Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Mining Diverse Patterns
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Sequential Pattern Mining
- Graph Pattern Mining
- Summary
Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Mining Diverse Patterns
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Sequential Pattern Mining
- Graph Pattern Mining
- Summary
Mining Diverse Patterns

- Mining Multiple-Level Associations
- Mining Multi-Dimensional Associations
- Mining Quantitative Associations
- Mining Negative Correlations
- Mining Compressed and Redundancy-Aware Patterns

Mining Multiple-Level Frequent Patterns

- Items often form hierarchies
  - Ex.: Reduced Fat milk; Wonder wheat bread
- How to set min-support thresholds?
  - Uniform min-support across multiple levels (reasonable?)
  - Level-reduced min-support: Items at the lower level are expected to have lower support
- Efficient mining: Shared multi-level mining (How?)
Mining Multiple-Level Frequent Patterns

- Items often form hierarchies
  - Ex.: Reduced Fat milk; Wonder wheat bread
- How to set min-support thresholds?
  - Uniform min-support across multiple levels (reasonable?)
  - Level-reduced min-support: Items at the lower level are expected to have lower support
- Efficient mining: *Shared* multi-level mining
  - Use the lowest min-support to pass down the set of candidates

Reduced support

<table>
<thead>
<tr>
<th>Level 1</th>
<th>Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_sup = 5%</td>
<td>min_sup = 5%</td>
</tr>
<tr>
<td>Milk [support = 10%]</td>
<td>Reduced Fat Milk [support = 6%]</td>
</tr>
<tr>
<td>Reduced Fat Milk [support = 6%]</td>
<td>Skim Milk [support = 2%]</td>
</tr>
<tr>
<td>Skim Milk [support = 2%]</td>
<td></td>
</tr>
</tbody>
</table>

Redundancy Filtering at Mining Multi-Level Associations

Multi-level association mining may generate many redundant rules

- *Redundancy filtering*: Some rules may be redundant due to “ancestor” relationships between items
  - (Suppose the *Reduced Fat milk* sold in about ¼ of milk sold in gallons)
- milk \(\Rightarrow\) wheat bread [support = 8%, confidence = 70%] \hspace{1cm} (1)
- Reduced fat milk \(\Rightarrow\) wheat bread [ what do you expect ? ] \hspace{1cm} (2)
Redundancy Filtering at Mining Multi-Level Associations

Multi-level association mining may generate many redundant rules.

- **Redundancy filtering**: Some rules may be redundant due to “ancestor” relationships between items.
  - (Suppose the reduced fat milk sold in about ¼ of milk sold in gallons)
  - milk ⇒ wheat bread [support = 8%, confidence = 70%] (1)
  - Reduced fat milk ⇒ wheat bread [support = 2%, confidence = 72%] (2)

- A rule is redundant if its support is close to the “expected” value, according to its “ancestor” rule, and it has a similar confidence as its “ancestor”.
  - Rule (1) is an ancestor of rule (2), which one to prune?

Customized Min-Supports for Different Kinds of Items

Until now: the same min-support threshold for all the items or item sets to be mined in each association mining.

- But, some items (e.g., diamond, watch, ...) are valuable but less frequent
- Necessary to have customized min-support settings for different kinds of items
- One Method: Use group-based “individualized” min-support
  - E.g., valuable group {diamond, watch}: 0.05%; whereas {bread, milk}: 5%; ...
  - How to mine such rules efficiently?
  - Existing scalable mining algorithms can be easily extended to cover such cases
Mining Multi-Dimensional Associations

- **Single-dimensional rules** (e.g., items are all in “product” dimension)
  - buys(X, “milk”) ⇒ buys(X, “bread”)

- **Multi-dimensional rules** (i.e., items in ≥ 2 dimensions or predicates)
  - **Inter-dimension association rules** (*no repeated predicates*)
  - **Hybrid-dimension association rules** (*repeated predicates*)

Attributes can be **categorical or numerical**

- Categorical Attributes (e.g., profession, product: *no ordering* among values):
  - Data cube for inter-dimension association

