

# Data Mining

---

CS57300

Purdue University

November 18, 2010

Pattern mining: representation & learning

# Data mining components

---

- Task specification: **Pattern discovery**
- Data representation: **Homogeneous IID data**
- Knowledge representation
- Learning technique

# Pattern discovery

---

- Models describe entire dataset (or large part of it)
- Pattern characterize local aspects of data
- Pattern: predicate that returns “true” for the instances in the data where the pattern occurs and “false” otherwise
- Task: find descriptive associations between variables

# Examples

---

- Supermarket transaction database
  - 10% of the customers buy wine and cheese
- Telecommunications alarms database
  - If alarms A and B occur within 30 seconds of each other then alarm C occurs within 60 seconds with  $p=0.5$
- Web log dataset
  - If a person visits the CNN website, there is a 60% chance the person will visit the ABC News website in the same month

# Pattern in tabular data

---

- Primitive pattern: subset of all possible observations over variables  $X_1, \dots, X_p$ 
  - If  $X_k$  is categorical then  $X_k=c$  is a primitive pattern
  - If  $X_k$  is ordinal then  $X_k \leq c$  is a primitive pattern
- Start from primitive patterns and combine using logical connectives such as AND and OR
  - $\text{age} < 40$  AND  $\text{income} < 100,000$
  - $\text{chips} = 1$  AND ( $\text{beer} = 1$  OR  $\text{soda} = 1$ )

# Pattern space

---

- Set of legal patterns
- Defined through the set of primitive patterns and operators to combine primitives
- Example
  - If variable  $X_1, \dots, X_p$  are all binary we can define the space of patterns to be all conjunctions of the form  
 $(X_{i_1}=1) \text{ AND } (X_{i_2}=1) \text{ AND } \dots \text{ AND } (X_{i_k}=1)$

# Pattern properties

---

- Patterns that occur frequently are often of interest
  - Frequency  $fr(\theta)$  of a pattern  $\theta$  is defined as the relative number of observations in the dataset about which  $\theta$  is true
  - Also known as **support**
- May also be interested in:
  - Novelty
  - Understandability
  - Informativeness

# Pattern discovery task

---

- Find all “interesting” patterns in the data
- Approach: find all patterns that satisfy certain conditions
- Challenge: find the right balance between
  - Pattern complexity
  - Pattern understandability
  - Computational complexity

Knowledge representation

# Association rule mining

---

- Task:
  - Find frequent patterns, associations, correlations, or causal structures among items
- Data:
  - Transaction databases, relational database, or other information repositories
- Applications:
  - Market basket analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification

# Rule

---

- A rule is an expression of the form  $\theta \rightarrow \varphi$
- Association rules:
  - All variables are binary  $\{A_1, \dots, A_k\} \rightarrow \{B_1, \dots, B_h\}$
  - Probabilistic statement about the co-occurrence of certain events in the database
- Mining rules
  - Number of rules grows exponentially with dimensions
  - How to find patterns in an efficient manner?
  - How to determine which rules are interesting?

# Example

The image shows a software window titled "Qt Association Rules Tree Viewer". On the left, there are two panels: "Shown measures" with checkboxes for Support, Confidence, Lift, Leverage, Strength, and Coverage (all checked); and "Options" with a "Tree depth" slider set to 2 and a "Display whole rules" checkbox (unchecked). The main area displays a tree of association rules. The root node is "age=adult", which branches into several sub-nodes. Each node contains one or more rules, each with associated numerical values for Support, Confidence, Lift, Leverage, Strength, and Coverage.

