CS 521
Data Mining Techniques
Instructor: Abdullah Mueen

LECTURE 4: FREQUENT PATTERN MINING
What Is Frequent Pattern Analysis?

**Frequent pattern**: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set

First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining

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.
Why Is Freq. Pattern Mining Important?

Freq. pattern: An intrinsic and important property of datasets

Foundation for many essential data mining tasks
- Association, correlation, and causality analysis
- Sequential, structural (e.g., sub-graph) patterns
- Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
- Classification: discriminative, frequent pattern analysis
- Cluster analysis: frequent pattern-based clustering
- Data warehousing: iceberg cube and cube-gradient
- Semantic data compression: fascicles
- Broad applications
Basic Concepts: Frequent Patterns

**itemset**: A set of one or more items

**k-itemset** $X = \{x_1, \ldots, x_k\}$

*(absolute) support*, or, *support count* of $X$: Frequency or occurrence of an itemset $X$

*(relative) support*, $s$, is the fraction of transactions that contains $X$ (i.e., the probability that a transaction contains $X$)

An itemset $X$ is *frequent* if $X$’s support is no less than a $\text{minsup}$ threshold
Basic Concepts: Association Rules

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 $\text{minsup} = 50\%$, $\text{minconf} = 50\%$

Freq. Pat.: Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3

- Association rules: (many more!)
  - *Beer* $\rightarrow$ *Diaper* (60%, 100%)
  - *Diaper* $\rightarrow$ *Beer* (60%, 75%)
Closed Patterns and Max-Patterns

A long pattern contains a combinatorial number of sub-patterns, e.g., \{a_1, \ldots, a_{100}\} contains \binom{100}{1} + \binom{100}{2} + \ldots + \binom{1}{100} = 2^{100} - 1 = 1.27 \times 10^{30} sub-patterns!

Solution: Mine closed patterns and max-patterns instead

An itemset \(X\) is closed if \(X\) is frequent and there exists no super-pattern \(Y \supset X\), with the same support as \(X\) (proposed by Pasquier, et al. @ ICDT’99)

An itemset \(X\) is a max-pattern if \(X\) is frequent and there exists no frequent super-pattern \(Y \supset X\) (proposed by Bayardo @ SIGMOD’98)

Closed pattern is a lossless compression of freq. patterns

- Reducing the # of patterns and rules
Closed Patterns and Max-Patterns

Exercise: Suppose a DB contains only two transactions

- \( \langle a_1, \ldots, a_{100} \rangle, \langle a_1, \ldots, a_{50} \rangle \)
- Let \( \text{min}\_\text{sup} = 1 \)

What is the set of closed itemset?

- \( \{a_1, \ldots, a_{100}\} \): 1
- \( \{a_1, \ldots, a_{50}\} \): 2

What is the set of max-pattern?

- \( \{a_1, \ldots, a_{100}\} \): 1

What is the set of all patterns?

- \( \{a_1\}: 2, \ldots, \{a_1, a_2\}: 2, \ldots, \{a_1, a_{51}\}: 1, \ldots, \{a_1, a_2, \ldots, a_{100}\}: 1 \)
- A big number: \( 2^{100} - 1 \)? Why?
Chapter 5: Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

Basic Concepts

Frequent Itemset Mining Methods

Which Patterns Are Interesting?—Pattern Evaluation Methods

Summary
Scalable Frequent Itemset Mining Methods

Apriori: A Candidate Generation-and-Test Approach

Improving the Efficiency of Apriori

FPGrowth: A Frequent Pattern-Growth Approach

ECLAT: Frequent Pattern Mining with Vertical Data Format
The Downward Closure Property and Scalable Mining Methods

The **downward closure** property of frequent patterns

- **Any subset of a frequent itemset must be frequent**
- If \{beer, diaper, nuts\} is frequent, so is \{beer, diaper\}
- i.e., every transaction having \{beer, diaper, nuts\} also contains \{beer, diaper\}

Scalable mining methods: Three major approaches

- Apriori (Agrawal & Srikant@VLDB’94)
- Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD’00)
- Vertical data format approach (Charm—Zaki & Hsiao @SDM’02)
Apriori: A Candidate Generation & Test Approach

**Apriori pruning principle:** If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB’94, Mannila, et al. @ KDD’94)

**Method:**
- Initially, scan DB once to get frequent 1-itemset
- **Generate** length (k+1) candidate itemsets from length k frequent itemsets
- **Test** the candidates against DB
- Terminate when no frequent or candidate set can be generated
The Apriori Algorithm—An Example