- Quantitative Attributes: Numeric, *implicit ordering* among values—
  - *discretization (binning)*, clustering, and gradient approaches
Mining Quantitative Associations

- Mining associations with numerical attributes
  - Ex.: Numerical attributes: age and salary

Methods

- Static discretization based on predefined concept hierarchies
- Data cube-based aggregation
- Dynamic discretization based on data distribution
- Clustering: Distance-based association
  - First one-dimensional clustering, then association

Deviation analysis:
- Gender = female $\Rightarrow$ Wage: mean=$7/\text{hr}$ (whereas overall mean = $9$

Mining Extraordinary Phenomena in Quantitative Association Mining

- Mining extraordinary (i.e., interesting) phenomena
  - Ex.: Gender = female $\Rightarrow$ Wage: mean=$7/\text{hr}$ (overall mean = $9$

LHS: a subset of the population: Gender = female

RHS: an extraordinary behavior of this subset: Wage: mean=$7/\text{hr}$

The rule is accepted only if a statistical test (e.g., Z-test) confirms the inference with high confidence

Subrule: Highlights the extraordinary behavior of a subset of the population of the super rule
  - Ex.: (Gender = female) $\land$ (South = yes) $\Rightarrow$ mean wage = $6.3/\text{hr}$

Rule condition can be categorical or numerical (quantitative rules)

- Ex.: Education in [14-18] (yrs) $\Rightarrow$ mean wage = $11.64/\text{hr}$

Efficient methods have been developed for mining such extraordinary rules (e.g., Aumann and Lindell@KDD'99)
Rare Patterns vs. Negative Patterns

- **Rare patterns**
  - Very low support but interesting (e.g., buying Rolex watches)
  - How to mine them? Setting individualized, group-based min-support thresholds for different groups of items (similar to valuable items)

- **Negative patterns**
  - Negatively correlated: Unlikely to happen together
  - Ex.: Since it is unlikely that the same customer buys both a Ford Expedition (an SUV car) and a Ford Fusion (a hybrid car), buying a Ford Expedition and buying a Ford Fusion are likely negatively correlated patterns

How to define negative patterns?

Defining Negative Correlated Patterns

- A support-based definition of Negative Correlated Patterns
  - If itemsets A and B are both frequent but rarely occur together, i.e., $\text{sup}(A \cup B) \ll \text{sup}(A) \times \text{sup}(B)$
  - Then A and B are negatively correlated
  - Is this a good definition for large transaction datasets?
Defining Negative Correlated Patterns

- A support-based definition of Negative Correlated Patterns
  - If itemsets A and B are both frequent but rarely occur together, i.e., \( \text{sup}(A \cup B) << \text{sup}(A) \times \text{sup}(B) \)
  - Then A and B are negatively correlated
  - Is this a good definition for large transaction datasets?

Ex.: Suppose a store sold two needle packages A and B 100 times each, but only one transaction contained both A and B

- When there are in total 200 transactions, we have
  - \( s(A \cup B) = 0.005 \), \( s(A) \times s(B) = 0.25 \), \( s(A \cup B) << s(A) \times s(B) \)
  - But when there are \( 10^5 \) transactions, we have
    - \( s(A \cup B) = 1/10^5 \), \( s(A) \times s(B) = 1/10^3 \times 1/10^3 \), \( s(A \cup B) > s(A) \times s(B) \)

PROBLEM: Null transactions: The support-based definition is not null-invariant!
Defining Negative Correlation: Need Null-Invariance in Definition

- A good definition on negative correlation should take care of the null-invariance problem
- Whether two itemsets A and B are negatively correlated should not be influenced by the number of null-transactions

A Kulczynski measure-based definition
- If itemsets A and B are frequent but \( \frac{(P(A|B) + P(B|A))/2}{\epsilon} < \epsilon \), where \( \epsilon \) is a negative pattern threshold, then A and B are negatively correlated

