COMP5121
Data Mining and Data Warehousing Applications

Week 5: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

Dr. Fengmei Jin

- Email: fengmei.jin@polyu.edu.hk
- Office: PQ747 (+852 3400 3327)
- Consultation Hours: 2.30-4.30 pm every Thursday

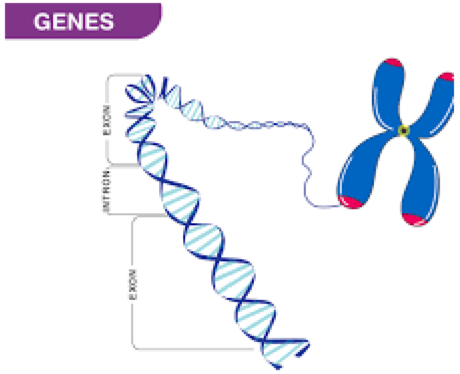
Outline

- Basic Concepts
  - Pattern discovery, frequent itemsets, association rules, support, confidence, closed patterns, maximal patterns
- Frequent Itemset Mining Methods
  - Apriori: downward closure property
  - Other more efficient methods
- Pattern Evaluation Methods – *Interestingness*
  - Support-Confidence framework
  - Lift and Chi-Square
  - Null-invariant measures

What Is Pattern Discovery?

- To identify meaningful relationships, trends, structures, etc.





- Frequent Pattern Mining: a key method in pattern discovery
  - Search for **patterns** that occur frequently in massive data
    - Frequent occurrences of items (e.g., products purchased together)
    - Sequential patterns (e.g., events or actions happening in a sequence)
    - Structured patterns (e.g., subgraphs in networks)

What Is Pattern Discovery?

- Motivation examples in real world
  - What products were often purchased together?
  - What combinations of symptoms frequently co-occur among patients?
  - What are the subsequent purchases after buying an iPad?
  - What word sequences likely form phrases in a corpus?
  - What kinds of network structures indicate influential groups in Twitter?
  - What spatial patterns in road networks imply traffic congestion?

Why Is 'Frequent Pattern' Important?

- Revealing inherent regularities and important properties of data
- Foundation for many essential data mining tasks:
  - Association, correlation, and causality analysis
  - Classification: Discriminative pattern-based analysis
  - Cluster analysis: Pattern-based subspace clustering
  - Other various patterns in spatiotemporal, multimedia, time-series data
- Broad applications:
  - E-commerce: market-basket analysis, cross-marketing, catalog design
  - Management: sale campaign analysis, Web log analysis
  - Healthcare: biological sequence analysis

Market Basket Analysis: A Motivating Example

- *"Which groups or sets of items are customers likely to purchase on a given trip to the store?"*
  - Finding frequent itemsets
    - Items frequently purchased together by customers
  - **Benefits:**
    - Design store layouts,
    - Understand buying habits
    - Plan marketing / advertising strategies

Basic Concepts: Frequent Itemsets

- **Itemset**: A set of one or more items
  - ***k*-itemset**: $X = \{x_1, \ldots, x_k\}$ with k items
- Support of an itemset
  - **Absolute Support (Count)**: the number of transactions containing the given itemset X
  - **Relative Support**: the fraction of transactions containing X (i.e., the probability that a transaction contains X)

- **Frequent Itemset:** An itemset X is *frequent* if the support of X is no less than σ – a *minsup* threshold.

Basic Concepts: Frequent Itemsets

- Let *minsup* = 50%
- Frequent 1-itemsets:
  - **Beer**: 3 (60%)
  - **Nuts**: 3 (60%)
  - **Eggs**: 3 (60%)
  - **Diaper**: 4 (80%)
- Frequent 2-itemsets:
  - **{Beer, Diaper}**: 3 (60%)

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

From Frequent Itemsets to Association Rules

- **Association Rules** written as X → Y [support, confidence]
  - Both X and Y are non-empty itemsets, and $X \cap Y = \emptyset$.
  - It describes an '*if-then*' relationship between two sets of items.

