logistics
  - Optimize inventory levels at different locations
  - Improve the store layout and sales promotions
  - Optimize logistics by predicting seasonal effects
  - Minimize losses due to limited shelf life
- Manufacturing and Maintenance
  - Predict/prevent machinery failures
  - Identify anomalies in production systems to optimize the use manufacturing capacity
  - Discover novel patterns to improve product quality

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# Data Mining Applications (cont.)

- Brokerage and Securities Trading
  - Predict changes on certain bond prices
  - Forecast the direction of stock fluctuations
  - Assess the effect of events on market movements
  - Identify and prevent fraudulent activities in trading
- Insurance
  - Forecast claim costs for better business planning
  - Determine optimal rate plans
  - Optimize marketing to specific customers
  - Identify and prevent fraudulent claim activities

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# Data Mining Applications (cont.)

- Computer hardware and software
- Science and engineering
- Government and defense
- Homeland security and law enforcement
- Travel industry
- Healthcare
- Medicine
- Entertainment industry
- Sports
- Etc.

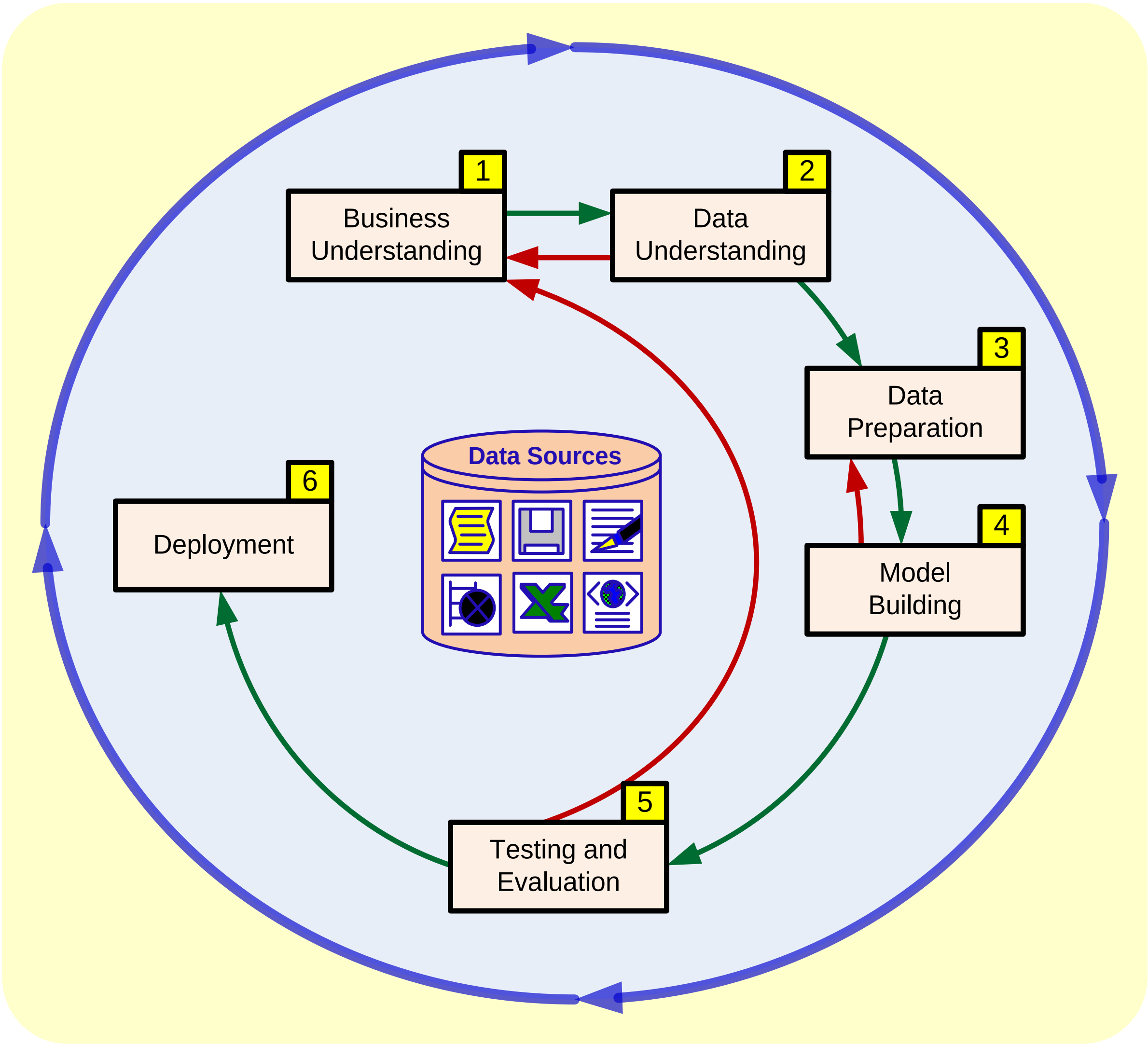
Highly popular application areas for data mining

# Data Mining Process

- A manifestation of best practices
- A systematic way to conduct DM projects
- Different groups has different versions
- Most common standard processes:
  - CRISP-DM
    (Cross-Industry Standard Process for Data Mining)
  - SEMMA
    (Sample, Explore, Modify, Model, and Assess)
  - KDD
    (Knowledge Discovery in Databases)

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# Data Mining Process: CRISP-DM




Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# Data Mining Process: CRISP-DM

Step 1: Business Understanding

Step 2: Data Understanding

Step 3: Data Preparation (!)

