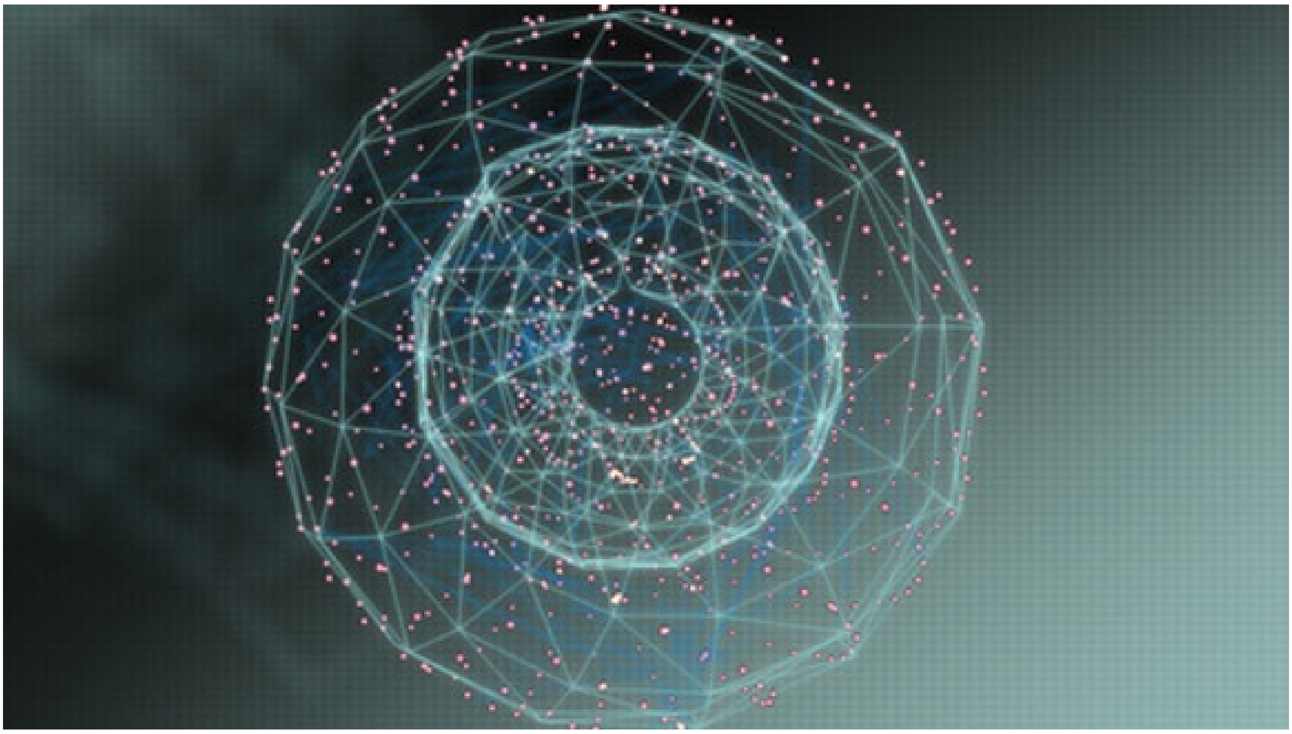
PATTERN DISCOVERY
IN
DATA MINING

Concepts and challenges in pattern discovery and analysis.
Pattern evaluation, mining and classification

Course author:

JIAWEI HAN



University of Illinois at Urbana-Champaign
&
Coursera

2015

Contents

1 Lecture 2: Pattern Discovery Basic Concepts

1.1 Frequent Itemsets (Patterns)

X = itemset

- **(absolute) support (count) of X:** Frequency or the number of occurrences of an itemset X
- **(relative) support, s:** The fraction of transactions that contains X (i.e., the probability that a transaction contains X)
- An itemset X is **frequent** if the support of X is no less than a $minsup$ threshold (denoted as σ): $sup(X) \geqslant \sigma$.

1.2 Association Rules

Association rules: $X \to Y(s, c)$:

- **Support** (s): the probability that a transaction contains $X \cup Y$:

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

- **Confidence** (c): the conditional probability that a transaction containing X also contains Y:

$$c = \mathrm{P}(Y \mid X) = \frac{sup(X \cup Y)}{sup(X)}$$

1.3 Expressing Patterns in Compressed Form

Definition. Closed patterns: *A pattern (itemset) X is closed if X is frequent, and there exists no super-pattern $Y \supset X$, with the same support as X.*

Closed pattern is a lossless compression of frequent patterns.