  - **Support**: The percentage of transactions containing both X and Y

$$\sup(X \rightarrow Y) = \mathrm{P}(X \cup Y)$$

    - $P(X \cup Y)$: the percentage of transactions that contains every item in X and Y, i.e., how frequently both X and Y appear together in the dataset
  - **Confidence**: The conditional probability that a transaction having X also contains Y, that is,

$$\mathrm{conf}(X \rightarrow Y) = \mathrm{P}(Y|X) = \sup(X \rightarrow Y)/\sup(X)$$

Association Rule Mining

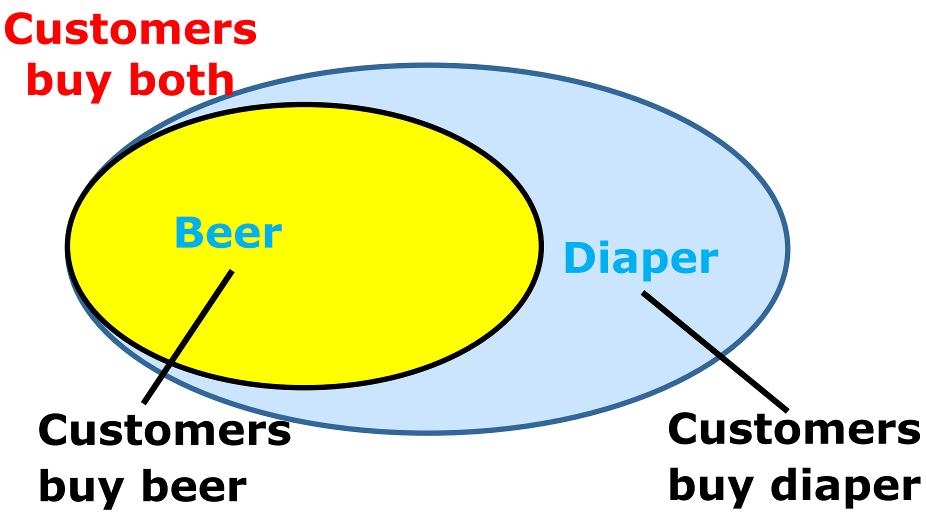
- **Strong Rules:** Find all rules, $X \rightarrow Y$ that satisfy
  - minimum support: frequency of X and Y appear together.
  - minimum confidence: likelihood that Y occurs when X occurs.

- What does a strong rule do really?
  - It correlates the presence of one set of items with another set.
  - Applications:
    - $* \rightarrow \boldsymbol{Y}$: What actions can boost sales of Y?
    - $\boldsymbol{X} \rightarrow *$: What other products should be stocked up if X is popular?

Association Rule Mining: An Example

- **Frequent itemsets**: Let *minsup* = 50%
  - Freq. 1-itemset: Beer: 3; Nuts: 3; Eggs: 3; Diaper: 4
  - Freq. 2-itemsets: {Beer, Diaper}: 3
- **Association rules**: Let *minconf* = 50%
  - Beer → Diaper (60%, 100%)
  - Diaper → Beer (60%, 75%)

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



Challenge: *Too Many Frequent Patterns!*

- Long patterns generate an exponential number of sub-patterns.
- Given two transactions with minimum support = 1:
  - $\boldsymbol{T_1}$: $\{\boldsymbol{a_1, \dots, a_{50}}\}$; $\boldsymbol{T_2}$: $\{\boldsymbol{a_1, \dots, a_{100}}\}$.
  - How many frequent itemsets?
    - 1-itemsets: $\{a_1\}$: 2, $\{a_1\}$: 2, …, $\{a_{50}\}$: 2, $\{a_{51}\}$: 1, …, $\{a_{100}\}$: 1
    - 2-itemsets: $\{a_1, a_2\}$: 2, …, $\{a_1, a_{50}\}$: 2, $\{a_1, a_{51}\}$: 1, …, …, $\{a_{99}, a_{100}\}$: 1
    - …, …, …, …
    - 99-itemsets: $\{a_1, a_2, \dots, a_{99}\}$: 1, …, $\{a_2, a_3, \dots, a_{100}\}$: 1
    - 100-itemsets: $\{a_1, a_2, \dots, a_{100}\}$: 1
  - In total: $\binom{100}{1} + \binom{100}{2} + \cdots + \binom{100}{100} = 2^{100} - 1$ sub-patterns!

