Machine Learning for Business

Module 13: Association Rule Mining
Day 7, 9.00 – 12.00

Asst. Prof. Dr. Santitham Prom-on

Department of Computer Engineering, Faculty of Engineering
King Mongkut's University of Technology Thonburi

Overview

Learning Outcome

- Understand the basic concepts of association rule
- Analyze itemsets
- Perform association analysis

Agenda

- Itemsets
- Association rules
- Rule generation

Association rule mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

{Diaper} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {Milk},

Implication means co-occurrence, not causality!

Frequent itemset

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - e.g. σ({Milk, Bread, Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - e.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

Frequent itemset

- **Association rule**
  - An implication expression of the form X → Y, where X and Y are itemsets
  - Example:

    {Milk, Diaper} → {Beer}
- **Rule evaluation metrics**
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measure how often items in Y appear in transactions that contain X

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Association rule mining tasks

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- Brute-force approach
  - List all possible association rules
  - Compute the support and confidence of each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds
  - Computationally prohibitive!!!

Mining association rules

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

Observation:

- All the above rules are binary partitions of the same itemset:
  - {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

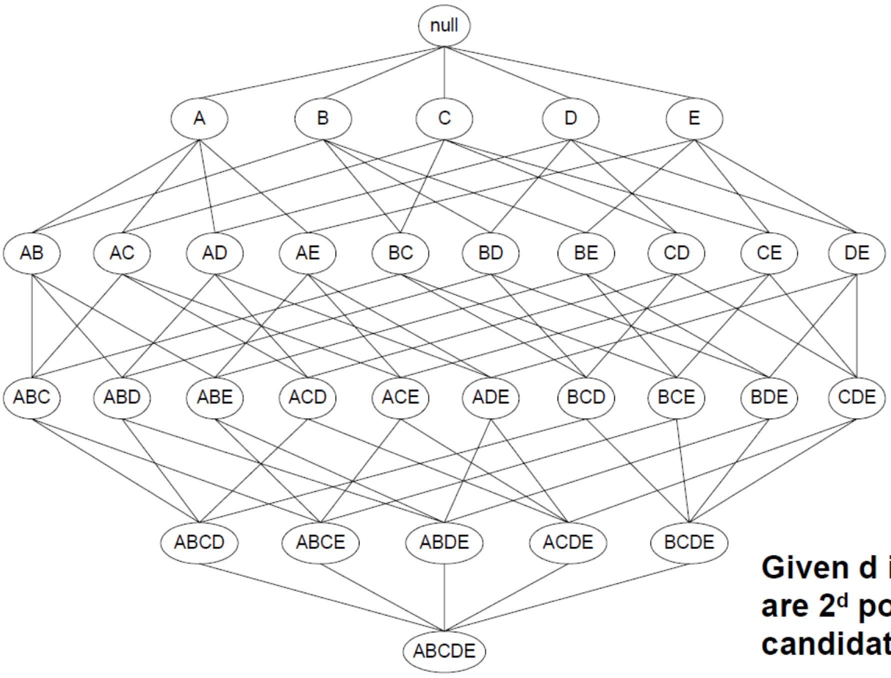
Mining association rules

Two-step approach:

1. Frequent itemset generation
   - Generate all itemsets whose support ≥ minsup
2. Rule generation
   - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

- Frequent itemset generation is still computationally expensive

Frequent itemset generation



Given d items, there are 2^d possible candidate itemsets

Frequent itemset generation

Brute-force approach:

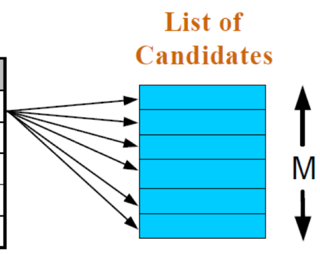
- Each itemset in the lattice is a candidate frequent itemset
- Count the support of each candidate by scanning the database

Transactions

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

N

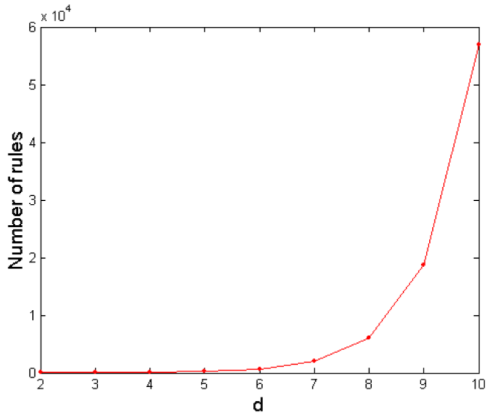
w



- Match each transaction against every candidate
- Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!