Definition. Max-patterns: *A pattern X is a max-pattern if X is frequent and there exists no frequent super-pattern $Y \supset X$.*

Max-pattern is a lossy compression!

1.4 Recommended readings

- R. Agrawal, T. Imielinski, and A. Swami, «Mining association rules between sets of items in large databases», in Proc. of SIGMOD'93
- R. J. Bayardo, «Efficiently mining long patterns from databases», in Proc. of SIGMOD'98
- N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, «Discovering frequent closed itemsets for association rules», in Proc. of ICDT'99
- J. Han, H. Cheng, D. Xin, and X. Yan, «Frequent Pattern Mining: Current Status and Future Directions», Data Mining and Knowledge Discovery, 15(1): 55-86, 2007

2 Lecture 3. Efficient Pattern Mining Methods

2.1 The Downward Closure Property of Frequent Patterns

The downward closure (also called «Apriori») property of frequent patterns: **Any subset of a frequent itemset must be frequent**. Apriori pruning principle: **If there is any itemset which is infrequent, its superset should not even be generated!**

Scalable mining Methods: Three major approaches

- Level-wise, join-based approach: Apriori (2.2)
- Vertical data format approach: Eclat (2.4)
- Frequent pattern projection and growth: FPgrowth (2.5)

2.2 The Apriori Algorithm

2.2.1 Algorithm pseudocode

C_k: Candidate itemset of size k
F_k : Frequent itemset of size k
TDB = transactional database

Algorithm 1 The Apriori Algorithm

```
k := 1
F_k := frequent items                                # frequent 1-itemset
while F_k ≠ ∅ do
   C_{k+1} := candidates generated from F_k          # candidate generation
   Derives F_{k+1} by counting candidates in C_{k+1} with respect to TDB at minsup
   k := k + 1
end while
return ∪_k F_k                          # return F_k generated at each level
```

2.2.2 How to generate candidates?

- Step1: self-joining F_k
- Step2: pruning

Algorithm 2 Step1: self-joining F_k

```
insert into C_k
select p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}
from F_{k-1} as p, F_{k-1} as q
where p.item_1= q.item_1, ..., p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}
```

Algorithm 3 Step2: pruning

```
for all itemsets c in C_k do
   for all (k-1) subsets s of c do
      if s is not in F_k-1 then
         delete c from C_k
      end if
   end for
end for
```

2.3 Extensions or Improvements of Apriori

- Reduce passes of transaction database scans
  - Partitioning
  - Dynamic itemset counting
- Shrink the number of candidates
  - Hashing
  - Pruning by support lower bounding
  - Sampling
- Exploring special data structures
  - Tree projection
  - H-miner
  - Hypecube decomposition

2.3.1 Partitioning

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

Method: Scan Database Only Twice:

- Scan 1: Partition database (how?) and find local frequent patterns
- Scan 2: Consolidate global frequent patterns (how to?)

2.3.2 Direct Hashing and Pruning (DHP)

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

2.4 Vertical Data Format

ECLAT - Equivalence Class Transformation

Frequent patterns are derived based on vertical intersections. To accelerate data mining you can use **diffset**: only keep track of differences of tids.

2.5 A Pattern Growth Approach

FP-tree - frequent pattern tree

TID	Items in the Transaction	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	$\{a, f, c, e, l, p, m, n\}$	$\{f, c, a, m, p\}$

Figure 1: Transational DB

Header Table

Item	Frequency	header
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

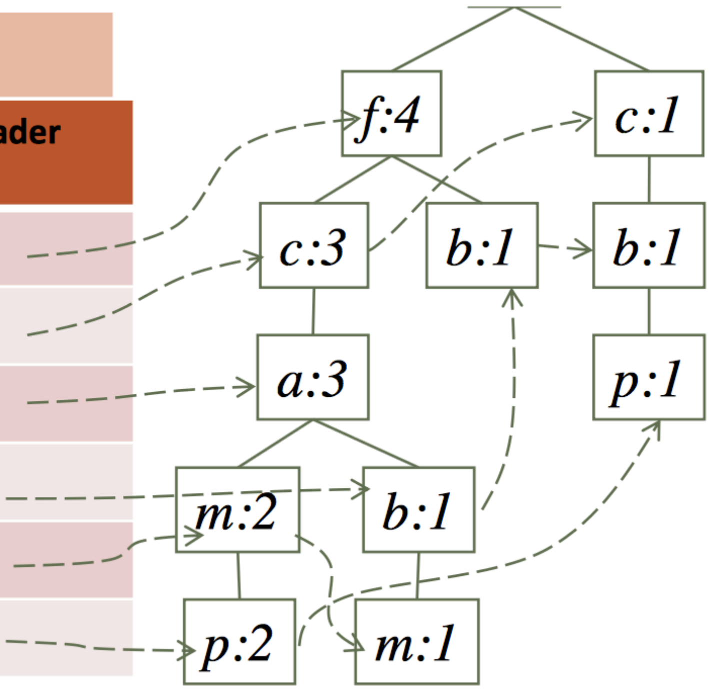


Figure 2: FP-tree

2.6 CLOSET+: Mining Closed Itemsets by Pattern-Growth

Itemset merging: *If Y appears in every occurrence of X, then Y is merged with X*