Too huge for any computer to compute or store!

Expressing Patterns in Compressed Form: **Closed Patterns**

- How to handle such a scalability challenge?
- **Solution 1: Closed patterns**

> **An itemset X is a closed pattern if:**
> - X is frequent
> - NO super-pattern $Y \supset X$ exists with the same support as X.

  - Given T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$ with *minsup* = 1
    - Only two closed patterns → $\{\boldsymbol{a_1, \ldots, a_{50}}\}: \mathbf{2}$, $\{\boldsymbol{a_1, \ldots, a_{100}}\}: \mathbf{1}$

- Closed pattern is a lossless compression of frequent patterns.
  - Reduces # patterns to process
  - Retains all support information: "$\{a_2, \ldots, a_{40}\}: 2$", "$\{a_5, \ldots, a_{51}\}: 1$", …

Expressing Patterns in Compressed Form: **Maximal Patterns**

- **Solution 2: Max-patterns**

A pattern X is a maximal pattern if:
- X is frequent
- NO frequent super-pattern $Y \supset X$ exist.

  - Given T_1: $\{a_1, \ldots, a_{50}\}$; T_2: $\{a_1, \ldots, a_{100}\}$ with *minsup* = 1
    - Only one max-pattern → $\{a_1, \ldots, a_{100}\}: 1$
- Limitation: Maximal patterns are lossy compression!
  - Compared to close patterns, this method does NOT reveal the real support for sub-patterns of a max-pattern.
  - Example: We only know $\{a_1, \ldots, a_{40}\}$ is frequent, but cannot know its real support.

Therefore, **closed patterns** is more desirable than maximal patterns.

Apriori, FPGrowth, strong association rules, …

FREQUENT ITEMSET MINING METHODS

A Frequent Pattern Implies Frequent Subsets

Key Observation

- Given T_1: $\{a_1, \ldots, a_{50}\}$ and T_2: $\{a_1, \ldots, a_{100}\}$, we get a frequent itemset: $\{a_1, \ldots, a_{50}\}$
- All subsets are also frequent: $\{a_1\}$, $\{a_2\}$, …, $\{a_{50}\}$, $\{a_1, a_2\}$, …, $\{a_1, \ldots, a_{49}\}$, …

There must be some hidden relationships among these frequent patterns!

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

Min. support 50%

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

The Downward Closure Property in Apriori

- **Any subset of a frequent itemset must be frequent!**
  - e.g., if *{beer, diaper, nuts}* is frequent, so is *{beer, diaper}*
    - Reason: Every transaction containing *{beer, diaper, nuts}* also contains *{beer, diaper}*.

- **Apriori**: an efficient mining algorithm with downward closure.
  - **Pruning strategy:** If any subset of an itemset S is infrequent, then S cannot be frequent. In this case, why consider S at all?
  - Eliminate infrequent itemsets early. Focus on promising ones.

Apriori Pruning and Scalable Mining Methods

- **Apriori**'s pruning principle
  - If any itemset is infrequent, its superset should not even be generated!
  - *Example*: If {A, B} is infrequent, {A, B, C} will not be considered.