Computational complexity

- Given d unique items:
  - Total number of itemsets = 2^d
  - Total number of possible association rules:



$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right]$$
$$= 3^d - 2^{d+1} + 1$$

If d=6, R = 602 rules

Frequent itemset generation strategies

- Reduce the number of candidates (M)
  - Complete search: M = 2^d
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce the size of N as the size of itemset increases
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

Reducing number of candidates

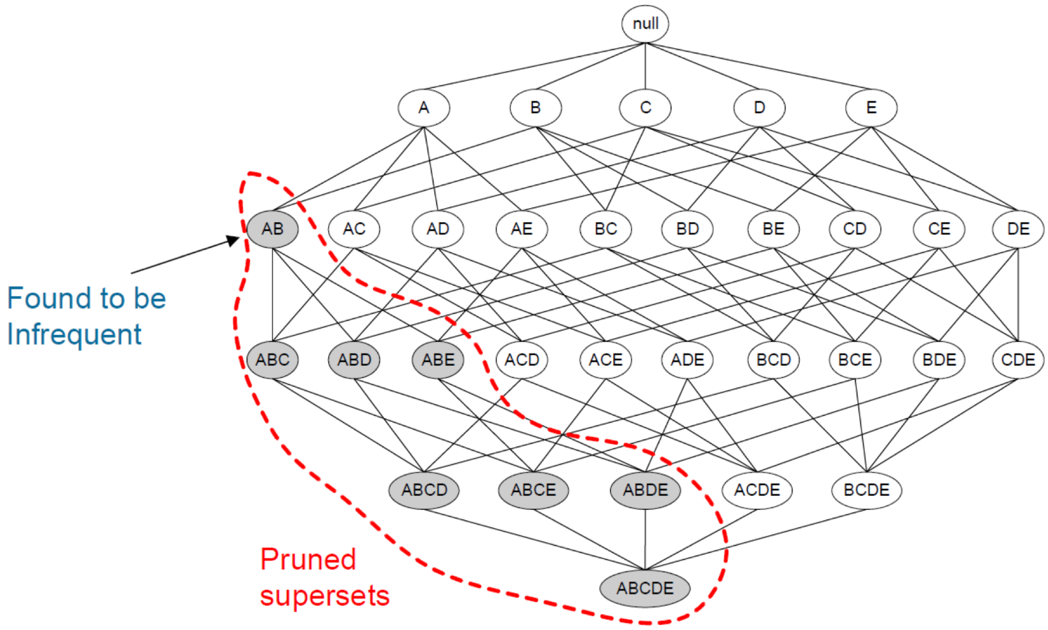
- Apriori principle
  - If an itemset is frequent, then all of its subsets must also be frequent

- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

  - Support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of support

Illustrating apriori principle



Illustrating apriori principle

Items (1-itemsets)

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

If every subset is considered,
$^6C_1 + {}^6C_2 + {}^6C_3 = 41$
With support-based pruning,
$6 + 6 + 1 = 13$

Apriori algorithm – Method

- Let k = 1
- Generate frequent itemset of length 1
- Repeat until no new frequent itemsets are identified
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Prune candidate itemsets containing subsets of length k that are infrequent
  - Count the support of each candidate by scanning the DB
  - Eliminate candidates that are infrequent, leaving only those that are frequent

Closed itemset

- An itemset is closed if none of its immediate supersets has the same support as the itemset

