# CSC7072: Databases, fall 2015

Dr. Kim Bauters

data mining

## Data Mining

what is data mining?

what is data mining (knowledge discovery)?

it is the extraction of useful patterns from data sources
*e.g.* databases, texts, web, images, social media, blogs, forums *etc.*

a pattern is *useful* when:

- valid;
- novel;
- potentially useful; and
- understandable

## Data Mining

one name for a variety of tasks

typical data mining tasks

- classification
  is the process of mining patterns so that we can readily classify new instances into classes we already know (*e.g.* is this new email spam?)

- **association rule mining**
  mining rules of the form X → Y with X and Y data sets
  (e.g. cheese, milk → bread [sup=5%, conf=80%])

- clustering
  identifying a set of similarity groups in data
  (*e.g.* tweets *w.r.t.* classes taught in conversion MSc in QUB 2015-16)

## Data Mining

one name for a variety of tasks

typical data mining tasks cont.

- sequential pattern mining
  a sequential rule of the form A → B (A, then B) says that event A will be immediately followed by event B with a given confidence
- deviation detection
  discover the most significant changes in the data
  (*e.g.* are all logins to my server still ok or is there odd behaviour?)
- data visualisation
  using graphical methods to show patterns in data

## Data Mining

### importance of data mining

so why is data mining important?

it can offer a huge competitive advantage (and pressure)!

the data is readily available:

- computerisation of business has lead to huge amounts;
- online e-business produce *even more* data!

*e.g.* companies like Amazon are data-driven businesses;
search engines like Google are information retrieval
and data mining companies.

computing power is typically not an issue

data mining tries to answer: *how can we best use all this data?*

## Data Mining

the need for data mining

why can't we just use *"the data"*?

there is a big gap between the stored data, and knowledge; the transition to knowledge doesn't happen automatically …

many interesting things require more than database queries:

- find people likely to buy my product;
- who is likely to respond to this promotion; or
- which products should I recommend to this customer?

the need for this is everywhere: marketing, engineering (identify problems), bioinformatics, fraud detection, language processing *etc.*

## Data Mining

general steps in data mining

general approach to data mining:

1. understand the application domain;
2. identify data sources, and target data;
3. pre-process: clean, select important attributes ...;
4. **perform data mining** to extract patters or models;
5. post-processing: identify interesting patters/knowledge;
6. incorporate patters/knowledge in real life

## Data Mining

association rule mining: introduction

association rule mining

proposed in 1993 by Agrawal, Imielinski, and Swami

it is an important data mining model used and studied extensively  within the database and data mining community

initially used for Market Basket Analysis:

cheese, milk → bread [support=5%, confidence=80%]

*i.e.* find how items purchased by customers are relates

# Data Mining

## association rule mining: data model

our data model:

we have a set of items (*e.g.* products): $I = \{\overset{\text{cheese}}{\overset{\uparrow}{i_1}}, \ldots, \overset{\text{milk}}{\overset{\uparrow}{i_n}}\}$

*i.e.* the items you can buy

we have a transaction (*e.g.* a basket): $t \subseteq I$

*i.e.* the items someone bought on one shopping trip

*e.g.* { milk, honey, bread }

we have a transaction database: $T = \{t_1, \ldots, t_m\}$

*i.e.* the shopping trips of all the customers

*e.g.* $t_1$ = { milk, honey, bread }

$t_2$ = { chicken, tomato, mozzarella }

…

$t_m$ = { chicken, honey }

# Data Mining

## association rule mining: data model

our data model:

we have a set of items (*e.g.* products): $I = \{\overset{\text{cheese}}{\overset{\uparrow}{i_1}}, \ldots, \overset{\text{milk}}{\overset{\uparrow}{i_n}}\}$

*i.e.* the items you can buy

we have a transaction (*e.g.* a basket): $t \subseteq I$

*i.e.* the items someone bought on one shopping trip

*e.g.* { milk, honey, bread }

we have a transaction database: $T = \{t_1, \ldots, t_m\}$

*i.e.* the shopping trips of all the customers

*e.g.* $t_1$ = { milk, honey, bread }

$t_2$ = { chicken, tomato, mozzarella }

…

$t_m$ = { chicken, honey }

simplistic view of a shopping basket (*missing:* price, quantity, …), but often good enough!