- Scalable Mining Methods
  - **Apriori:** Level-wise, join-based approach (Agrawal, et al. @VLDB'94)
    - Iterative procedure: k-itemsets are used to explore $(k+1)$-itemsets.
  - **Eclat:** Use *vertical data format* to compute intersections of transactions (Zaki, et al. @KDD'97)
  - **FP-growth:** Avoid candidate generation by building a compact FP-tree (Han, et al. @SIGMOD'00)

The Apriori Algorithm: Framework

- Outline of **Apriori**: level-wise, candidate generation and test

- Initially, scan DB once to get frequent 1-itemset
- **Repeat**
  - Generate length-$(k+1)$ candidate itemsets based on frequent k-itemsets
  - Test the candidates against DB to find frequent $(k+1)$-itemsets
  - Set $k := k+1$
- **Until** no frequent or candidate set can be generated
- Return all the frequent itemsets derived

The Apriori Algorithm: Pseudo-Code

- C_k: Candidate itemsets of size k
- F_k: Frequent itemsets of size k

```
k = 1;
F_k = {frequent items};                                          // frequent 1-itemset
While (F_k ≠ ∅) Do {                                             // as long as F_k is non-empty
    C_{k+1} = candidates generated from F_k;                     // candidate generation
    Derive F_{k+1} by counting candidates in C_{k+1} w.r.t. TDB at minsup;    // test candi
    k = k + 1;
}
Return ∪_k F_k;                                                  // return F_k generated at each level
```

The Apriori Algorithm: An Example

minsup = 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

F_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

F_2

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

C_3

Itemset
{B, C, E}

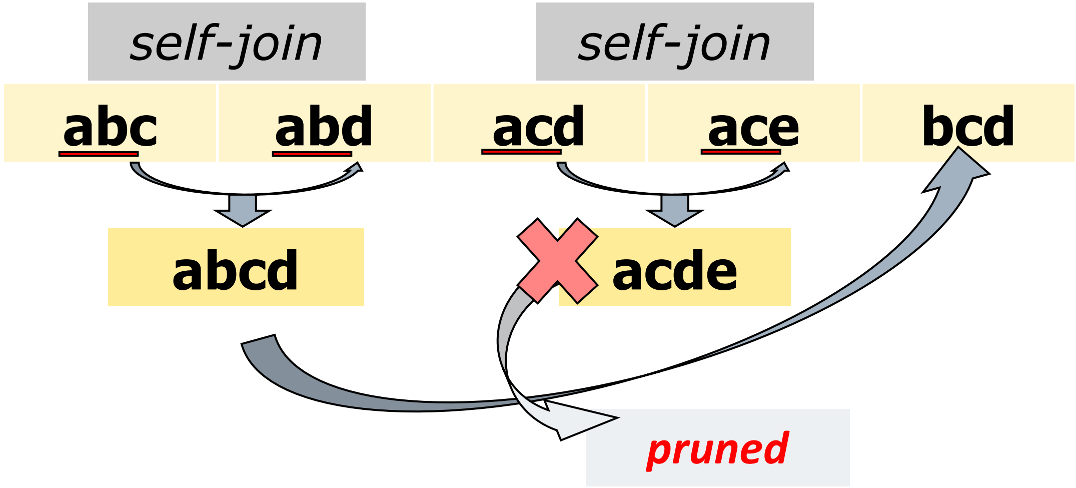
3rd scan

F_3

Itemset	sup
{B, C, E}	2

Candidate Generation in Apriori

- Efficiently generate candidates
  - Step 1: self-join F_k to generate candidates of size $k+1$
  - Step 2: prune candidates whose subsets are not all frequent



- Example
  - **Input**: $F_3 = \{abc, abd, acd, ace, bcd\}$
  - **Self-joining**: $F_3 \times F_3 \rightarrow abcd, acde$
  - **Pruning**: $acde$ is removed because ade is NOT in F_3
  - **Output:** $C_4 = \{abcd\}$

Candidate Generation: An SQL Implementation

- Suppose the items in F_{k-1} are listed in an order (e.g., bac → abc)

Step 1: self-joining F_{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 $\boldsymbol{F_{k-1}}$ as $\boldsymbol{p}$, $\boldsymbol{F_{k-1}}$as $\boldsymbol{q}$