2.7 Recommended readings

- R. Agrawal and R. Srikant, «Fast algorithms for mining association rules», VLDB'94
- A. Savasere, E. Omiecinski, and S. Navathe, «An efficient algorithm for mining association rules in large databases», VLDB'95
- J. S. Park, M. S. Chen, and P. S. Yu, «An effective hash-based algorithm for mining association rules», SIGMOD'95

- S. Sarawagi, S. Thomas, and R. Agrawal, «Integrating association rule mining with relational database systems: Alternatives and implications», SIGMOD'98
- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, «Parallel algorithm for discovery of association rules», Data Mining and Knowledge Discovery, 1997
- J. Han, J. Pei, and Y. Yin, «Mining frequent patterns without candidate generation», SIGMOD'00
- M. J. ZakiandHsiao, «CHARM: An Efficient Algorithm for Closed Itemset Mining», SDM'02
- J. Wang, J. Han, and J. Pei, «CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets», KDD'03
- C. C. Aggarwal, M.A., Bhuiyan, M. A. Hasan, «Frequent Pattern Mining Algorithms: A Survey», in Aggarwal and Han (eds.): Frequent Pattern Mining, Springer, 2014

3 Lecture 4: Pattern Evaluation

3.1 Interestingness Measures: Lift and χ^2

3.1.1 Interestingness Measure: Lift

Lift - measure of dependent/correlated events:

$$\text{lift}(B, C) = \frac{c(B \to C)}{s(C)} = \frac{s(B \cup C)}{s(B) \times s(C)}$$

Lift(B, C) may tell how B and C are correlated:

- $\text{Lift}(B, C) = 1$: B and C are independent
- $\text{Lift}(B, C) > 1$: positively correlated
- $\text{Lift}(B, C) < 1$: negatively correlated

3.1.2 Interestingness Measure: χ^2

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

General rules:

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

Too many null transactions may lead to invalid correlation result!

3.2 Null Invariance Measures

$$\text{AllConf}(A,B) = \frac{s(A \cup B)}{\max\{s(A), s(B)\}}$$
$$\text{Jaccard}(A,B) = \frac{s(A \cup B)}{s(A) + s(B) - s(A \cup B)}$$
$$\text{Cosine}(A,B) = \frac{s(A \cup B)}{\sqrt{s(A) \times s(B)}}$$
$$\text{Kulczynsky}(A,B) = \frac{1}{2}\left(\frac{s(A \cup B)}{s(A)} + \frac{s(A \cup B)}{s(B)}\right)$$
$$\text{MacConf}(A,B) = \max\left\{\frac{s(A)}{s(A \cup B)}, \frac{s(B)}{s(A \cup B)}\right\}$$

3.3 Imbalance Ratio

IR (Imbalance Ratio): measure the imbalance of two itemsets A and B in rule implications:

$$\text{IR}(A,B) = \frac{|s(A) - s(B)|}{s(A) + s(B) - s(A \cup B)}$$

Kulczynski and Imbalance Ratio (IR) together present a clear picture

3.4 Recommended Readings

- C. C. Aggarwal and P. S. Yu. A New Framework for Itemset Generation. PODS'98
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97
- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94
- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE'03
- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02
- T. Wu, Y. Chen and J. Han, Re-Examination of Interestingness Measures in Pattern Mining: A Unified Framework, Data Mining and Knowledge Discovery, 21(3):371-397, 2010

4 Lecture 4: Mining Diverse Patterns

4.1 Mining Multi-Level Associations

Items often form hierarchies. How to set min-support thresholds? **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.

Redundancy[1] filtering: some rules may be redundant due to «ancestor»[2] relationships between items. A rule is **redundant** if:

- its support is close to the «expected» value, according to its «ancestor» rule
- it has a similar confidence as its «ancestor».

It is necessary to have customized min-support settings for different kinds of items: group-based «individualized» min-support.