Rules	Supp	Conf	Lift	Lev	Strg	Cov
age=adult						
-> sex=male survived=no	0.604	0.635	1.025	0.015	0.652	0.950
-> status=crew	0.402	0.423	1.052	0.020	0.423	0.950
-> status=crew sex=male	0.392	0.412	1.052	0.019	0.412	0.950
-> survived=no	0.653	0.687	1.015	0.010	0.712	0.950
sex=male						
-> status=crew	0.392	0.517	1.286	0.087	0.531	0.757
-> survived=no	0.604	0.797	1.178	0.091	0.894	0.757
sex=male						
-> age=adult survived=no	0.604	0.768	1.175	0.090	0.831	0.786
-> status=crew	0.392	0.498	1.238	0.075	0.511	0.786
-> status=crew age=adult	0.392	0.498	1.238	0.075	0.511	0.786
-> survived=no	0.620	0.788	1.164	0.087	0.861	0.786
age=adult						
-> status=crew	0.392	0.517	1.286	0.087	0.531	0.757
-> survived=no	0.604	0.797	1.178	0.091	0.894	0.757

# Sequential pattern mining

---

- Task:
  - Find frequent substring patterns
- Data:
  - Sequence data, biological data, text collections
- Applications:
  - Bioinformatics, text analysis, clustering, classification

# Sequential patterns

---

- Substring
- Regular expression
- Episode
  - Partially ordered collection of events occurring together
  - Can take time or sequential ordering into account but is insensitive to intervening events
  - Eg. headache followed by sense of disorientation within a given time period

# Graph mining

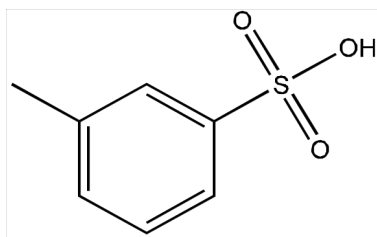
---

- Task:
  - Find frequent subgraph patterns
- Data:
  - Graph databases or relational databases
- Applications:
  - Graph indexing, similarity search, clustering, classification

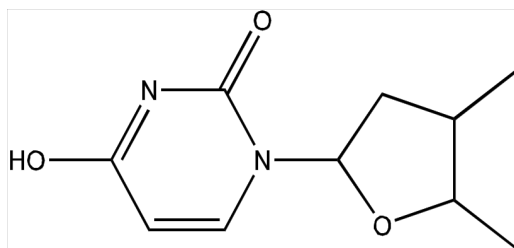
# Subgraphs

---

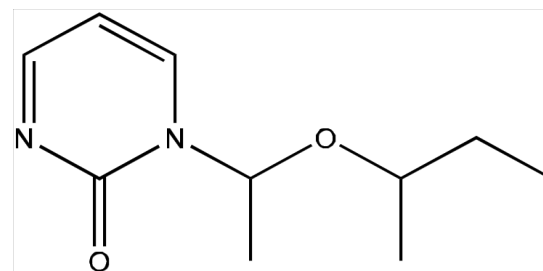
## GRAPH DATASET



(A)



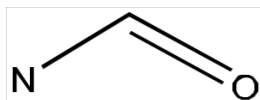
(B)



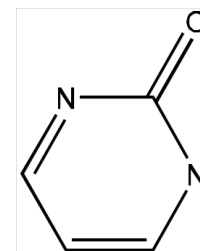
(C)

## FREQUENT PATTERNS (MIN SUPPORT IS 2)

(1)



(2)