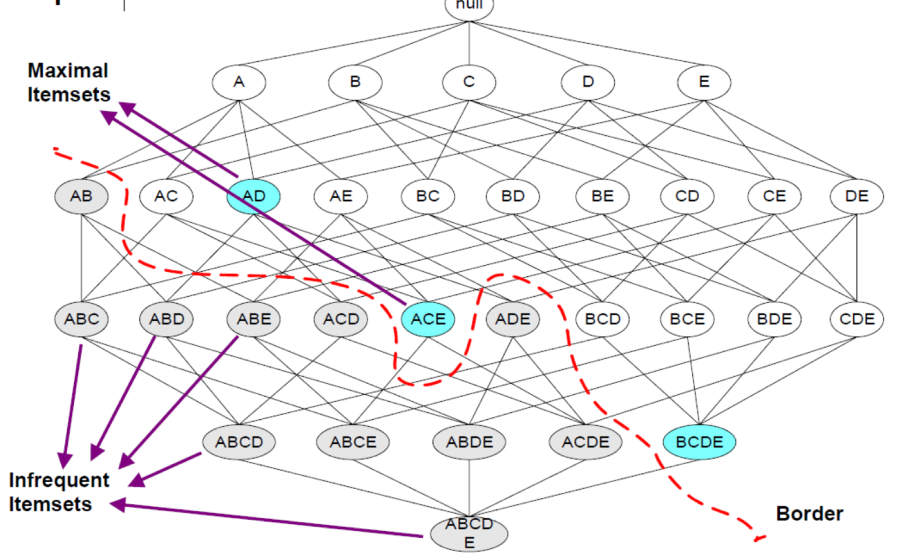
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