For the same needle package problem:
- No matter if there are in total 200 or \( 10^5 \) transactions
- If \( \epsilon = 0.01 \), we have \( \frac{(P(A|B) + P(B|A))/2}{(0.01 + 0.01)/2} < \epsilon \)
Mining Compressed Patterns

- Why mining compressed patterns?
  - Too many scattered patterns but not so meaningful
- Pattern distance measure
  \[ \text{Dist}(P_1, P_2) = 1 - \frac{|T(P_1) \cap T(P_2)|}{|T(P_1) \cup T(P_2)|} \]
- \( \delta \)-clustering: For each pattern \( P \), find all patterns which can be expressed by \( P \) and whose distance to \( P \) is within \( \delta \) (\( \delta \)-cover)
- All patterns in the cluster can be represented by \( P \)
- Method for efficient, direct mining of compressed frequent patterns (e.g., D. Xin, J. Han, X. Yan, H. Cheng, "On Compressing Frequent Patterns", Knowledge and Data Engineering, 60:5-29, 2007)

<table>
<thead>
<tr>
<th>Pat-ID</th>
<th>Item-Sets</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>{38,16,18,12}</td>
<td>205227</td>
</tr>
<tr>
<td>P2</td>
<td>{38,16,18,12,17}</td>
<td>205211</td>
</tr>
<tr>
<td>P3</td>
<td>{39,38,16,18,12,17}</td>
<td>101758</td>
</tr>
<tr>
<td>P4</td>
<td>{39,16,18,12,17}</td>
<td>161563</td>
</tr>
<tr>
<td>P5</td>
<td>{39,16,18,12}</td>
<td>161576</td>
</tr>
</tbody>
</table>

- Closed patterns
  - P1, P2, P3, P4, P5
  - Emphasizes too much on support
  - There is no compression
- Max-patterns
  - P3: information loss
- Desired output (a good balance):
  - P2, P3, P4

Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Mining Diverse Patterns
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Sequential Pattern Mining
- Graph Pattern Mining
- Summary
Why Constraint-Based Mining?

- Finding all the patterns in a dataset autonomously? — unrealistic!
- Too many patterns but not necessarily user-interested!
- Pattern mining should be an interactive process
- User directs what to be mined using a data mining query language (or a graphical user interface)

- Constraint-based mining
  - User flexibility: provides constraints on what to be mined
  - Optimization: explores such constraints for efficient mining
  - **Constraint-based mining**: Constraint-pushing, similar to push selection first in DB query processing

Constraints in General Data Mining

A data mining query can be in the form of a meta-rule or with the following language primitives

- **Knowledge type constraint**:
  - Ex.: classification, association, clustering, outlier finding, ....
- **Data constraint** — using SQL-like queries
  - Ex.: find products sold together in NY stores this year
- **Dimension/level constraint**
  - Ex.: in relevance to region, price, brand, customer category
- **Rule (or pattern) constraint**
  - Ex.: small sales (price < $10) triggers big sales (sum > $200)
- **Interestingness constraint**
  - Ex.: strong rules: min_sup ≥ 0.02, min_conf ≥ 0.6, min_correlation ≥ 0.7
Meta-Rule Guided Mining

- A meta-rule can contain partially instantiated predicates & constants
  - \( P_1(X, Y) \land P_2(X, W) \Rightarrow \text{buys}(X, "iPad") \)
- The resulting mined rule can be
  - \( \text{age}(X, "15-25") \land \text{profession}(X, "student") \Rightarrow \text{buys}(X, "iPad") \)
- In general, (meta) rules can be in the form of
  - \( P_1 \land P_2 \land \ldots \land P_l \Rightarrow Q_1 \land Q_2 \land \ldots \land Q_r \)

- Method to find meta-rules
  - Find frequent \((l + r)\) predicates (based on min-support)
  - Push constants deeply when possible into the mining process
  - Also, push min_conf, min_correlation, and other measures as early as possible (measures acting as constraints)