Learning

# Search problem

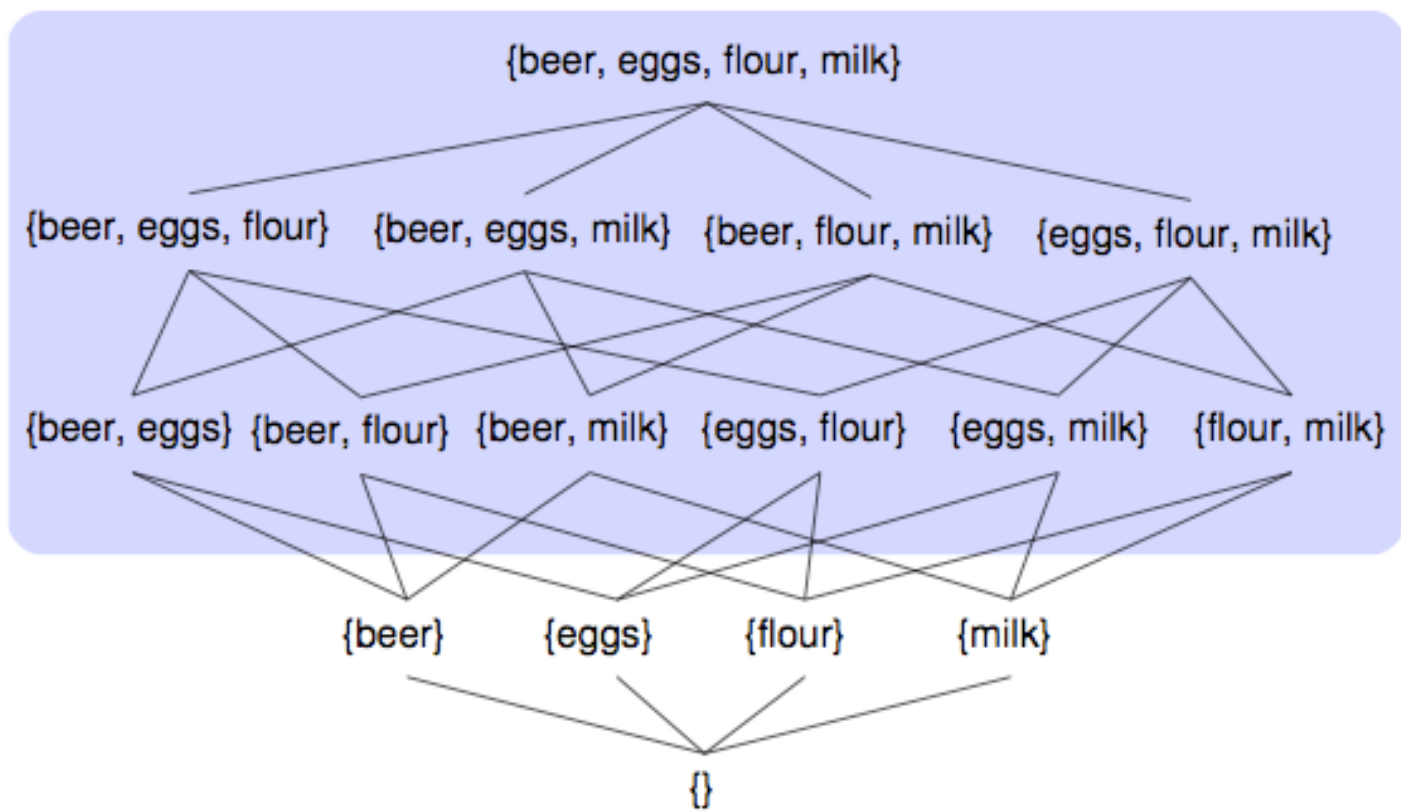
---

- Searching for all patterns is computationally intractable
- Consider market basket data where each row has 1000 binary variables
- How many possible patterns?

Transaction ID	beer	eggs	flour	milk
1	0	1	1	1
2	1	1	0	0
3	0	1	0	1
4	0	1	1	1
5	0	0	0	1

# Example

---



# Solution

---

- Take advantage of the nature of the score function to prune parts of the search space and reduce run time
- The Apriori principle:
  - Any subset of a frequent itemset must be frequent

# Rule evaluation

---

- Support or frequency
  - $fr(\theta \rightarrow \varphi)$
  - Proportion of items with antecedent  $\theta$  and consequent  $\varphi$
- Confidence or accuracy
  - $p(\varphi | \theta) = fr(\theta \wedge \varphi) / fr(\theta)$
  - Proportion of items which have antecedent  $\theta$  that also have consequent  $\varphi$