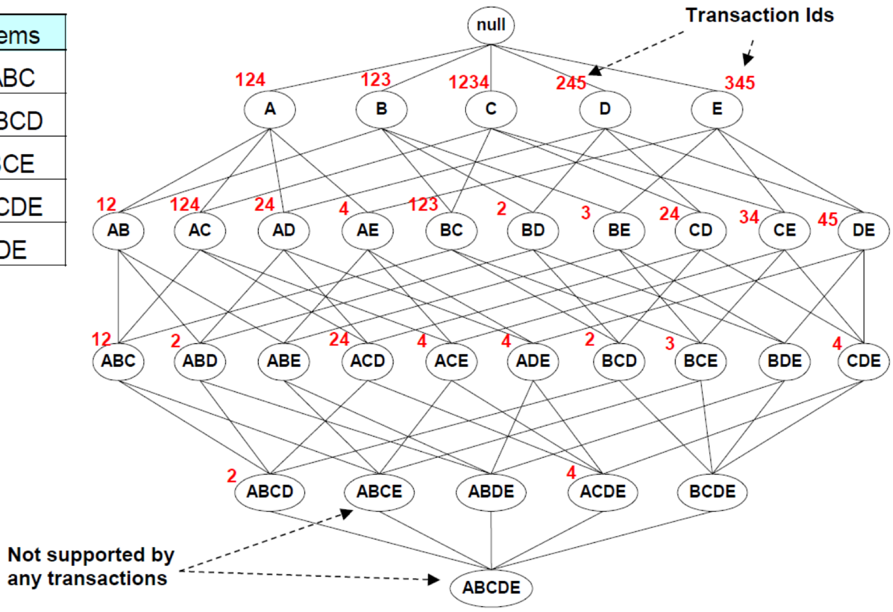
Maximal frequent itemset

An itemset is maximal frequent if none of its immediate supersets is frequent



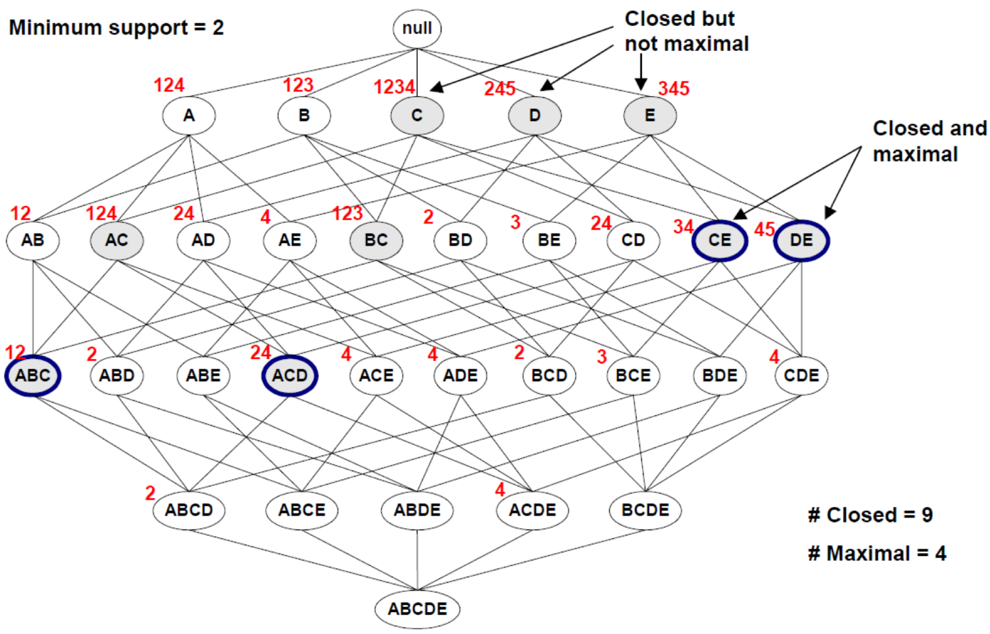
Maximal vs closed itemsets

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Maximal vs closed itemsets

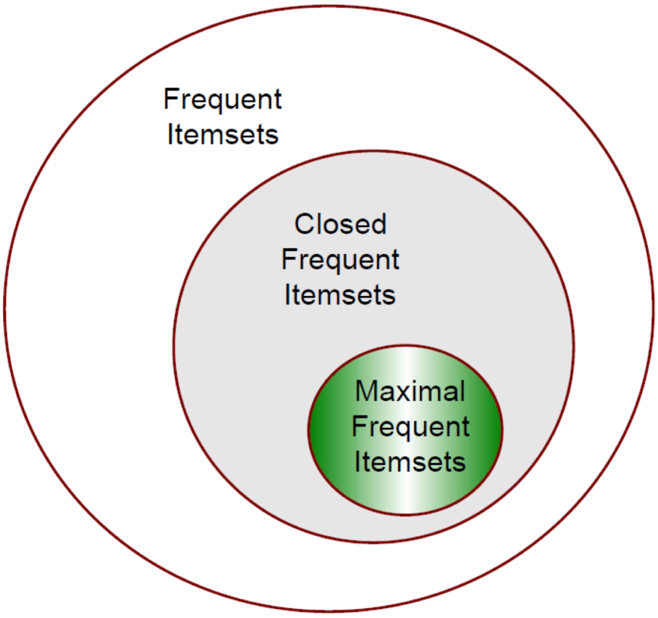
Minimum support = 2



Closed = 9

Maximal = 4

Maximal vs closed itemsets



Rule generation

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
  - If {A,B,C,D} is a frequent itemset, candidate rules:

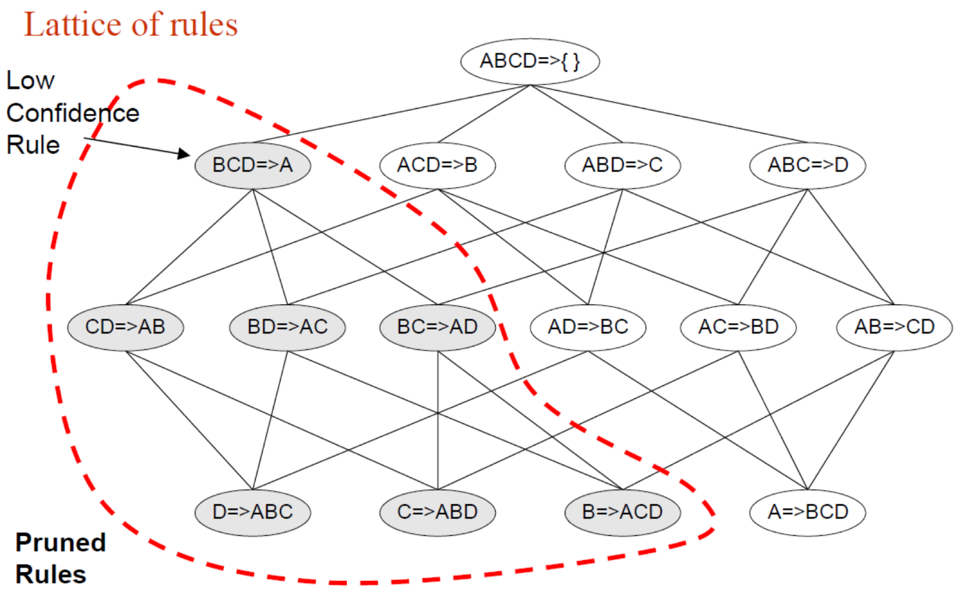
ABC →D,	ABD →C,	ACD →B,	BCD →A,
A →BCD,	B →ACD,	C →ABD,	D →ABC
AB →CD,	AC → BD,	AD → BC,	BC →AD,
BD →AC,	CD →AB,		