Different Kinds of Constraints Lead to Different Pruning Strategies

- Constraints can be categorized as
  - Pattern space pruning constraints vs. data space pruning constraints
  - Pattern space pruning constraints
  - Anti-monotonic: If constraint \( c \) is violated (by pattern of items), its further mining can be terminated
  - Monotonic: If constraint \( c \) is satisfied, no need to check \( c \) again
  - Succinct: if the constraint \( c \) can be enforced by directly manipulating the data
  - Convertible: constraint \( c \) can be converted to monotonic or anti-monotonic if items can be properly ordered in processing
Different Kinds of Constraints Lead to Different Pruning Strategies

- Constraints can be categorized as:
  - Pattern space pruning constraints vs. data space pruning constraints
  - Pattern space pruning constraints
    - Anti-monotonic: If constraint $c$ is violated (by pattern of items), its further mining can be terminated
    - Monotonic: If constraint $c$ is satisfied, no need to check $c$ again
    - Succinct: if the constraint $c$ can be enforced by directly manipulating the data
    - Convertible: constraint $c$ can be converted to monotonic or anti-monotonic if items can be properly ordered in processing

- Data space pruning constraints
  - Data succinct: Data space can be pruned at the initial pattern mining process
  - Data anti-monotonic: If a transaction $t$ (data) does not satisfy constraint $c$, then $t$ can be pruned to reduce data processing effort

Pattern Space Pruning with Pattern Anti-Monotonicity

- Constraint $c$ is anti-monotone
  - If an itemset $S$ violates constraint $c$, so does any of its superset
  - That is, mining on itemset $S$ can be terminated
  - Ex. 1: $c_1$: $\text{sum}(S,\text{price}) \leq v$ is anti-monotone
  - Ex. 2: $c_2$: $\text{range}(S,\text{profit}) \leq 15$ is anti-monotone
  - Itemset $ab$ violates $c_2$ (range(ab) = 40)
  - So does every superset of $ab$
  - Ex. 3. $c_3$: $\text{sum}(S,\text{Price}) \geq v$ is not anti-monotone
  - Ex. 4. Is $c_4$: $\text{support}(S) \geq \sigma$ anti-monotone?
    - Yes! Apriori pruning is essentially pruning with an anti-monotonic constraint!
Pattern Monotonicity and Its Roles

- A constraint $c$ is monotone: if an itemset $S$ satisfies the constraint $c$, then so does any of its superset.
- That is, we do not need to check $c$ in subsequent mining.
- Ex. 1: $c_1$: $\text{sum}(S.Price) \geq v$ is monotone.
- Ex. 2: $c_2$: $\text{min}(S.Price) \leq v$ is monotone.
- Ex. 3: $c_3$: $\text{range}(S.profit) \geq 15$ is monotone.

- Itemset $ab$ satisfies $c_3$.
- So does every superset of $ab$.

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
<th>Item</th>
<th>Price</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, b, c, d, f, h</td>
<td>a</td>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>20</td>
<td>b, c, d, f, g, h</td>
<td>b</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>b, c, d, f, g</td>
<td>c</td>
<td>150</td>
<td>-20</td>
</tr>
<tr>
<td>40</td>
<td>a, c, e, f, g</td>
<td>d</td>
<td>35</td>
<td>-15</td>
</tr>
</tbody>
</table>

**Data Space Pruning with Data Anti-Monotonicity**

- A constraint $c$ is data anti-monotone: In the mining process, if a data entry $t$ cannot satisfy a pattern $p$ under $c$, $t$ cannot satisfy $p$'s superset either.
- Data space pruning: Data entry $t$ can be pruned.
- Ex. 1: $c_1$: $\text{sum}(S.Profit) \geq v$ is data anti-monotone.
- Let constraint $c_1$ be: $\text{sum}(S.Profit) \geq 25$.
- Data entry $T_{30}$: $\{b, c, d, f, g\}$ can be removed since none of their combinations can make an $S$ whose sum of the profit is $\geq 25$.
- Ex. 2: $c_2$: $\text{min}(S.Price) \leq v$ is data anti-monotone.
- Consider $v = 5$ but every item in transaction $T_{50}$ has a price higher than 10.
- Ex. 3: $c_3$: $\text{range}(S.Profit) \geq 25$ is data anti-monotone.
Data Space Pruning Should Be Explored Recursively