Accounts for ~85% of total project time

Step 4: Model Building

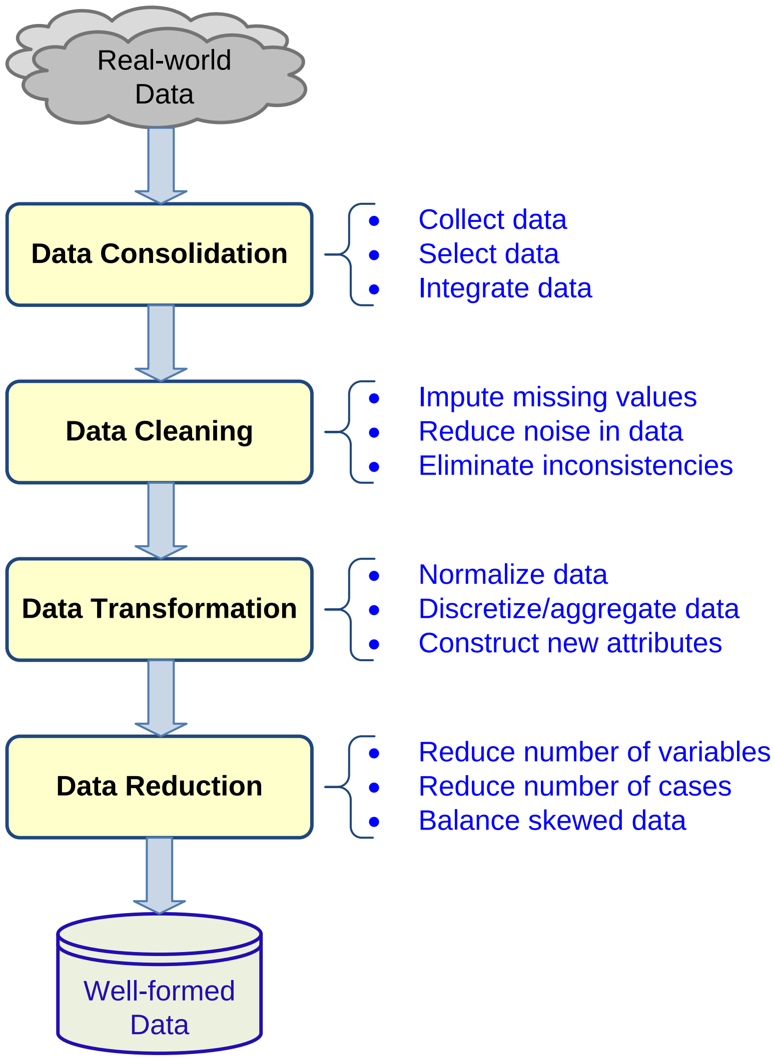
Step 5: Testing and Evaluation

Step 6: Deployment

- The process is highly repetitive and experimental (DM: art versus science?)

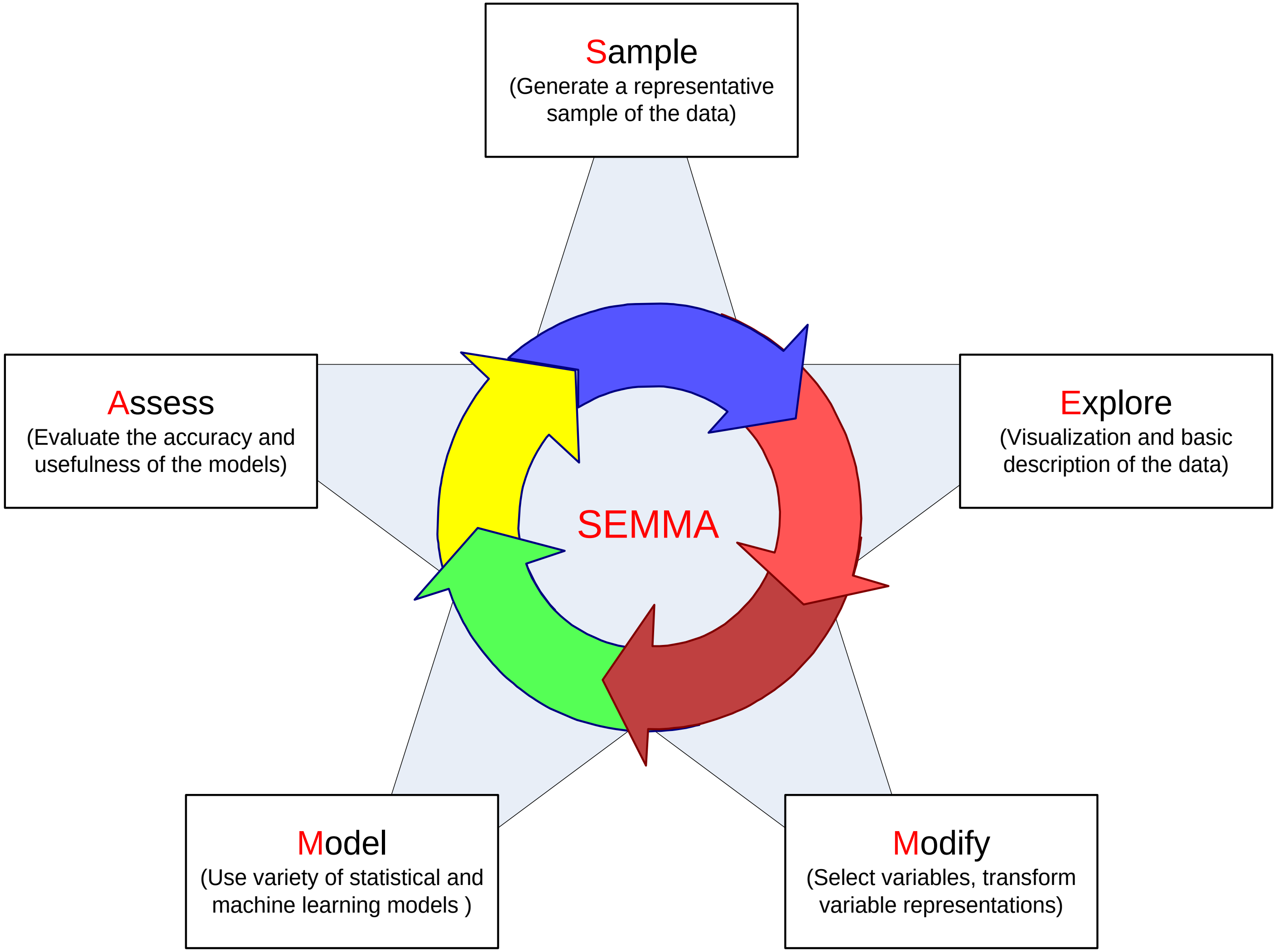
Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# Data Preparation – A Critical DM Task




Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# Data Mining Process: SEMMA



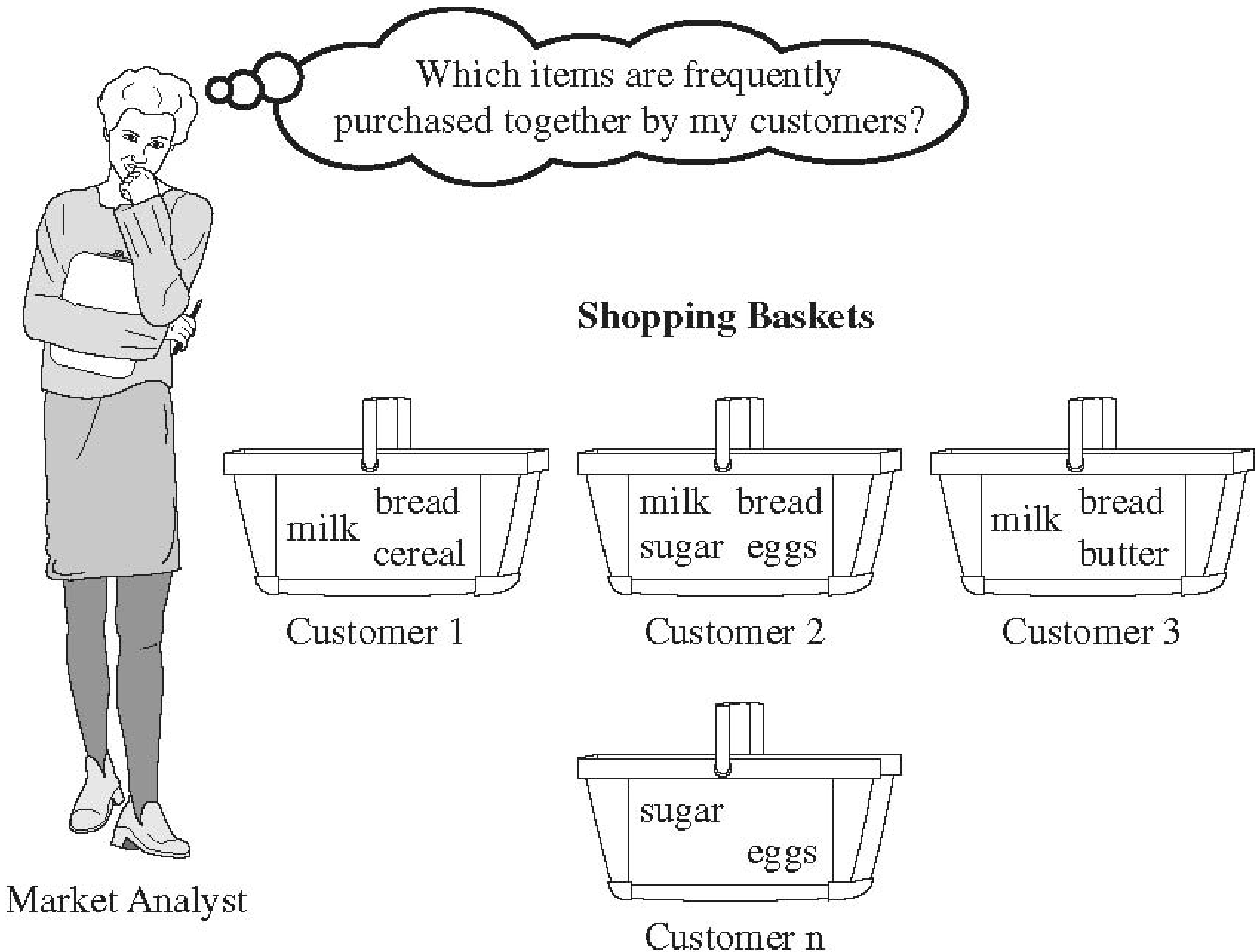

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# Association Analysis: Mining Frequent Patterns, Association and Correlations