# Association rules

---

- Find all rules that satisfy the constraints:
  - Confidence is greater than threshold  $c$
  - Support is greater than threshold  $s$
- Data
  - Basket: customer transaction; items: products
  - Basket: document; items: words
  - Basket: web pages; items: links
- E.g., 98% of people who purchase tires and auto accessories also have automotive service done

# Example

---

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Min. support 50%  
Min. confidence 50%

Frequent Itemset	Support
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

For rule  $A \Rightarrow C$ :

support =  $\text{support}(\{A,C\}) = 50\%$

confidence =  $\text{support}(\{A,C\})/\text{support}(\{A\}) = 66.6\%$

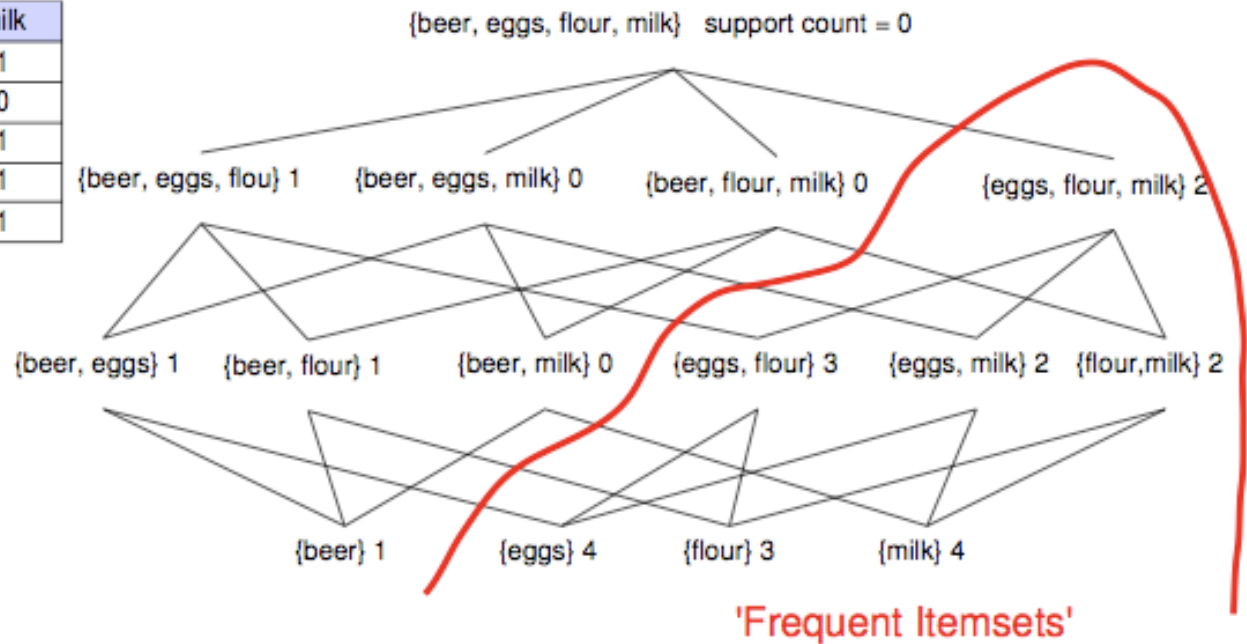
# Finding frequent itemsets

---

- Find sets of items with minimum support
- Support is monotonic
  - A subset of a frequent itemset must also be frequent
  - Eg. If  $\{A,B\}$  is a frequent itemset then both  $\{A\}$  and  $\{B\}$  are frequent itemsets as well
- Approach
  - Iteratively find frequent itemsets with cardinality from 1 to  $k$  ( $k$ -itemset)
  - Prune any sets of size  $k$  that are not frequent

# Example

Transaction ID	beer	eggs	flour	milk
1	0	1	1	1
2	1	1	1	0
3	0	1	0	1
4	0	1	1	1
5	0	0	0	1