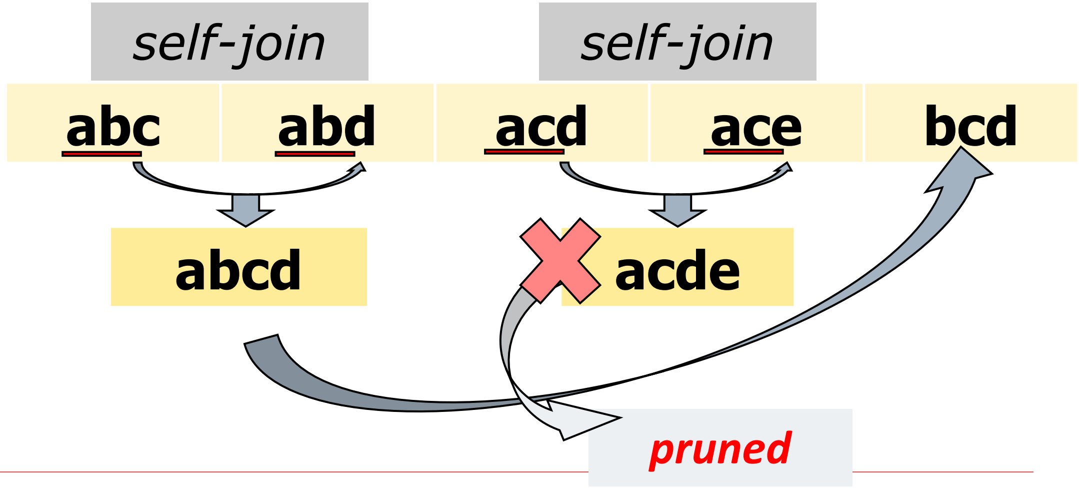
where $\boldsymbol{p.item_1 = q.item_1, \ldots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}}$

Step 2: pruning

for all itemsets $\boldsymbol{c}$ in $\boldsymbol{C_k}$ do

for all $(k-1)$-subsets $\boldsymbol{s}$ of $\boldsymbol{c}$ do

if ($\boldsymbol{s}$ is not in $\boldsymbol{F_{k-1}}$) then delete $\boldsymbol{c}$ from $\boldsymbol{C_k}$



Final Step: Rule Generation via Frequent Itemsets

- Support (*min-sup*): used to mine the frequent itemsets
- Confidence (*min-conf*): used by the rule generation step to qualify the strength of the derived association rules
  - For each frequent itemset F, generate F's all non-empty subsets
  - For every non-empty subset s, generate a rule:

$$R: s \rightarrow (F - s)$$

  - If the rule R satisfies the minimum confidence, i.e.,

$$\text{conf}(s \rightarrow F - s) = \frac{\text{sup}(F)}{\text{sup}(s)} \geq min\_conf$$

    then R is a strong association rule and should be output.

Rule Generation: An Example

- For $F_3 = \{2,3,5\}$, there are six non-empty subsets:
  - $\{2\}$, $\{3\}$, $\{5\}$, $\{2,3\}$, $\{2,5\}$, $\{3,5\}$
- Thus, six candidate rules can be generated
  - $\{2\} \rightarrow \{3,5\}$, $\{3\} \rightarrow \{2,5\}$, $\{5\} \rightarrow \{2,5\}$
  - $\{2,3\} \rightarrow \{5\}$, $\{2,5\} \rightarrow \{3\}$, $\{3,5\} \rightarrow \{2\}$
- If any of them satisfies the minimum confidence, it will be output to the end user.

Is Apriori Fast Enough?