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \emptyset$ and $\emptyset \rightarrow L$)

Rule generation

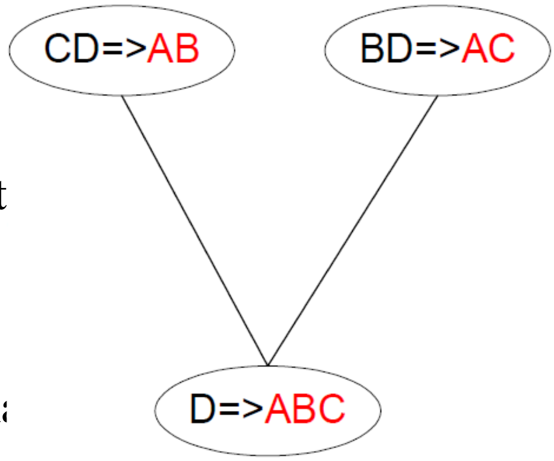
- How to efficiently generate rules from frequent itemsets?
  - In general, confidence does not have an antimonotone property
    $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$
  - But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., L = {A,B,C,D}:
    $c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$
  - Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

Rule generation for apriori algorithm



Rule generation for apriori algorithm

- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent

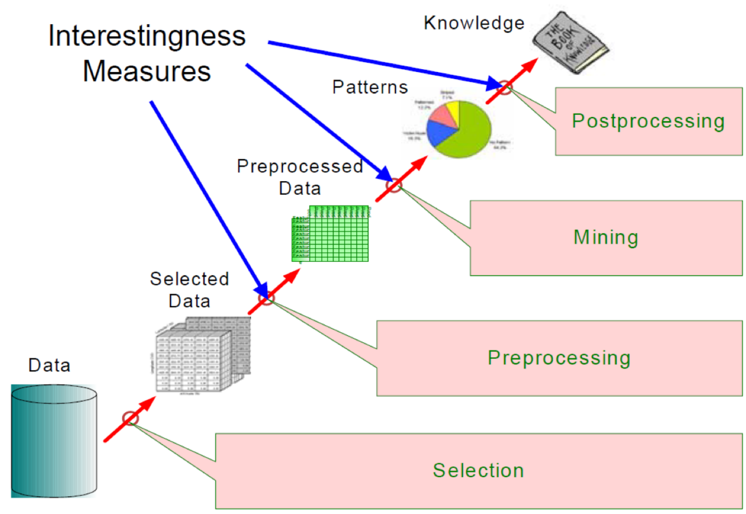


- join(CD=>AB,BD=>AC) would produce the candidat rule D => ABC
- Prune rule D=>ABC if its subset AD=>BC does not h high confidence

Pattern evaluation

- Association rule algorithms tend to produce too many rules
  - many of them are uninteresting or redundant
  - Redundant if {A,B,C} → {D} and {A,B} → {D} have same support & confidence
- Interestingness measures can be used to prune/rank the derived patterns
- In the original formulation of association rules, support & confidence are the only measures used

Application of interesting measures



Computing interestingness measure

- Given a rule $X \rightarrow Y$, information needed to compute rule interestingness can be obtained from a contingency table

Contingency table for $X \rightarrow Y$

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	$\|T\|$

f_{11}: support of X and Y
f_{10}: support of X and $\overline{Y}$
f_{01}: support of $\overline{X}$ and Y
f_{00}: support of $\overline{X}$ and $\overline{Y}$

Used to define various measures

- support, confidence, lift, Gini, J-measure, etc.