- Example. \( c_3: \text{range}(S.\text{Profit}) > 25 \)
  - We check b's projected database
  - But item “a” is infrequent (sup = 1)
  - After removing “a (40)” from \( T_{10} \)
  - \( T_{10} \) cannot satisfy \( c_3 \) any more
    - since “b (0)” and “c (-20), d (-15), f (-10), h (5)”
  - By removing \( T_{10} \), we can also prune “h” in \( T_{20} \)

- Note: \( c_3 \) prunes \( T_{10} \) effectively only after “a” is pruned (by min-sup) in b’s projected DB

---

Succinctness: Pruning Both Data and Pattern Spaces

- Succinctness: if the constraint \( c \) can be enforced by directly manipulating the data
- Ex. 1: To find those patterns without item \( i \)
  - Remove \( i \) from DB and then mine (pattern space pruning)
- Ex. 2: To find those patterns containing item \( i \)
  - Mine only \( i \)-projected DB (data space pruning)
- Ex. 3: \( c_3: \text{min}(S.\text{Price}) \leq v \) is succinct
  - Start with only items whose \( \text{price} \leq v \) (pattern space pruning) and remove transactions that only have high-price items (data space pruning)
- Ex. 4: \( c_4: \text{sum}(S.\text{Price}) \geq v \) is not succinct
  - Satisfying the constraint cannot be determined beforehand since sum of the price of itemset \( S \) keeps increasing
Convertible Constraints: Ordering Data in Transactions

- Convert tough constraints into (anti-)monotone by proper ordering of items in transactions
  - Examine $c_1$: $\text{avg}(S.\text{profit}) > 20$
    - Order items in value-descending order
    - $<a, g, f, h, b, d, c, e>$
    - An itemset $ab$ violates $c_1$ ($\text{avg}(ab) = 20$)
    - So does $ab^*$ (i.e., $ab$-projected DB)
    - $C_1$: anti-monotone if patterns grow in the right order!
  - Can item-reordering work for Apriori?
    - Does not work for level-wise candidate generation!
    - $\text{avg}(agf) = 23.3 > 20$, but $\text{avg}(gf) = 15 < 20$, hence $gf$ would have been incorrectly pruned

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a, b, c, d, f, h</td>
</tr>
<tr>
<td>20</td>
<td>b, c, d, f, g, h</td>
</tr>
<tr>
<td>30</td>
<td>b, c, d, f, g</td>
</tr>
<tr>
<td>40</td>
<td>a, c, e, f, g</td>
</tr>
</tbody>
</table>

How to Handle Multiple Constraints?

- It is beneficial to use multiple constraints in pattern mining
- But different constraints may require potentially conflicting item-ordering
  - If there exists an order $R$ making both $c_1$ and $c_2$ convertible, try to sort items in the order that benefits pruning most
  - If there exists conflicting orderings between $c_1$ and $c_2$
    - Try to sort data and enforce one constraint first (which one?)
    - Then enforce the other when mining the projected databases
- Ex. $c_1$: $\text{avg}(S.\text{profit}) > 20$, and $c_2$: $\text{avg}(S.\text{price}) < 50$
  - Sorted in profit descending order and use $c_1$ first (assuming $c_1$ has more pruning power)
  - For each projected DB, sort transactions in price ascending order (now $c_2$ becomes ant-monotone) and use $c_2$ at mining
Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Mining Diverse Patterns
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Sequential Pattern Mining
- Graph Pattern Mining
- Summary