- Core of the Apriori algorithm:
  - Use frequent k-itemsets to generate candidate $(k+1)$-itemsets
  - Use scanning and pattern matching to calculate support for candidates

- The performance bottleneck of Apriori: **candidate generation**
  - **Huge candidate sets (exponential growth)**
    - 10^4 frequent 1-itemset will generate $> 10^7$ candidate 2-itemsets
    - To discover a frequent pattern of size 100, e.g., {a1, a2, …, a100}, one needs to generate $2^{100} \approx 10^{30}$ candidates.
  - **Multiple scans of database**
    - Needs $n+1$ scans, where n is the length of the longest pattern

*Techniques to Enhance Apriori's Efficiency

- **Shrink # candidates**
  - Hashing: A k-itemset whose hashing bucket count < threshold cannot be frequent
  - Sampling: mining on a subset of data with lower min-sup to ensure completeness
- **Reduce database scans**
  - Partitioning: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Dynamic itemset counting: Adding new candidate itemsets only when all of their subsets are estimated to be frequent
  - Transaction reduction: A transaction that does not contain any frequent k-itemset is useless in subsequent scans
- **Explore special data structures**
  - Tree projection, H-miner, Hypercube decomposition

*Partitioning: Scan Database Only Twice

- **Theorem**: Any itemset that is potentially frequent in TDB must be frequent in at least one of the partitions of TDB

Support threshold: σ
Partition: $|\mathrm{TDB}_1| + |\mathrm{TDB}_2| + \cdots + |\mathrm{TDB}_k| = |\mathrm{TDB}|$

Proof by contradiction!

TDB_1 + TDB_2 + . . . + TDB_k = TDB

$\mathrm{sup}_1(X) < \sigma|\mathrm{TDB}_1|$ · · · $\mathrm{sup}_2(X) < \sigma|\mathrm{TDB}_2|$ · · · $\mathrm{sup}_k(X) < \sigma|\mathrm{TDB}_k|$ $\mathrm{sup}(X) < \sigma|\mathrm{TDB}|$

- **Method** [A. Savasere, E. Omiecinski, S. Navathe, VLDB'95]
  - Scan 1: Partition database and find *local frequent patterns*
  - Scan 2: Consolidate *global frequent patterns*

*DHP: Direct Hashing and Pruning

- Reduce # candidates [J. Park, M. Chen, P. Yu, SIGMOD'95]
- **Observation**: A k-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
  - Frequent items: a, b, d, e
    - *minsup = 2*
  - Candidate 2-itemsets:
    - ab, ad, ae, bd, be, de
  - Hash buckets with counts
    - $\{bd\}$ is not a promising candidate 2-itemset as the count of $\{bd, be, de\}$ is below the support threshold

Tid	Items
1	a, b
2	a, b, d, e
3	a, e, f
4	c, d

Itemsets	Count
{ab, ad, ae}	3
{bd, be, de}	1

Hash Table

*ECLAT: Exploring Vertical Data Format

- A depth-first search algorithm using set intersection [Zaki et al. @KDD'97]
- ***Tid-List***: List of transaction-ids containing an itemset
  - Vertical format: $t(a) = \{T_{10}, T_{20}\}$; $t(e) = \{T_{10}, T_{20}, T_{30}\}$ → $t(ae) = \{T_{10}, T_{20}\}$
  - Deriving frequent patterns based on vertical intersections

Properties of Tid-Lists

- If X and Y always occur together in the same transactions, then $t(X) = t(Y)$. e.g., $t(ac) = t(d)$.
- If $X \subseteq Y$, then $t(Y) \subseteq t(X)$, as transactions having Y must have X. For example, $t(ce) \subseteq t(c)$.

A transaction DB in Horizontal Data Format

Tid	Itemset
10	a, c, d, e
20	a, b, e
30	b, c, e

The transaction DB in Vertical Data Format

Item	TidList
a	10, 20
b	20, 30
c	10, 30
d	10
e	10, 20, 30

- Using **diffset** to accelerate mining
  - Only keep track of differences of Tids
  - $t(e) = \{T_{10}, T_{20}, T_{30}\}$, $t(ce) = \{T_{10}, T_{30}\}$ → Diffset$(ce, e) = \{T_{20}\}$

pattern evaluation methods

WHICH PATTERNS ARE INTERESTING?

How to Evaluate if a Rule/Pattern Is Interesting?

- Pattern-mining will generate a large set of patterns/rules.
- Not all the generated patterns/rules are interesting.