Database TDB

<table>
<thead>
<tr>
<th>Tid</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A, C, D</td>
</tr>
<tr>
<td>20</td>
<td>B, C, E</td>
</tr>
<tr>
<td>30</td>
<td>A, B, C, E</td>
</tr>
<tr>
<td>40</td>
<td>B, E</td>
</tr>
</tbody>
</table>

Sup\text{min} = 2

\(C_1\)

1\text{st scan}

\(C_1\)

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A}</td>
<td>2</td>
</tr>
<tr>
<td>{B}</td>
<td>3</td>
</tr>
<tr>
<td>{C}</td>
<td>3</td>
</tr>
<tr>
<td>{D}</td>
<td>1</td>
</tr>
<tr>
<td>{E}</td>
<td>3</td>
</tr>
</tbody>
</table>

\(L_1\)

\(C_2\)

2\text{nd scan}

\(C_2\)

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A, B}</td>
<td>1</td>
</tr>
<tr>
<td>{A, C}</td>
<td>2</td>
</tr>
<tr>
<td>{A, E}</td>
<td>1</td>
</tr>
<tr>
<td>{B, C}</td>
<td>2</td>
</tr>
<tr>
<td>{B, E}</td>
<td>3</td>
</tr>
<tr>
<td>{C, E}</td>
<td>2</td>
</tr>
</tbody>
</table>

\(L_2\)

\(C_3\)

3\text{rd scan}

\(C_3\)

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{B, C, E}</td>
<td>2</td>
</tr>
</tbody>
</table>

\(L_3\)
The Apriori Algorithm (Pseudo-Code)

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

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

How to generate candidates?
- Step 1: self-joining $L_k$
- Step 2: pruning

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 $L_3$
- $C_4 = \{abcd\}$
Further Improvement of the Apriori Method

Major computational challenges
- Multiple scans of transaction database
- Huge number of candidates
- Tedious workload of support counting for candidates

Improving Apriori: general ideas
- Reduce passes of transaction database scans
- Shrink number of candidates
- Facilitate support counting of candidates
Partition: Scan Database Only Twice

Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB

- Scan 1: partition database and find local frequent patterns
- Scan 2: consolidate global frequent patterns

A. Savasere, E. Omiecinski and S. Navathe, *VLDB’95*
DHP: Reduce the Number of Candidates

A $k$-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent

- Candidates: $a, b, c, d, e$
- Hash entries
  - $\{ab, ad, ae\}$
  - $\{bd, be, de\}$
  - ...
- Frequent 1-itemset: $a, b, d, e$
- $ab$ is not a candidate 2-itemset if the sum of count of $\{ab, ad, ae\}$ is below support threshold


<table>
<thead>
<tr>
<th>count</th>
<th>itemsets</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>${ab, ad, ae}$</td>
</tr>
<tr>
<td>88</td>
<td>${bd, be, de}$</td>
</tr>
<tr>
<td>102</td>
<td>${yz, qs, wt}$</td>
</tr>
</tbody>
</table>
Sampling for Frequent Patterns

Select a sample of original database, mine frequent patterns within sample using Apriori

Scan database once to verify frequent itemsets found in sample, only borders of closure of frequent patterns are checked

- Example: check abcd instead of ab, ac, ..., etc.

Scan database again to find missed frequent patterns

H. Toivonen. Sampling large databases for association rules. In VLDB’96
DIC: Reduce Number of Scans

Once both A and D are determined frequent, the counting of AD begins

Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins

Transactions

1-itemsets

2-itemsets

...
Pattern-Growth Approach: Mining Frequent Patterns Without Candidate Generation

Bottlenecks of the Apriori approach
- Breadth-first (i.e., level-wise) search
- Candidate generation and test
  - Often generates a huge number of candidates

The FP-Growth Approach (J. Han, J. Pei, and Y. Yin, SIGMOD’ 00)
- Depth-first search
- Avoid explicit candidate generation

Major philosophy: Grow long patterns from short ones using local frequent items only
- “abc” is a frequent pattern
- Get all transactions having “abc”, i.e., project DB on abc: DB|abc
- “d” is a local frequent item in DB|abc \(\rightarrow\) abcd is a frequent pattern
Construct FP-tree from a Transaction Database

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

---

**Header Table**

<table>
<thead>
<tr>
<th>Item</th>
<th>Frequency</th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

**min_support = 3**

**TID** | **Items bought** | **(ordered) frequent items**
---|------------------|---------------------|
100 | \{f, a, c, d, g, i, m, p\} | \{f, c, a, m, p\} |
200 | \{a, b, c, f, l, m, o\} | \{f, c, a, b, m\} |
300 | \{b, f, h, j, o, w\} | \{f, b\} |
400 | \{b, c, k, s, p\} | \{c, b, p\} |
500 | \{a, f, c, e, l, p, m, n\} | \{f, c, a, m, p\} |