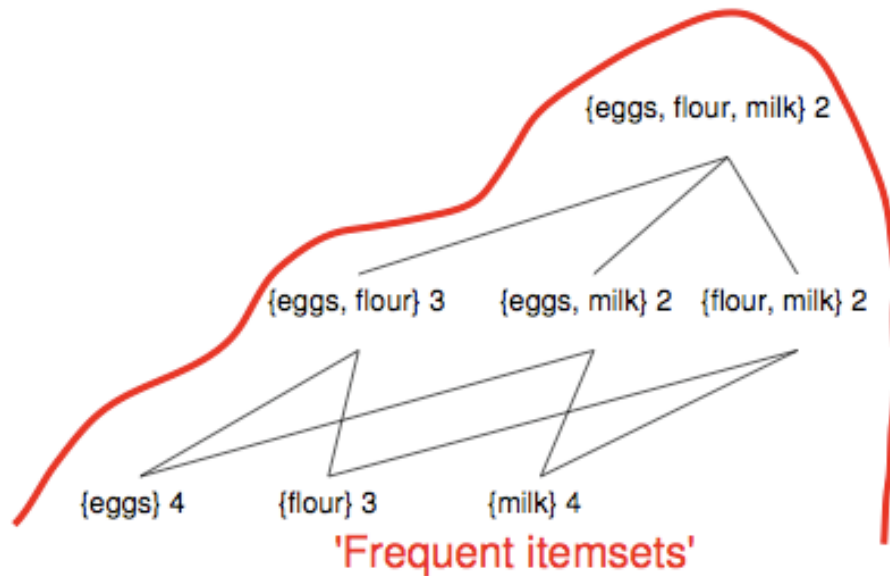


# Apriori

---

- Use frequent itemsets to form association rules
- Given a pruned list of candidate frequent sets of size  $k$ 
  - Algorithm performs a linear scan of the data to determine which of these sets are frequent
- Confirmed frequent sets of size  $k$  are combined to generate possible frequent sets of size  $k+1$ 
  - Followed by another pruning step
  - Cardinality of largest frequent set is quite small (relative to  $m$ ) for large support values

# Example



		Confidence
{eggs}	→ {flour}	$3/4 = 0.75$
{flour}	→ {eggs}	$3/3 = 1$
{eggs}	→ {milk}	$2/4 = 0.5$
{milk}	→ {eggs}	$2/4 = 0.5$
{flour}	→ {milk}	$2/3 = 0.67$
{milk}	→ {flour}	$2/4 = 0.5$
{eggs, flour}	→ {milk}	$2/3 = 0.67$
{eggs, milk}	→ {flour}	$2/2 = 1$
{flour, milk}	→ {eggs}	$2/2 = 1$
{eggs}	→ {flour, milk}	$2/4 = 0.5$
{flour}	→ {eggs, milk}	$2/3 = 0.67$
{milk}	→ {eggs, flour}	$2/4 = 0.5$

# Algorithm

---

- $C_k$ : candidate itemset of size  $k$
- $L_k$ : frequent itemset of size  $k$
- $L_1 = \{\text{frequent single items}\}$
- for ( $k=1$ ;  $L_k \neq \emptyset$ ;  $k++$ )
  - $C_{k+1} = \text{candidates generated from } L_k$
  - for each transaction  $t$  in  $D$ 
    - increment the count of all candidates in  $C_{k+1}$  contained in  $t$
  - $L_{k+1} = \text{candidates in } C_{k+1} \text{ with min\_support}$
- Return  $\bigcup_k L_k$

# Example

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

$C_1$

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

$L_1$

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

$C_2$

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

Scan D

$C_2$

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

$L_2$

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

$C_3$

itemset
{2 3 5}

Scan D

$L_3$

itemset	sup
{2 3 5}	2