- **Interestingness Measures**
  - Objective: based on statistics behind the data
    - Support, confidence, correlation, …
  - Subjective: “*one man’s trash could be other’s treasure*”
    - Query-based: Relevant to a user’s particular request (actionable)
    - Against one’s knowledge-base: *unexpected*, freshness, timeliness
    - Visualization tools: Multi-dimensional, interactive examination

Limitation of the **Support-Confidence** Framework

- Strong rules are not necessarily interesting: "$A \to B$" $[s, c]$?
- **Example**: Suppose a school may have the following statistics on # students related to playing basketball and/or eating cereal:

	Play basketball	**Not play basketball**	***sum***
Eat cereal	400	350	750
Not eat cereal	200	50	250
sum	600	400	1000 (TOTAL)

  - **Association rule mining** may generate a rule:

    play-basketball → *eat-cereal* [40%, 66.7%] ?

  - But this strong association rule is misleading → The overall % of students eating cereal is 75% > 66.7%.
  - **A more telling rule**:

    not play-basketball → *eat-cereal* [35%, 87.5%] (high s & c)

Interestingness Measure: **Lift**

- Measure of dependent / correlated events:

$$\boldsymbol{lift(B,C)} = \frac{\mathrm{P}(B \cup C)}{P(B)P(C)} = \frac{\sup(B \rightarrow C)}{\sup(B)\sup(C)} = \frac{\mathrm{conf}(B \rightarrow C)}{\sup(C)}$$

  - Tell how B and C are correlated
    - $lift(B,C) = 1$: B and C are independent
    - $lift(B,C) > 1$: positively correlated
    - $lift(B,C) < 1$: negatively correlated

	B	**Not B**	***sum***
C	400	350	750
Not C	200	50	250
sum	600	400	1000

lift is more telling than **s & c**

  - Example:

$$lift(B,C) = \frac{400/1000}{600/1000 \times 750/1000} = 0.89 \qquad lift(B,\neg C) = \frac{200/1000}{600/1000 \times 250/1000} = 1.33$$

    - Thus, B and C are negatively correlated since $lift(B,C) < 1$.
    - B and $\neg C$ are positively correlated since $lift(B,\neg C) > 1$.

Interestingness Measure: χ^2

- To test correlated events: $\chi^2 = \frac{\sum(Observed - Expected)^2}{Expected}$

  - $\chi^2 = 0$: independent
  - $\chi^2 > 0$: correlated, either positive or negative → needs additional test

	B	Not B	*sum*
C	400 (450)	350 (300)	750
Not C	200 (150)	50 (100)	250
sum	600	400	1000

Expected value

Observed value

$$\chi^2 = \frac{(400-450)^2}{450} + \frac{(350-300)^2}{300} + \frac{(200-150)^2}{150} + \frac{(50-100)^2}{100} = 55.56$$

- Thus, B and C are negatively correlated since the expected value is 450 but the observed is only 400.

- χ^2 is also more telling than the support-confidence framework

Lift and χ^2: *Are They Always Good Measures?*

- Null transactions: Transactions that contain **neither B nor C**

Examine the dataset:

- BC $(100, 0.1\%)$ is much rarer than $B\neg C$ (1000) and $\neg BC$ (1000)
- There are many $\neg B\neg C$ $(100000, 98\%)$.
- Unlikely B & C will happen together!

	B	¬B	Σ_{row}
C	100	1000	1100
¬C	1000	100000	101000
$\Sigma_{col.}$	1100	101000	102100

null transactions

- However, B and C seem to be strongly positively correlated based on:
  - $lift(B, C) = 8.44 \gg 1$
  - $\chi^2(B, C) = 670$ and Observed (100) >> Expected (11.85)
- Too many null transactions may "spoil the soup"!