- Association Analysis
- Mining Frequent Patterns
- Association and Correlations
- Apriori Algorithm

Source: Han & Kamber (2006)

# Market Basket Analysis




Source: Han & Kamber (2006)

# Association Rule Mining

- Apriori Algorithm

**Raw Transaction Data**

| Transaction No | SKUs (Item No) |
|---|---|
| 1 | 1, 2, 3, 4 |
| 1 | 2, 3, 4 |
| 1 | 2, 3 |
| 1 | 1, 2, 4 |
| 1 | 1, 2, 3, 4 |
| 1 | 2, 4 |

**One-item Itemsets**

| Itemset (SKUs) | Support |
|---|---|
| 1 | 3 |
| 2 | 6 |
| 3 | 4 |
| 4 | 5 |

**Two-item Itemsets**

| Itemset (SKUs) | Support |
|---|---|
| 1, 2 | 3 |
| 1, 3 | 2 |
| 1, 4 | 3 |
| 2, 3 | 4 |
| 2, 4 | 5 |
| 3, 4 | 3 |

**Three-item Itemsets**

| Itemset (SKUs) | Support |
|---|---|
| 1, 2, 4 | 3 |
| 2, 3, 4 | 3 |

Source: Turban et al. (2011), Decision Support and Business Intelligence Systems

# Association Rule Mining

- A very popular DM method in business
- Finds interesting relationships (affinities) between variables (items or events)
- Part of machine learning family
- Employs unsupervised learning
- There is no output variable
- Also known as market basket analysis
- Often used as an example to describe DM to ordinary people, such as the famous “relationship between diapers and beers!”

# Association Rule Mining

- Input: the simple point-of-sale transaction data
- Output: Most frequent affinities among items
- Example: according to the transaction data…

  "Customer who bought a laptop computer and a virus protection software, also bought extended service plan 70 percent of the time."
- How do you use such a pattern/knowledge?
  - Put the items next to each other for ease of finding
  - Promote the items as a package (do not put one on sale if the other(s) are on sale)
  - Place items far apart from each other so that the customer has to walk the aisles to search for it, and by doing so potentially seeing and buying other items

# Association Rule Mining

- A representative applications of association rule mining include
  - In business: cross-marketing, cross-selling, store design, catalog design, e-commerce site design, optimization of online advertising, product pricing, and sales/promotion configuration
  - In medicine: relationships between symptoms and illnesses; diagnosis and patient characteristics and treatments (to be used in medical DSS); and genes and their functions (to be used in genomics projects)…

# Association Rule Mining

- Are all association rules interesting and useful?

A Generic Rule: **X $\Rightarrow$ Y [S%, C%]**

**X, Y**: products and/or services

**X:** Left-hand-side (LHS)

**Y:** Right-hand-side (RHS)

**S:** Support: how often **X** and **Y** go together

**C:** Confidence: how often **Y** go together with the **X**

Example: {Laptop Computer, Antivirus Software} $\Rightarrow$ {Extended Service Plan} [30%, 70%]

# Association Rule Mining

- Algorithms are available for generating association rules
  - Apriori
  - Eclat
  - FP-Growth
  - + Derivatives and hybrids of the three
- The algorithms help identify the frequent item sets, which are, then converted to association rules

# Association Rule Mining

- Apriori Algorithm
  - Finds subsets that are common to at least a minimum number of the itemsets
  - uses a bottom-up approach
    - frequent subsets are extended one item at a time (the size of frequent subsets increases from one-item subsets to two-item subsets, then three-item subsets, and so on), and
    - groups of candidates at each level are tested against the data for minimum

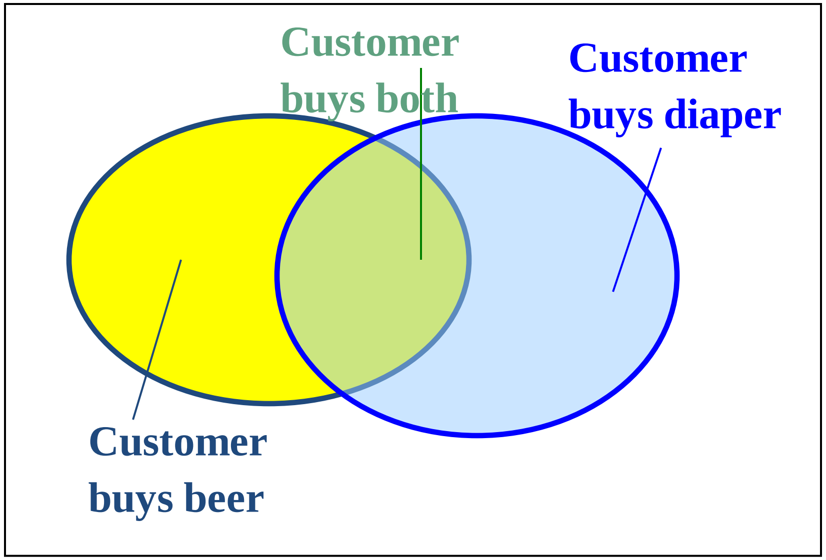
# What Is Frequent Pattern Analysis?