Mining Long Patterns: Challenges

- Mining long patterns is needed in bioinformatics, social network analysis, software engineering, ...
- But the methods introduced so far mine only short patterns (e.g., length < 10)
- Challenges of mining long patterns
  - The curse of “downward closure” property of frequent patterns
    - Any sub-pattern of a frequent pattern is frequent
    - If \{a_1, a_2, ..., a_{100}\} is frequent, then \{a_1\}, \{a_2\}, ..., \{a_{100}\}, \{a_1, a_2\}, \{a_1, a_3\}, ..., \{a_1, a_{100}\}, \{a_1, a_2, a_3\}, ... are all frequent! There are about \(2^{100}\) such frequent itemsets!
  - No matter searching in breadth-first (e.g., Apriori) or depth-first (e.g., FPgrowth), if we still adopt the “small to large” step-by-step growing paradigm, we have to examine so many patterns, which leads to combinatorial explosion!
Colossal Patterns: A Motivating Example

- $T_1 = 2 3 4 ...... 39 40$
- $T_2 = 1 3 4 ...... 39 40$
- $T_3 = \ldots$
- $T_{40} = 1 2 3 4 ...... 39$
- $T_{41} = 41 42 43 ...... 79$
- $T_{42} = 41 42 43 ...... 79$
- $T_{60} = 41 42 43 ... 79$

- Let min-support $\sigma = 20$
- # of closed/maximal patterns of size 20: about $\binom{40}{20}$
- But there is only one pattern with size close to 40 (i.e., long or colossal):
  - $\alpha = \{41, 42, \ldots, 79\}$ of size 39
- Q: How to find it without generating an exponential number of size-20 patterns?

The existing fastest mining algorithms (e.g., FPClose, LCM) fail to complete running.

A new algorithm, Pattern-Fusion, outputs this colossal pattern in seconds.

What Is Pattern-Fusion?

- Not strive for completeness (why?)
- Jump out of the swamp of the mid-sized intermediate “results”
- Strive for mining almost complete and representative colossal patterns: identify “short-cuts” and take “leaps”
- Key observation
  - The larger the pattern or the more distinct the pattern, the greater chance it will be generated from small ones
- Philosophy: Collection of small patterns hints at the larger patterns
- Pattern fusion strategy (“not crawl but jump”): Fuse small patterns together in one step to generate new pattern candidates of significant sizes
Observation: Colossal Patterns and Core Patterns

- Suppose dataset D contains 4 colossal patterns (below) plus many small patterns
  - \{a_1, a_2, ..., a_{50}\}: 40,
  - \{a_3, a_6, ..., a_{99}\}: 60,
  - \{a_5, a_{10}, ..., a_{95}\}: 80,
  - \{a_{10}, a_{20}, ..., a_{100}\}: 100

- If you check the pattern pool of size-3, you may likely find
  - \{a_2, a_4, a_{45}\}: ~40; \{a_3, a_{34}, a_{39}\}: ~40;
  - ...
  - \{a_5, a_{15}, a_{85}\}: ~80; ..., \{a_{20}, a_{40}, a_{85}\}: ~80
  - ...

- If you merge the patterns with similar support, you may obtain candidates of much bigger size and easily validate whether they are true patterns

Note:
- A colossal pattern has far more core patterns than a small-sized pattern
- A random draw from a complete set of patterns of size c would be more likely to pick a core pattern (or its descendant) of a colossal pattern
- A colossal pattern can be generated by merging a set of its core patterns
Robustness of Colossal Patterns

- Let $D_\alpha$ be the set of transactions containing pattern $\alpha$.
- Core Patterns: For a frequent pattern $\alpha$, a subpattern $\beta$ is a $\tau$-core pattern of $\alpha$ if $\beta$ shares a similar support set with $\alpha$, i.e.,
  \[
  \frac{|D_\alpha|}{|D_\beta|} \geq \tau \quad 0 < \tau \leq 1
  \]
  where $\tau$ is called the core ratio.
- Note that for any subpattern $\beta$ of pattern $\alpha$, we have $|D_\alpha| \leq |D_\beta|$.

Core Patterns: For a frequent pattern $\alpha$, a subpattern $\beta$ is a $\tau$-core pattern of $\alpha$ if $\beta$ shares a similar support set with $\alpha$, i.e.,

- $(d, \tau)$-robustness: A pattern $\alpha$ is $(d, \tau)$-robust if $d$ is the maximum number of items that can be removed from $\alpha$ for the resulting pattern to remain a $\tau$-core pattern of $\alpha$.