Contingency table with expected values added

	B	¬B	Σ_{row}
C	100 (11.85)	1000	1100
¬C	1000 (988.15)	100000	101000
$\Sigma_{col.}$	1100	101000	102100

Interestingness Measures: Null-Invariant

- Null invariance: value does not change with # null-transactions
  - χ^2 and $lift$ are NOT null-invariant with the range of $[0, \infty]$.
- Null-invariant Measures:
  - **All Confidence**: the minimum confidence of the two association rules related to A and B, namely, "A → B" and "B → A"

$$all\_conf(A,B) = \frac{sup(A \cup B)}{max\{sup(A), sup(B)\}} = min\{P(A|B), P(B|A)\}$$

$$max\_conf(A, B) = max\{P(A|B), P(B|A)\}$$

  - **Max Confidence**: the maximum confidence of the two rules
  - **Kulczynski** (*Kulc*): an average of two confidence values
  - **Cosine**: a harmonized *lift* measure (unaffected by # total transactions)

$$Kulc(A, B) = \frac{1}{2}(P(A|B) + P(B|A))$$

$$cosine(A, B) = \frac{P(A \cup B)}{\sqrt{P(A) \times P(B)}} = \frac{sup(A \cup B)}{\sqrt{sup(A) \times sup(B)}} = \sqrt{P(A|B) \times P(B|A)}.$$

Null Invariance: An Example

- Why is null invariance crucial for the analysis of transactions?
  - Many transactions may not contain any itemsets being examined.

milk vs. coffee contingency table

	$milk$	$\neg milk$	Σ_{row}
$coffee$	mc	$\neg mc$	c
$\neg coffee$	$m\neg c$	$\neg m\neg c$	$\neg c$
Σ_{col}	m	$\neg m$	Σ

Null-transactions *w.r.t.* milk and coffee

Data Set	mc	$\overline{m}c$	$m\overline{c}$	$\overline{mc}$	χ^2	*lift*
D_1	10,000	1000	1000	100,000	90557	9.26
D_2	10,000	1000	1000	100	0	1
D_3	100	1000	1000	100,000	670	8.44
D_4	1000	1000	1000	100,000	24740	25.75
D_5	1000	100	10,000	100,000	8173	9.18
D_6	1000	10	100,000	100,000	965	1.97

Lift and χ^2 are not null-invariant – not good to evaluate data that contain either too many or too few null transactions!

Comparison of Null-Invariant Measures

- Not all null-invariant measures are created equal.
- Which one is better?

- **Kulc** (Kulczynski 1927) holds firm and is in balance of both directional implications.

milk vs. coffee contingency table

	$milk$	$\neg milk$	Σ_{row}
$coffee$	mc	$\neg mc$	c
$\neg coffee$	$m\neg c$	$\neg m \neg c$	$\neg c$
Σ_{col}	m	$\neg m$	Σ

Data Set	mc	$\overline{m}c$	$m\overline{c}$	$\overline{mc}$	all_conf.	max_conf.	Kulc.	cosine
D_1	10,000	1000	1000	100,000	0.91	0.91	0.91	0.91
D_2	10,000	1000	1000	100	0.91	0.91	0.91	0.91
D_3	100	1000	1000	100,000	0.09	0.09	0.09	0.09
D_4	1000	1000	1000	100,000	0.5	0.5	0.5	0.5
D_5	1000	100	10,000	100,000	0.09	0.91	0.5	0.29
D_6	1000	10	100,000	100,000	0.01	0.99	0.5	0.10

Subtle: They disagree on those cases

Summary

- The discovery of frequent patterns, associations, and correlation relationships is useful in many applications.
  - Customers' buying habits: itemsets that are frequently bought together
- **Association rule mining**: 1) frequent *k*-itemsets ($A \cup B$, *min_sup*); 2) generating strong association ($A \rightarrow B$, *min_conf*).
  - Apriori: any subset of a frequent itemset must be frequent!
  - Efficiency bottleneck: reduce # candidates or DB scans
- Not all strong association rules are interesting.
  - The support–confidence framework *vs* other interestingness measures
  - A measure is null-invariant if its value is free from the influence of null-transactions (that do not contain any itemsets being examined).

Email: fengmei.jin@polyu.edu.hk

Office: PQ747

THANK YOU!