Drawback of confidence

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea → Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.9

⇒ Although confidence is high, rule is misleading

⇒ P(Coffee|$\overline{\text{Tea}}$) = 0.9375

Statistical independence

Population of 1000 students

- 600 students know how to swim (S)
- 700 students know how to bike (B)
- 420 students know how to swim and bike (S,B)

- $P(S \wedge B) = 420/1000 = 0.42$
- $P(S) \times P(B) = 0.6 \times 0.7 = 0.42$

- $P(S \wedge B) = P(S) \times P(B)$ => Statistical independence
- $P(S \wedge B) > P(S) \times P(B)$ => Positively correlated
- $P(S \wedge B) < P(S) \times P(B)$ => Negatively correlated

Statistical-based measure

- Measures that take into account statistical dependence

$$Lift = \frac{P(Y \mid X)}{P(Y)}$$

Lift/Interest

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea $\rightarrow$ Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.9

$\Rightarrow$ Lift = 0.75/0.9= 0.8333 (< 1, therefore is negatively associated)

Drawback of lift/interest

	Y	$\overline{Y}$	
X	10	0	10
$\overline{X}$	0	90	90
	10	90	100

	Y	$\overline{Y}$	
X	90	0	90
$\overline{X}$	0	10	10
	90	10	100

$$Lift = \frac{0.1}{(0.1)(0.1)} = 10$$

$$Lift = \frac{0.9}{(0.9)(0.9)} = 1.11$$

Statistical independence:

If P(X,Y)=P(X)P(Y) => Lift = 1

Summary

- Association analysis allows us to derive frequent patterns from a huge amount of data
- Combining with statistical evaluation, interesting frequent patterns can be found and could be utilized further
- Further studies in this direction
  - Collaborative filtering
  - Content-based filtering

R Lab

Association Rule Mining

Install packages

- Install the following packages using install.packages
  - arules
  - arulesViz

R: Load data

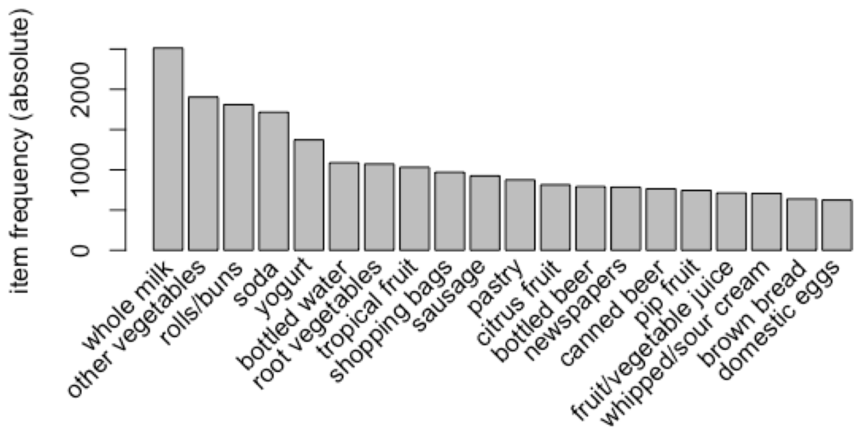
```
data("Groceries")
summary(Groceries)
inspect(Groceries[1:3])
```

```
    items
[1] {citrus fruit,
     semi-finished bread,
     margarine,
     ready soups}
[2] {tropical fruit,
     yogurt,
     coffee}
[3] {whole milk}
```

```
transactions as itemMatrix in sparse format with
 9835 rows (elements/itemsets/transactions) and
 169 columns (items) and a density of 0.02609146
```

R: visualize the transactions

```
itemFrequencyPlot(Groceries, topN = 20,
type = "absolute")
```



How to prepare transactions for arules?