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?
    — Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Source: Han & Kamber (2006)

# Basic Concepts: Frequent Patterns and Association Rules

| Transaction-id | Items bought |
|---|---|
| 10 | A, B, D |
| 20 | A, C, D |
| 30 | A, D, E |
| 40 | B, E, F |
| 50 | B, C, D, E, F |




- Itemset X = $\{x_1, \ldots, x_k\}$
- Find all the rules $X \rightarrow Y$ with minimum support and confidence
  - support, *s*, probability that a transaction contains $X \cup Y$
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

*Let* $sup_{min} = 50\%$, $conf_{min} = 50\%$
*Freq. Pat.:* {*A:3, B:3, D:4, E:3, AD:3*}
Association rules:

$A \rightarrow D$ (60%, 100%)
$D \rightarrow A$ (60%, 75%)

$A \rightarrow D$ (support = 3/5 = 60%, confidence = 3/3 =100%)
$D \rightarrow A$ (support = 3/5 = 60%, confidence = 3/4 = 75%)

Source: Han & Kamber (2006)

# Market basket analysis

- Example
  - Which groups or sets of items are customers likely to purchase on a given trip to the store?
- Association Rule
  - *Computer → antivirus_software [support = 2%; confidence = 60%]*
    - A support of 2% means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together.
    - A confidence of 60% means that 60% of the customers who purchased a computer also bought the software.

Source: Han & Kamber (2006)

# Association rules

- Association rules are considered interesting if they satisfy both
  - a minimum support threshold and
  - a minimum confidence threshold.

Source: Han & Kamber (2006)

# Frequent Itemsets, Closed Itemsets, and Association Rules

Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of items. Let $D$, the task-relevant data, be a set of database transactions where each transaction $T$ is a set of items such that $T \subseteq I$. Each transaction is associated with an identifier, called TID. Let $A$ be a set of items. A transaction $T$ is said to contain $A$ if and only if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$, and $A \cap B = \phi$. The rule $A \Rightarrow B$ holds in the transaction set $D$ with **support** $s$, where $s$ is the percentage of transactions in $D$ that contain $A \cup B$ (i.e., the *union* of sets $A$ and $B$, or say, both $A$ and $B$). This is taken to be the probability, $P(A \cup B)$.[1] The rule $A \Rightarrow B$ has **confidence** $c$ in the transaction set $D$, where $c$ is the percentage of transactions in $D$ containing $A$ that also contain $B$. This is taken to be the conditional probability, $P(B|A)$. That is,

$$Support\ (A \rightarrow B) = P(A \cup B)$$

$$Confidence\ (A \rightarrow B) = P(B|A)$$

Source: Han & Kamber (2006)

# ***Support*** $(A \rightarrow B) = P(A \cup B)$
# ***Confidence*** $(A \rightarrow B) = P(B|A)$

- The notation $P(A \cup B)$ *indicates the probability that a transaction contains the union of set A and set B*
  - *(i.e., it contains every item in A and in B).*
- *This should not be confused with P(A or B),* which indicates the probability that a transaction contains either *A or B.*

Source: Han & Kamber (2006)

- Rules that satisfy both a minimum support threshold (*min_sup*) and a minimum confidence threshold (*min_conf*) are called **strong**.
- By convention, we write support and confidence values so as to occur between 0% and 100%, rather than 0 to 1.0.

- itemset
  - A set of items is referred to as an itemset.
- K-itemset
  - An itemset that contains *k items is a k*-itemset*.*
- Example:
  - The set {*computer, antivirus software*} is a 2-itemset.

Source: Han & Kamber (2006)

# Absolute Support and Relative Support

- Absolute Support
  - The occurrence frequency of an itemset is the number of transactions that contain the itemset
    - frequency, support count, or count of the itemset
  - Ex: 3
- Relative support
  - Ex: 60%

Source: Han & Kamber (2006)

- If the relative support of an itemset *I satisfies a prespecified minimum support threshold, then I is a* frequent itemset.
  - *i.e., the absolute* support of *I satisfies the corresponding minimum support count threshold*
- The set of frequent *k-itemsets is commonly denoted by* $L_K$

Source: Han & Kamber (2006)

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} = \frac{support\_count(A \cup B)}{support\_count(A)}$$

- the confidence of rule *A* → *B can be easily derived from the* support counts of *A and A* ∪ *B.*
- once the support counts of *A, B, and A* ∪ *B are* found, it is straightforward to derive the corresponding association rules *A* →*B and B* →*A* and check whether they are strong.
- Thus the problem of mining association rules can be reduced to that of mining frequent itemsets.

Source: Han & Kamber (2006)

# Association rule mining: Two-step process

1. Find all frequent itemsets
   – By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, *min_sup.*
2. Generate strong association rules from the frequent itemsets
   – By definition, these rules must satisfy minimum support and minimum confidence.