F-list = f-c-a-b-m-p
Partition Patterns and Databases

Frequent patterns can be partitioned into subsets according to f-list

- $F$-list = f-c-a-b-m-p
- Patterns containing p
- Patterns having m but no p
- ...
- Patterns having c but no a nor b, m, p
- Pattern f

Completeness and non-redundency
Find Patterns Having P From P-conditional Database

Starting at the frequent item header table in the FP-tree

 Traverse the FP-tree by following the link of each frequent item p

 Accumulate all of transformed prefix paths of item p to form p’s conditional pattern base
From Conditional Pattern-bases to Conditional FP-trees

For each pattern-base
- Accumulate the count for each item in the base
- Construct the FP-tree for the frequent items of the pattern base

<table>
<thead>
<tr>
<th>Item</th>
<th>frequency</th>
<th>head</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{m-conditional pattern base: } \{ fca:2, fcab:1 \} \]

\[ \text{All frequent patterns relate to } m \]
\[ \{ \}
\[ m, \]
\[ \{ f:3 \}
\[ fm, cm, am, \]
\[ \{ c:3 \}
\[ fc, fam, cam, \]
\[ \{ a:3 \}
\[ fcam \] \]

\[ \text{m-conditional FP-tree} \]

Header Table
Recursion: Mining Each Conditional FP-tree

{}  
| {}  
| f:3  
| c:3  
| a:3

$m$-conditional FP-tree

Cond. pattern base of “am”: (fc:3)

f:3  
| c:3  

$am$-conditional FP-tree

Cond. pattern base of “cm”: (f:3)

f:3  

$cm$-conditional FP-tree

Cond. pattern base of “cam”: (f:3)

f:3  

$cam$-conditional FP-tree
A Special Case: Single Prefix Path in FP-tree

Suppose a (conditional) FP-tree $T$ has a shared single prefix-path $P$

$\{\}$

Mining can be decomposed into two parts

- Reduction of the single prefix path into one node
- Concatenation of the mining results of the two parts

$r_1 = \{\} + b_1 : m_1 + a_1 : n_1 + a_2 : n_2 + a_3 : n_3 + C_1 : k_1 + C_2 : k_2 + C_3 : k_3$
Benefits of the FP-tree Structure

Completeness

- Preserve complete information for frequent pattern mining
- Never break a long pattern of any transaction

Compactness

- Reduce irrelevant info—ininfrequent items are gone
- Items in frequency descending order: the more frequently occurring, the more likely to be shared
- Never be larger than the original database (not count node-links and the count field)
The Frequent Pattern Growth Mining Method

Idea: Frequent pattern growth
- Recursively grow frequent patterns by pattern and database partition

Method
- For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
- Repeat the process on each newly created conditional FP-tree
- Until the resulting FP-tree is empty, or it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern
Advantages of the Pattern Growth Approach

Divide-and-conquer:
- Decompose both the mining task and DB according to the frequent patterns obtained so far
- Lead to focused search of smaller databases

Other factors
- No candidate generation, no candidate test
- Compressed database: FP-tree structure
- No repeated scan of entire database
- Basic ops: counting local freq items and building sub FP-tree, no pattern search and matching

A good open-source implementation and refinement of FP-Growth
- FP-Growth+ (Grahne and J. Zhu, FIMI'03)
Mining Frequent Closed Patterns: CLOSET

Flist: list of all frequent items in support ascending order
- Flist: d-a-f-e-c

Divide search space
- Patterns having d
- Patterns having d but no a, etc.

Find frequent closed pattern recursively
- Every transaction having d also has $cfa \rightarrow cfad$ is a frequent closed pattern

J. Pei, J. Han & R. Mao. “CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets”, DMKD'00.
CLOSET+: Mining Closed Itemsets by Pattern-Growth

Itemset merging: if $Y$ appears in every occurrence of $X$, then $Y$ is merged with $X$

Sub-itemset pruning: if $Y \subseteq X$, and $\text{sup}(X) = \text{sup}(Y)$, $X$ and all of $X$’s descendants in the set enumeration tree can be pruned

Hybrid tree projection
  ◦ Bottom-up physical tree-projection
  ◦ Top-down pseudo tree-projection

Item skipping: if a local frequent item has the same support in several header tables at different levels, one can prune it from the header table at higher levels

Efficient subset checking
MaxMiner: Mining Max-Patterns

1st scan: find frequent items
- A, B, C, D, E

2nd scan: find support for
- AB, AC, AD, AE, ABCDE
- BC, BD, BE, BCDE
- CD, CE, CDE, DE

Since BCDE is a max-pattern, no need to check BCD, BDE, CDE in later scan

R. Bayardo. Efficiently mining long patterns from databases. *SIGMOD’98*
Computational Complexity of Frequent Itemset Mining

How many itemsets are potentially to be generated in the worst case?