## Data Mining

association rule mining: data model

this applies to a lot of problems!

we have a set of items (*e.g. a bag of* words):
*i.e.* the words that can be found in a twitter message

we have a transaction (*e.g.* a tweet):
*i.e.* the words used together to form a tweet
*e.g.* { brilliant, course, CS7052 }

we have a transaction database:
*i.e.* the tweets from this semester

*e.g.* $t_1$ = { brilliant, course, CS7052 }
$t_2$ = { CS7052, databases, #QUB, course }
…
$t_m$ = { CS7052, awesome }

## Data Mining

association rule mining: data model

our data model: some extra terminology

a transaction $t_i$ **contains** $X$ when $X \subseteq t_i$

an **association rule** is an implication of the form $X \to Y$
where $X, Y \subset I$ and $X \cap Y = \emptyset$
*i.e.* both must be sets of items, and there must be no overlap

an **itemset** is a set of items
*e.g.* X = { milk, honey, bread } is an itemset

a ***k*-itemset** is a set with *k* items (with *k* some positive natural number)
*e.g.* X = { milk, honey, bread } is a 3-itemset

## Data Mining

association rule mining: rule strength

measuring the strength of a rule X → Y:

**support:** a rule is said to hold with support *sup* in *T* if *sup*% of the transactions contain both *X* and *Y*

$$sup = \Pr(X \cup Y) \qquad support = \frac{\text{count}(X \cup Y)}{m}$$

**confidence:** a rule holds with confidence *conf* in *T* if *conf*% of the transactions containing *X* also contain *Y*

$$conf = \Pr(Y \mid X) \qquad confidence = \frac{\text{count}(X \cup Y)}{count(X)}$$

where count(A) counts the number of transaction satisfying A

## Data Mining

association rule mining: rule strength

measuring the strength of a rule X → Y:

**support:** a rule is said to hold with support *sup* in *T* if *sup*% of the transactions contain both *X* and *Y*

$$\boxed{sup = \Pr(X \cup Y)} \qquad support = \frac{\text{count}(X \cup Y)}{m}$$

**confidence:** a rule holds with confidence *conf* in *T* if *conf*% of the transactions containing *X* also contain *Y*

$$\boxed{conf = \Pr(Y \mid X)} \qquad confidence = \frac{\text{count}(X \cup Y)}{count(X)}$$

where count(A) counts the number of transaction satisfying A

# Data Mining

## association rule mining: rule strength

measuring the strength of a rule X → Y:

**support:** a rule is said to hold with support *sup* in *T* if *sup*% of the transactions contain both *X* and *Y*

$sup = \Pr(X \cup Y)$

$$\boxed{support = \frac{\text{count}(X \cup Y)}{m}}$$

**confidence:** a rule holds with confidence *conf* in *T* if *conf*% of the transactions containing *X* also contain *Y*

$conf = \Pr(Y \mid X)$

$$\boxed{confidence = \frac{\text{count}(X \cup Y)}{count(X)}}$$

where count(A) counts the number of transaction satisfying A

## Data Mining

association rule mining: our goal

so … what is the **goal** of association rule mining?

find all rules with a user-specified minimum support/confidence!

ex: $t_1$ = { beef, chicken, milk }

$t_2$ = { beef, cheese }

$t_3$ = { cheese, boots }

$t_4$ = { beef, chicken, cheese }

$t_5$ = { beef, chicken, clothes, cheese, milk }

$t_6$ = { chicken, clothes, milk }

$t_7$ = { chicken, clothes, milk }

assume minsup=30% , minconf=80%:

clothes → milk, chicken [sup=3/7, conf=3/3]