Source: Han & Kamber (2006)

# Closed frequent itemsets and maximal frequent itemsets

- Suppose that a transaction database has only two transactions:
  - {(*a1, a2, …, a100*); (*a1, a2,… , a50*)}
- Let the minimum support count threshold be *min_sup*=1.
- We find two closed frequent itemsets and their support counts, that is,
  - *C* = {{*a1, a2, …, a100*}:1; {*a1, a2,… , a50*}: 2}
- There is one maximal frequent itemset:
  - *M* = {{*a1, a2, …, a100*}:1}
    - (We cannot include {*a1, a2,… , a50*} as a maximal frequent itemset because it has a frequent super-set, {*a1, a2, …, a100*})

Source: Han & Kamber (2006)

# Frequent Pattern Mining

- Based on the *completeness of patterns to be mined*
- Based on the *levels of abstraction involved in the rule set*
- Based on the *number of data dimensions involved in the rule*
- Based on the *types of values handled in the rule*
- Based on the *kinds of rules to be mined*
- Based on the *kinds of patterns to be mined*

Source: Han & Kamber (2006)

# Based on the *levels of abstraction involved in the rule set*

- *buys(X, “computer”)) → buys(X, “HP printer”)*
- *buys(X, “laptop computer”)) → buys(X, “HP printer”)*

Source: Han & Kamber (2006)

# Based on the *number of data dimensions involved in the rule*

- Single-dimensional association rule
  - *buys(X, “computer”)) → buys(X, “antivirus software”)*
- Multidimensional association rule
  - *age(X, “30,…,39”) ^ income (X, “42K,…,48K”)) → buys (X, “high resolution TV”)*

Source: Han & Kamber (2006)

# Efficient and Scalable Frequent Itemset Mining Methods

- The Apriori Algorithm
  - Finding Frequent Itemsets Using Candidate Generation

Source: Han & Kamber (2006)

# Apriori Algorithm

- **Apriori** is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules.
- The name of the algorithm is based on the fact that the algorithm uses *prior knowledge of frequent itemset properties, as we shall* see following.

Source: Han & Kamber (2006)

# Apriori Algorithm

- Apriori employs an iterative approach known as a *level-wise search, where k-itemsets are used to explore (k+1)-itemsets.*
- *First, the set of frequent 1-itemsets is found* by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted $L_1$.
- *Next,* $L_1$ *is used to find* $L_2$, the set of frequent 2-itemsets, which is used to find $L_3$, *and so on, until no more frequent k-itemsets can be found.*
- *The finding of each* $L_k$ *requires one full scan of the database.*

Source: Han & Kamber (2006)

# Apriori Algorithm

- To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property.
- Apriori property
  - *All nonempty subsets of a frequent itemset must also be frequent.*

Source: Han & Kamber (2006)

- *How is the Apriori property used in the algorithm?*
  - How *$L_{k-1}$ is used to find $L_k$ for k >= 2.*
  - A two-step process is followed, consisting of join and prune actions.

# Apriori property used in algorithm
# 1. The join step

1. **The join step:** To find $L_k$, a set of **candidate** $k$-itemsets is generated by joining $L_{k-1}$ with itself. This set of candidates is denoted $C_k$. Let $l_1$ and $l_2$ be itemsets in $L_{k-1}$. The notation $l_i[j]$ refers to the $j$th item in $l_i$ (e.g., $l_1[k-2]$ refers to the second to the last item in $l_1$). By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. For the $(k-1)$-itemset, $l_i$, this means that the items are sorted such that $l_i[1] < l_i[2] < \ldots < l_i[k-1]$. The join, $L_{k-1} \bowtie L_{k-1}$, is performed, where members of $L_{k-1}$ are joinable if their first $(k-2)$ items are in common. That is, members $l_1$ and $l_2$ of $L_{k-1}$ are joined if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$. The condition $l_1[k-1] < l_2[k-1]$ simply ensures that no duplicates are generated. The resulting itemset formed by joining $l_1$ and $l_2$ is $l_1[1], l_1[2], \ldots, l_1[k-2], l_1[k-1], l_2[k-1]$.

Source: Han & Kamber (2006)

# Apriori property used in algorithm 2. The prune step

**2.** The **prune step:** $C_k$ is a superset of $L_k$, that is, its members may or may not be frequent, but all of the frequent $k$-itemsets are included in $C_k$. A scan of the database to determine the count of each candidate in $C_k$ would result in the determination of $L_k$ (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to $L_k$). $C_k$, however, can be huge, and so this could involve heavy computation. To reduce the size of $C_k$, the Apriori property is used as follows. Any $(k-1)$-itemset that is not frequent cannot be a subset of a frequent $k$-itemset. Hence, if any $(k-1)$-subset of a candidate $k$-itemset is not in $L_{k-1}$, then the candidate cannot be frequent either and so can be removed from $C_k$. This **subset testing** can be done quickly by maintaining a hash tree of all frequent itemsets.

Source: Han & Kamber (2006)

# Transactional data for an *AllElectronics* branch

| *TID* | *List of item_IDs* |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

Source: Han & Kamber (2006)

# Example: Apriori

- Let’s look at a concrete example, based on the *AllElectronics transaction database, D.*
- *There are nine transactions in this database, that is, |D| = 9.*
- Apriori algorithm for finding frequent itemsets in *D*

| *TID* | *List of item_IDs* |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

Source: Han & Kamber (2006)

# Example: Apriori Algorithm

**Generation of candidate itemsets and frequent itemsets, where the minimum support count is 2.**

| TID | List of item_IDs |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

Scan $D$ for count of each candidate →

$C_1$

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

Compare candidate support count with minimum support count →

$L_1$

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

Generate $C_2$ candidates from $L_1$ →

$C_2$

| Itemset |
|---|
| {I1, I2} |
| {I1, I3} |
| {I1, I4} |
| {I1, I5} |
| {I2, I3} |
| {I2, I4} |
| {I2, I5} |
| {I3, I4} |
| {I3, I5} |
| {I4, I5} |