- A $(d, \tau)$-robust pattern $\alpha$ has $\Omega(2^d)$ core patterns. (Lemma A)
Robustness of Colossal Patterns

- **Core Patterns:** For a frequent pattern \( \alpha \), a subpattern \( \beta \) is a \( \tau \)-core pattern of \( \alpha \) if \( \beta \) shares a similar support set with \( \alpha \), i.e.,
  \[
  \frac{|D_\alpha \cap D_\beta|}{|D_\beta|} \geq \tau \quad 0 < \tau \leq 1 \] where \( \tau \) is called the core ratio

- **\( (d, \tau) \)-robustness:** A pattern \( \alpha \) is \( (d, \tau) \)-robust if \( d \) is the maximum number of items that can be removed from \( \alpha \) for the resulting pattern to remain a \( \tau \)-core pattern of \( \alpha \)

- For a \( (d, \tau) \)-robust pattern \( \alpha \), it has \( \Omega(2^d) \) core patterns (Lemma A)

- **Robustness of Colossal Patterns:** A colossal pattern tends to have much more core patterns than small patterns

- Such core patterns can be clustered together to form “dense balls” based on pattern distance defined by
  \[
  Dist(\alpha, \beta) = 1 - \frac{|D_\alpha \cap D_\beta|}{|D_\alpha \cup D_\beta|}
  \]

  Lemma A => a random draw in the pattern space will hit somewhere in the ball with high probability!

The Pattern-Fusion Algorithm

- **Initialization** (Creating initial pool):
  Use an existing algorithm to mine all frequent patterns up to a small size, e.g., 3

- **Iteration** (Iterative Pattern Fusion):
  - At each iteration, \( K \) seed patterns are randomly picked from the current pattern pool
  - For each seed pattern thus picked, we find all the patterns within a bounding ball centered at the seed pattern
  - All these patterns found are fused together to generate a set of super-patterns
  - All the super-patterns thus generated form a new pool for the next iteration

- **Termination:** when the current pool contains no more than \( K \) patterns at the beginning of an iteration
Experimental Results on Data Set: ALL

- ALL: a gene expression clinical data set on ALL-AML leukemia, with 38 transactions, each with 866 columns. There are 1736 items in total.
- When minimum support is high (e.g., 30), Pattern-Fusion gets all the largest colossal patterns with size greater than 85.

<table>
<thead>
<tr>
<th>Pattern Size</th>
<th>110</th>
<th>107</th>
<th>102</th>
<th>91</th>
<th>86</th>
<th>84</th>
<th>83</th>
</tr>
</thead>
<tbody>
<tr>
<td>The complete set</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Pattern-Fusion</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Pattern Size</td>
<td>82</td>
<td>77</td>
<td>76</td>
<td>75</td>
<td>74</td>
<td>73</td>
<td>71</td>
</tr>
<tr>
<td>The complete set</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Pattern-Fusion</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

![Algorithm runtime comparison on another dataset]

Chapter 7: Advanced Frequent Pattern Mining

- Pattern Mining: A Road Map
- Mining Diverse Patterns
- Constraint-Based Frequent Pattern Mining
- Mining High-Dimensional Data and Colossal Patterns
- Sequential Pattern Mining
- Graph Pattern Mining

Next Lectures
Data Mining Assignment 1

- Available tomorrow Wednesday 21-11 2018, 13.00 CET.
- Deadline Friday 7-12 2018, 23:59 CET.
- Groups of 1 – 4 persons.
  - Association Rule mining on a given data set.
  - Data pre-processing (Coding: Python, JAVA, ALGOL68, C++, …).
  - Data analysis (Weka, R, Python, C++, …).
  - Data Mining (Weka, other tools or libraries, Python, …).
  - Find interesting rules (Use interestingness measures, etc.).

Repeat, if necessary.