… … …

clothes, chicken → milk [sup=3/7, conf=3/3]

## Data Mining

### association rule mining: our goal

how can we do the rule mining in an automated way?

many, many algorithms exist

- they use different strategies and data structures
- they should all find the same set of rules!
- their main difference is in speed/memory usage/...

we look at one of them (best known one): **apriori algorithm**

## Data Mining

### apriori algorithm

the apriori algorithm works in two steps:

1. find all itemsets that have at least the desired support
   *a.k.a.* frequent itemsets, usually up to maximum size of $K$

2. use frequent itemsets to generate rules

for example, { chicken, clothes, milk } can be a frequent itemset as it occurs in 3 out of 7 rules, *i.e.* it has sup=3/7

it gives rise to the association rule: clothes, chicken → milk

## Data Mining

apriori algorithm: step 1

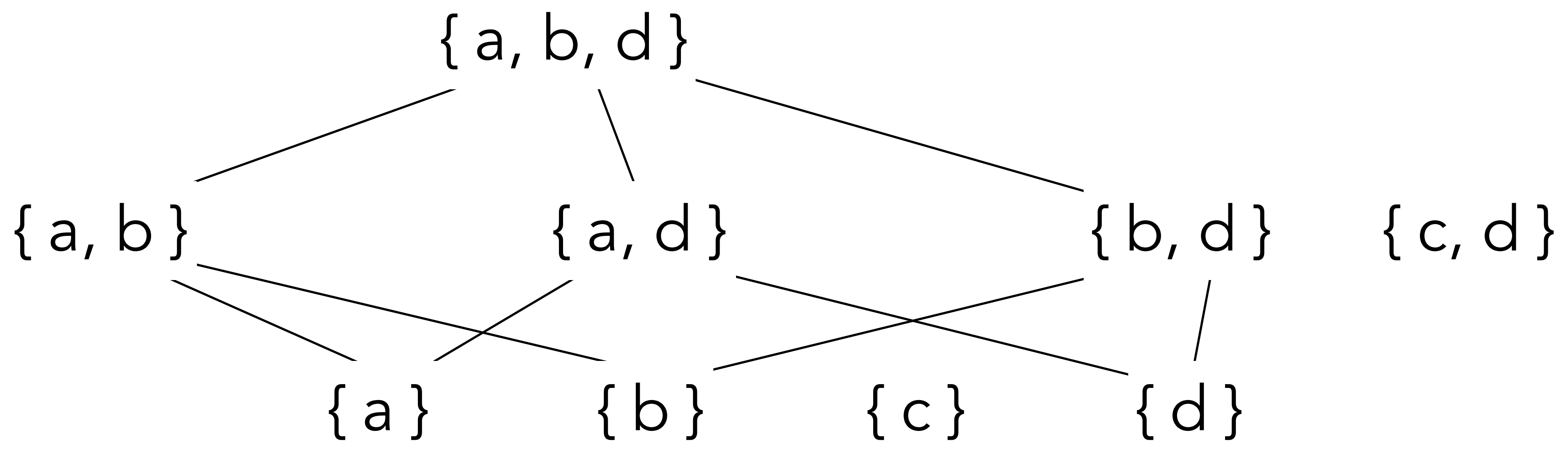
**step 1:** finding frequent itemsets

assume we are looking for all rules where minsup = 50%

hence, we are looking for itemsets such that sup ≥ 50%

interesting observation:

any subset of a frequent itemset is also a frequent itemset!