Scan $D$ for count of each candidate →

$C_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I4} | 1 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |
| {I3, I4} | 0 |
| {I3, I5} | 1 |
| {I4, I5} | 0 |

Compare candidate support count with minimum support count →

$L_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |

Generate $C_3$ candidates from $L_2$ →

$C_3$

| Itemset |
|---|
| {I1, I2, I3} |
| {I1, I2, I5} |

Scan $D$ for count of each candidate →

$C_3$

| Itemset | Sup. count |
|---|---|
| {I1, I2, I3} | 2 |
| {I1, I2, I5} | 2 |

Compare candidate support count with minimum support count →

$L_3$

| Itemset | Sup. count |
|---|---|
| {I1, I2, I3} | 2 |
| {I1, I2, I5} | 2 |

Source: Han & Kamber (2006)

# Example: Apriori Algorithm
## $C_1 \rightarrow L_1$

| *TID* | *List of item_IDs* |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

Scan *D* for count of each candidate →

$C_1$

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

Compare candidate support count with minimum support count →

$L_1$

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

Source: Han & Kamber (2006)

# Example: Apriori Algorithm
# $C_2 \rightarrow L_2$

| TID | List of item_IDs |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

Scan $D$ for count of each candidate →

$C_1$

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

Compare candidate support count with minimum support count →

$L_1$

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

Generate $C_2$ candidates from $L_1$ →

$C_2$

| Itemset |
|---|
| {I1, I2} |
| {I1, I3} |
| {I1, I4} |
| {I1, I5} |
| {I2, I3} |
| {I2, I4} |
| {I2, I5} |
| {I3, I4} |
| {I3, I5} |
| {I4, I5} |

Scan $D$ for count of each candidate →

$C_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I4} | 1 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |
| {I3, I4} | 0 |
| {I3, I5} | 1 |
| {I4, I5} | 0 |

Compare candidate support count with minimum support count →

$L_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |

Source: Han & Kamber (2006)

| *TID* | *List of item_IDs* |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

Scan $D$ for count of each candidate →

$C_1$

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

Compare candidate support count with minimum support count →

$L_1$

| Itemset | Sup. count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

Generate $C_2$ candidates from $L_1$ →

$C_2$

| Itemset |
|---|
| {I1, I2} |
| {I1, I3} |
| {I1, I4} |
| {I1, I5} |
| {I2, I3} |
| {I2, I4} |
| {I2, I5} |
| {I3, I4} |
| {I3, I5} |
| {I4, I5} |

Scan $D$ for count of each candidate →

$C_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I4} | 1 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |
| {I3, I4} | 0 |
| {I3, I5} | 1 |
| {I4, I5} | 0 |

Compare candidate support count with minimum support count →

$L_2$

| Itemset | Sup. count |
|---|---|
| {I1, I2} | 4 |
| {I1, I3} | 4 |
| {I1, I5} | 2 |
| {I2, I3} | 4 |
| {I2, I4} | 2 |
| {I2, I5} | 2 |

# Example: Apriori Algorithm
# $C_3 \rightarrow L_3$

Generate $C_3$ candidates from $L_2$ →

$C_3$

| Itemset |
|---|
| {I1, I2, I3} |
| {I1, I2, I5} |

Scan $D$ for count of each candidate →

$C_3$

| Itemset | Sup. count |
|---|---|
| {I1, I2, I3} | 2 |
| {I1, I2, I5} | 2 |

Compare candidate support count with minimum support count →

$L_3$

| Itemset | Sup. count |
|---|---|
| {I1, I2, I3} | 2 |
| {I1, I2, I5} | 2 |

Source: Han & Kamber (2006)

# The Apriori algorithm for discovering frequent itemsets for mining Boolean association rules.

**Algorithm: Apriori.** Find frequent itemsets using an iterative level-wise approach based on candidate generation.

**Input:**

- $D$, a database of transactions;
- *min_sup*, the minimum support count threshold.

**Output:** $L$, frequent itemsets in $D$.

**Method:**

```
L_1 = find_frequent_1-itemsets(D);
for (k = 2; L_{k-1} ≠ φ; k++) {
    C_k = apriori_gen(L_{k-1});
    for each transaction t ∈ D { // scan D for counts
        C_t = subset(C_k, t); // get the subsets of t that are candidates
        for each candidate c ∈ C_t
            c.count++;
    }
    L_k = {c ∈ C_k | c.count ≥ min_sup}
}
return L = ∪_k L_k;

procedure apriori_gen(L_{k-1}:frequent (k − 1)-itemsets)
for each itemset l_1 ∈ L_{k-1}
    for each itemset l_2 ∈ L_{k-1}
        if (l_1[1] = l_2[1]) ∧ (l_1[2] = l_2[2]) ∧ ... ∧ (l_1[k−2] = l_2[k−2]) ∧ (l_1[k−1] < l_2[k−1]) then {
            c = l_1 ⋈ l_2; // join step: generate candidates
            if has_infrequent_subset(c, L_{k-1}) then
                delete c; // prune step: remove unfruitful candidate
            else add c to C_k;
        }
return C_k;

procedure has_infrequent_subset(c: candidate k-itemset;
        L_{k-1}: frequent (k − 1)-itemsets); // use prior knowledge
for each (k − 1)-subset s of c
    if s ∉ L_{k-1} then
        return TRUE;
return FALSE;
```