- Create transactions from a list
- Create transactions from a matrix

R: transactions from a list

```
a_list <- list( c("a","b","c"), c("a","b"),
                c("a","b","d"), c("c","e"),
                c("a","b","d","e") )
## set transaction names
names(a_list) <- paste("Tr",c(1:5), sep = "")
a_list
## coerce into transactions
trans1 <- as(a_list, "transactions")
## analyze transactions
summary(trans1)
inspect(trans1)
```

R: transaction from a matrix

```
a_matrix <- matrix(c( 1,1,1,0,0, 1,1,0,0,0, 1,1,0,1,0,
0,0,1,0,1, 1,1,0,1,1 ), ncol = 5)
## set dim names
dimnames(a_matrix) <- list(
paste("Tr",c(1:5), sep = ""),
c("a","b","c","d","e"))
a_matrix
## coerce
trans2 <- as(a_matrix, "transactions")
trans2
inspect(trans2)
```

Mining association rules

- Let's setting the following parameters
  - We set the minimum support to 0.001
  - We set the minimum confidence of 0.8
  - We then show the top 5 rules

```
rules <- apriori(Groceries,
                 parameter = list(supp = 0.001,
                                  conf = 0.8))
```

```
Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       5   0.001      1
 maxlen target   ext
     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.01s].
writing ... [410 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

Inspect unsorted results

```
options(digits = 2)
inspect(rules[1:5])
```

```
    lhs                          rhs            support confidence lift count
[1] {liquor,red/blush wine}   => {bottled beer} 0.0019  0.90       11.2 19
[2] {curd,cereals}            => {whole milk}   0.0010  0.91        3.6 10
[3] {yogurt,cereals}          => {whole milk}   0.0017  0.81        3.2 17
[4] {butter,jam}              => {whole milk}   0.0010  0.83        3.3 10
[5] {soups,bottled beer}      => {whole milk}   0.0011  0.92        3.6 11
```

Rule summary

```
summary(rules)
```

```
set of 410 rules

rule length distribution (lhs + rhs):sizes
  3   4   5   6
 29 229 140  12

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    3.0     4.0     4.0     4.3     5.0     6.0

summary of quality measures:
    support           confidence        lift            count
 Min.   :0.00102   Min.   :0.80   Min.   : 3.1   Min.   :10.0
 1st Qu.:0.00102   1st Qu.:0.83   1st Qu.: 3.3   1st Qu.:10.0
 Median :0.00122   Median :0.85   Median : 3.6   Median :12.0
 Mean   :0.00125   Mean   :0.87   Mean   : 4.0   Mean   :12.3
 3rd Qu.:0.00132   3rd Qu.:0.91   3rd Qu.: 4.3   3rd Qu.:13.0
 Max.   :0.00315   Max.   :1.00   Max.   :11.2   Max.   :31.0

mining info:
      data ntransactions support confidence
 Groceries          9835   0.001        0.8
```

Sorting rules

```
rules_sorted  <- sort(rules,
             by="support",
             decreasing = TRUE)
inspect(rules_sorted[1:5])
```

```
     lhs                    rhs                   support confidence lift count
[1] {citrus fruit,
     tropical fruit,
     root vegetables,
     whole milk}         => {other vegetables} 0.0032       0.89  4.6    31
[2] {other vegetables,
     curd,
     domestic eggs}      => {whole milk}       0.0028       0.82  3.2    28
[3] {hamburger meat,
     curd}               => {whole milk}       0.0025       0.81  3.2    25
[4] {herbs,
     rolls/buns}         => {whole milk}       0.0024       0.80  3.1    24
[5] {tropical fruit,
     herbs}              => {whole milk}       0.0023       0.82  3.2    23
```

Limit rule length

```
rules <- apriori(Groceries,
                 parameter = list(supp = 0.001,
                                  conf = 0.8,
                                  maxlen  = 3))
```

```
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [29 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
Warning message:
In apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8,  :
  Mining stopped (maxlen reached). Only patterns up to a length of 3 returned!
```

Top rules with max length of 3

```
rules_sorted  <- sort(rules,
                       by="confidence",
                       decreasing = TRUE)
inspect(rules_sorted[1:5])
```

```
    lhs                          rhs           support confidence lift count
[1] {rice,
     sugar}                   => {whole milk}  0.0012       1.00  3.9    12
[2] {canned fish,
     hygiene articles}        => {whole milk}  0.0011       1.00  3.9    11
[3] {whipped/sour cream,
     house keeping products} => {whole milk}  0.0012       0.92  3.6    12
[4] {rice,
     bottled water}           => {whole milk}  0.0012       0.92  3.6    12
[5] {soups,
     bottled beer}            => {whole milk}  0.0011       0.92  3.6    11
```

Targeting items

- Now that we know how to generate rules, limit the output, lets say we wanted to target items to generate rules.

- There are two types of targets we might be interested in that are illustrated with an example of "whole milk":
  - What are customers likely to buy before buying whole milk
  - What are customers likely to buy if they purchase whole milk?

- This essentially means we want to set either the Left Hand Side and Right Hand Side.

Targeting items - RHS