# Data Mining

## apriori algorithm: step 1

**step 1:** algorithm

iterative algorithm (also called a *level-wise search*)

*basic idea:* find all 1-item frequent itemsets, then find all
2-item frequent itemsets, then find …

in iteration k, only consider k-1 frequent itemsets (see previous slide)

find 1-frequent itemsets, call this $F_1$

then repeatedly:

create $C_k$, *i.e.* all candidate itemsets
do this by making all possible combinations of $F_{k-1}$

create $F_k$ from $C_k$, *i.e.* those candidates that *are frequent*
do this we need to scan the transaction database once

# Data Mining

apriori algorithm: step 1

**step 1:** algorithm example

ex: $t_1 = \{ a, c, d \}$

$t_2 = \{ b, c, e \}$

$t_3 = \{ a, b, c, e \}$

$t_4 = \{ b, e \}$

$C_1 = \{ a \}, \quad \{ b \}, \quad \{ c \}, \quad \{ d \}, \quad \{ e \}$

$F_1 = \{ a \}{:}2, \{ b \}{:}3, \{ c \}{:}3,$ ~~$\{ d \}{:}1$~~, $\{ e \}{:}3$

since only $2/4 \geq 50\%$

$C_2 = \{ a, b \}, \quad \{ a, c \}, \quad \{ a, e \}, \quad \{ b, c \}, \quad \{ b, e \}, \quad \{ c, e \}$

$F_2 =$ ~~$\{ a, b \}{:}1$~~, $\{ a, c \}{:}2,$ ~~$\{ a, e \}{:}1$~~, $\{ b, c \}{:}2, \{ b, e \}{:}3, \{ c, e \}{:}2$

$C_3 = \{ a, b, c \}, \quad \{ a, c, e \}, \quad \{ b, c, e \}$

$F_3 =$ ~~$\{ a, b, c \}{:}1$~~, ~~$\{ a, c, e \}{:}1$~~, $\{ b, c, e \}{:}2$

algorithm stops because $F_4$ would be empty (we only have 3 items!)

## Data Mining

apriori algorithm: step 2

**step 2:** generating rules from frequent itemsets

```
for each frequent itemset F,
  for each proper non-empty subset $A \subset F$,     i.e. some elements of F
    take $B = F \setminus A$                           i.e. all other elements

    then $A \rightarrow B$ is an association rule if $conf(A \rightarrow B) \geq minconf$
    where  $confidence(A \rightarrow B) = support(A \cup B) / support(A)$
                                       $= count(A \cup B) / count(A)$
                                       $= count(F) / count(A)$
```

notice that $sup(A \rightarrow B) = sup(A \cup B) = sup(F)$ (since $A \cup B = F$)

# Data Mining

## apriori algorithm: step 2

**step 2:** generation example

we had that $F_3$ is { b, c, e }

all possible association rules are:

| | |
|---|---|
| b, c → e | confidence = 2/2 |
| b, e → c | confidence = 2/3 |
| c, e → b | confidence = 2/2 |
| b → c, e | confidence = 2/3 |
| c → b, e | confidence = 2/3 |
| e → b, c | confidence = 2/3 |

assume we are looking for minconf ≥ 0.75

the final rules are b, c → e and c, e → b

ex: $t_1$ = { a, c, d }
$t_2$ = { b, c, e }
$t_3$ = { a, b, c, e }
$t_4$ = { b, e }

# Data Mining

## apriori algorithm: step 2

**step 2:** generation example

we had that $F_3$ is { b, c, e }

all possible association rules are:

| | |
|---|---|
| b, c → e | confidence = 2/2 |
| b, e → c | confidence = 2/3 |
| c, e → b | confidence = 2/2 |
| b → c, e | confidence = 2/3 |
| c → b, e | confidence = 2/3 |
| e → b, c | confidence = 2/3 |

assume we are looking for minconf ≥ 0.75

the final rules are b, c → e and c, e → b

ex: {b, c, e}:2

{b, c}:2

{c, e}:2

{b, e}:3

{b}:3

{c}:3

{e}:3

now using

count(F) / count(A)

# Data Mining

## apriori algorithm: conclusion

the apriori algorithm in summary:

- it is a level-wise search
  we start with small itemsets, increasingly try to get bigger ones
- we limit the search to itemsets of maximum $K$
  typically this is bounded to *e.g.* 10
- the algorithm makes at most K passes over the database
  this is used to count the support level
- the algorithm can be very fast …
- but can use a lot of memory (hence research into other algorithms)