# The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

| Tid | Items |
|---|---|
| 10 | A, C, D |
| 20 | B, C, E |
| 30 | A, B, C, E |
| 40 | B, E |

1st scan →

$C_1$

| Itemset | sup |
|---|---|
| {A} | 2 |
| {B} | 3 |
| {C} | 3 |
| {D} | 1 |
| {E} | 3 |

→ $L_1$

| Itemset | sup |
|---|---|
| {A} | 2 |
| {B} | 3 |
| {C} | 3 |
| {E} | 3 |

$C_2$

| Itemset |
|---|
| {A, B} |
| {A, C} |
| {A, E} |
| {B, C} |
| {B, E} |
| {C, E} |

2nd scan →

$C_2$

| Itemset | sup |
|---|---|
| {A, B} | 1 |
| {A, C} | 2 |
| {A, E} | 1 |
| {B, C} | 2 |
| {B, E} | 3 |
| {C, E} | 2 |

→ $L_2$

| Itemset | sup |
|---|---|
| {A, C} | 2 |
| {B, C} | 2 |
| {B, E} | 3 |
| {C, E} | 2 |

$C_3$

| Itemset |
|---|
| {B, C, E} |

3rd scan →

$L_3$

| Itemset | sup |
|---|---|
| {B, C, E} | 2 |

Source: Han & Kamber (2006)

# The Apriori Algorithm

- Pseudo-code:

  $C_k$: Candidate itemset of size k
  $L_k$ : frequent itemset of size k

  $L_1$ = {frequent items};
  **for** ($k$ = 1; $L_k$ !=$\emptyset$; $k$++) **do begin**
  $C_{k+1}$ = candidates generated from $L_k$;
  **for each** transaction $t$ in database do
  increment the count of all candidates in $C_{k+1}$
  that are contained in $t$
  $L_{k+1}$ = candidates in $C_{k+1}$ with min_support
  **end**
  **return** $\cup_k L_k$;

Source: Han & Kamber (2006)

# Important Details of Apriori

- How to generate candidates?
  - Step 1: self-joining $L_k$
  - Step 2: pruning
- How to count supports of candidates?
- Example of Candidate-generation
  - $L_3$={*abc, abd, acd, ace, bcd*}
  - Self-joining: $L_3 * L_3$
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in $L_3$
  - $C_4$={*abcd*}

Source: Han & Kamber (2006)

# How to Generate Candidates?

- Suppose the items in $L_{k-1}$ are listed in an order
- Step 1: self-joining $L_{k-1}$

  insert into $\boldsymbol{C_k}$

  select $\boldsymbol{p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}}$

  from $\boldsymbol{L_{k-1}\ p, L_{k-1}\ q}$

  where $\boldsymbol{p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}}$
- Step 2: pruning

  forall ***itemsets c in*** $\boldsymbol{C_k}$ do

  forall ***(k-1)-subsets s of c*** do

  **if** *(s is not in* $L_{k-1}$*)* **then delete** *c* **from** $C_k$

Source: Han & Kamber (2006)

# Generating Association Rules from Frequent Itemsets

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}$$

- For each frequent itemset $l$, generate all nonempty subsets of $l$.
- For every nonempty subset $s$ of $l$, output the rule "$s \Rightarrow (l - s)$" if $\frac{support\_count(l)}{support\_count(s)} \geq min\_conf$, where $min\_conf$ is the minimum confidence threshold.

Source: Han & Kamber (2006)

# Example: Generating association rules

- frequent itemset *l = {I1, I2, I5}*

| *TID* | *List of item_IDs* |
|---|---|
| T100 | I1, I2, I5 |
| T200 | I2, I4 |
| T300 | I2, I3 |
| T400 | I1, I2, I4 |
| T500 | I1, I3 |
| T600 | I2, I3 |
| T700 | I1, I3 |
| T800 | I1, I2, I3, I5 |
| T900 | I1, I2, I3 |

$$
\begin{array}{ll}
I1 \wedge I2 \Rightarrow I5, & confidence = 2/4 = 50\% \\
I1 \wedge I5 \Rightarrow I2, & confidence = 2/2 = 100\% \\
I2 \wedge I5 \Rightarrow I1, & confidence = 2/2 = 100\% \\
I1 \Rightarrow I2 \wedge I5, & confidence = 2/6 = 33\% \\
I2 \Rightarrow I1 \wedge I5, & confidence = 2/7 = 29\% \\
I5 \Rightarrow I1 \wedge I2, & confidence = 2/2 = 100\%
\end{array}
$$

- If the minimum confidence threshold is, say, 70%, then only the second, third, and last rules above are output, because these are the only ones generated that are strong.

Source: Han & Kamber (2006)

# Summary

- Association Analysis
- Mining Frequent Patterns
- Association and Correlations
- Apriori Algorithm

Source: Han & Kamber (2006)

# References


- Jiawei Han and Micheline Kamber, Data Mining: Concepts and Techniques, Second Edition, 2006, Elsevier
- Efraim Turban, Ramesh Sharda, Dursun Delen, Decision Support and Business Intelligence Systems, Ninth Edition, 2011, Pearson.