- The number of frequent itemsets to be generated is sensitive to the minsup threshold
- When minsup is low, there exist potentially an exponential number of frequent itemsets
- The worst case: \( M^N \) where \( M \): # distinct items, and \( N \): max length of transactions

The worst case complexity vs. the expected probability

- Ex. Suppose Walmart has \( 10^4 \) kinds of products
  - The chance to pick up one product \( 10^{-4} \)
  - The chance to pick up a particular set of 10 products: \( \sim 10^{-40} \)
  - What is the chance this particular set of 10 products to be frequent \( 10^3 \) times in \( 10^9 \) transactions?
Interestingness Measure: Correlations (Lift)

play basketball ⇒ eat cereal [40%, 66.7%] is misleading

- The overall % of students eating cereal is 75% > 66.7%.

play basketball ⇒ not eat cereal [20%, 33.3%] is more accurate, although with lower support and confidence

Measure of dependent/correlated events: lift

\[
\text{lift}(B, \neg C) = \frac{1000/5000}{3000/5000 \cdot 1250/5000} = 1.33
\]

\[
\text{lift}(B, C) = \frac{2000/5000}{3000/5000 \cdot 3750/5000} = 0.89
\]

\[
\text{lift} = \frac{P(A \cup B)}{P(A)P(B)}
\]

<table>
<thead>
<tr>
<th></th>
<th>Basketball</th>
<th>Not basketball</th>
<th>Sum (row)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cereal</td>
<td>2000</td>
<td>1750</td>
<td>3750</td>
</tr>
<tr>
<td>Not cereal</td>
<td>1000</td>
<td>250</td>
<td>1250</td>
</tr>
<tr>
<td>Sum(col.)</td>
<td>3000</td>
<td>2000</td>
<td>5000</td>
</tr>
</tbody>
</table>
“Buy walnuts ⇒ buy milk [1%, 80%]” is misleading if 85% of customers buy milk.

Support and confidence are not good to indicate correlations.

Over 20 interestingness measures have been proposed (see Tan, Kumar, Sritastava @KDD’02).

Which are good ones?

<table>
<thead>
<tr>
<th>symbol</th>
<th>measure</th>
<th>range</th>
<th>formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>φ</td>
<td>φ-coefficient</td>
<td>-1...1</td>
<td>$P(A, B) - P(A)P(B)$</td>
</tr>
<tr>
<td>Q</td>
<td>Yule’s Q</td>
<td>-1...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>Y</td>
<td>Yule’s Y</td>
<td>-1...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>k</td>
<td>Cohens’s</td>
<td>-1...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>PS</td>
<td>Piatetsky-Shapiro’s</td>
<td>-0.25...0.25</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>F</td>
<td>Certainty factor</td>
<td>-1...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>AV</td>
<td>added value</td>
<td>-0.5...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>K</td>
<td>Klosgen’s Q</td>
<td>-0.33...0.38</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>g</td>
<td>Goodman-Kruskal’s</td>
<td>0...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>M</td>
<td>Mutual Information</td>
<td>0...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>J</td>
<td>J-Measure</td>
<td>0...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>G</td>
<td>Gini index</td>
<td>0...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>s</td>
<td>support</td>
<td>0...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>c</td>
<td>confidence</td>
<td>0...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>L</td>
<td>Laplace</td>
<td>0...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>IS</td>
<td>Cosine</td>
<td>0...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>γ</td>
<td>coherence(Jaccard)</td>
<td>0...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>α</td>
<td>all-confidence</td>
<td>0...1</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>o</td>
<td>odds ratio</td>
<td>0...∞</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>V</td>
<td>Conviction</td>
<td>0.5...∞</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>λ</td>
<td>lift</td>
<td>0...∞</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>S</td>
<td>Collective strength</td>
<td>0...∞</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
<tr>
<td>$\chi^2$</td>
<td>$\chi^2$</td>
<td>0...∞</td>
<td>$\frac{P(A, B) - P(A)P(B)}{P(A)P(B) - P(A)B}$</td>
</tr>
</tbody>
</table>
Summary

Basic concepts: association rules, support-confident framework, closed and max-patterns

Scalable frequent pattern mining methods
- Apriori (Candidate generation & test)
- Projection-based (FPgrowth, CLOSET+, ...)

- Which patterns are interesting?
  - Pattern evaluation methods