```
rules <- apriori(Groceries,
          parameter = list(supp = 0.001,
                           conf = 0.8),
          appearance = list(default = "lhs",
                            rhs = "whole milk"))
rules_sorted  <- sort(rules,
                      by="confidence",
                      decreasing = TRUE)
inspect(rules_sorted[1:5])
```

```
    lhs                          rhs            support confidence lift count
[1] {rice,
     sugar}                   => {whole milk}  0.0012            1  3.9    12
[2] {canned fish,
     hygiene articles}        => {whole milk}  0.0011            1  3.9    11
[3] {root vegetables,
     butter,
     rice}                    => {whole milk}  0.0010            1  3.9    10
[4] {root vegetables,
     whipped/sour cream,
     flour}                   => {whole milk}  0.0017            1  3.9    17
[5] {butter,
     soft cheese,
     domestic eggs}           => {whole milk}  0.0010            1  3.9    10
```

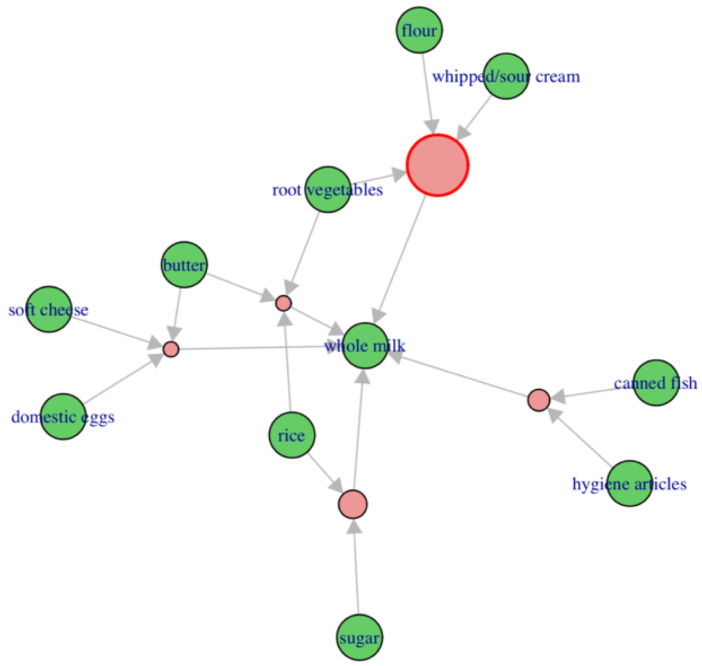
Targeting items - LHS

```
rules <- apriori(Groceries,
          parameter = list(supp = 0.001,
                           conf = 0.15,
                           minlen = 2),
          appearance = list(default = "rhs",
                            lhs = "whole milk"))
rules_sorted  <- sort(rules, by="confidence",
decreasing = TRUE)
inspect(rules_sorted[1:5])
```

```
    lhs             rhs                support confidence lift count
[1] {whole milk} => {other vegetables} 0.075   0.29       1.5  736
[2] {whole milk} => {rolls/buns}       0.057   0.22       1.2  557
[3] {whole milk} => {yogurt}           0.056   0.22       1.6  551
[4] {whole milk} => {root vegetables}  0.049   0.19       1.8  481
[5] {whole milk} => {tropical fruit}   0.042   0.17       1.6  416
```

Visualize the rules

```
rules <- apriori(Groceries,
                 parameter = list(supp = 0.001,
                                  conf = 0.8))
rules_sorted  <- sort(rules,
                      by="confidence",
                      decreasing = TRUE)
library(arulesViz)
plot(rules_sorted[1:5], method = "graph",
     interactive = TRUE, shading = NA)
```



Activity

- Adjust the arules parameters, determine the behavior of the itemset and rule generations

Questions

- Which items that are likely to be bought before purchasing beers?

Thank you

Question?