4.2 Mining Multi-Dimensional Associations

Rules can be single-dimensional or multi-dimensional:

- Single-dimentional:

$$\mathrm{buys}(X, \text{«milk»}) \Rightarrow \mathrm{buys}(X, \text{«bread»})$$

- Inter-dimension association rule:

$$\mathrm{age}(X, \text{«18-25»}) \wedge \mathrm{occupation}(X, \text{«student»}) \Rightarrow \mathrm{buys}(X, \text{«coke»})$$

- Hybrid-dimension association rules:

$$\mathrm{age}(X, \text{«18-25»}) \wedge \mathrm{buys}(X, \text{«popcorn»}) \Rightarrow \mathrm{buys}(X, \text{«coke»})$$

Attributes can be categorical or numerical

4.3 Mining Quantitative Associations

Methods:

- Static discretization based on predefined concept hierarchies
- Dynamic discretization based on data distribution
- Clustering: distance-based association
- Deviation analysis

4.4 Mining Negative Correlations

- Rare patterns = very low support but interesting
- Negative patterns = negatively correlated, unlikely to happen together

A support-based definition: if itemsets A and B are both frequent but rarely occur together, i.e., $\mathrm{sup}(A \cup B) << \mathrm{sup}(A) \times \mathrm{sup}(B)$ then A and B are negatively correlated.

The support-based definition is not null-invariant!

A Kulczynski measure-based definition: if itemsets A and B are frequent but $\frac{P(A|B)+P(B|A)}{2} < \varepsilon$, where ε is a negative pattern threshold, then A and B are negatively correlated.

[1]Redundancy - избыточность
[2]Ancestor – предок

4.5 Mining Compressed Patterns

4.5.1 Mining Compressed Patterns

Pattern distance measure:

$$Dist(P_1, P_2) = 1 - \frac{|T(P_1) \cap T(P_2)|}{|T(P_1) \cup T(P_2)|}$$

δ-clustering. For each pattern P, find all patterns which can be expressed by P and whose distance to P is within δ (δ-cover). All patterns in the cluster can be represented by P = compressed patterns.[3]

4.5.2 Redundancy-Aware Top-k Patterns

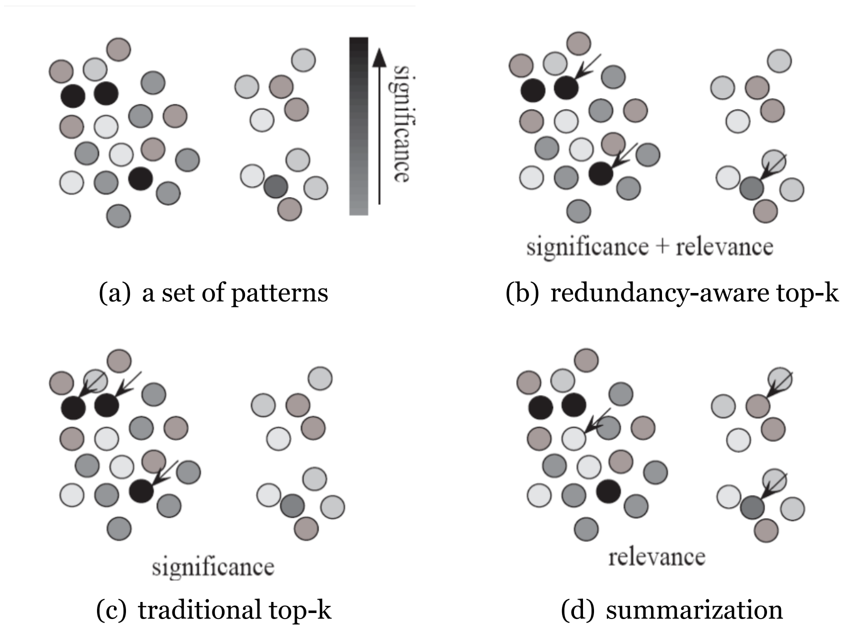


Figure 3: Desired patterns: high significance & low redundancy

Use **MMS (Maximal Marginal Significance)** for measuring the combined significance of a pattern set.[4]

4.6 Mining Colossal Patterns

4.6.1 Pattern-Fusion

Pattern fusion strategy: fuse small patterns together in one step to generate new pattern candidates of significant sizes.

Subpatterns α_1 to α_k cluster tightly around the colossal pattern α by sharing a similar support. Such subpatterns are **core patterns** of α. A colossal pattern can be generated by merging a set of core patterns.

[3]Method for efficient, direct mining of compressed frequent patterns: Xin et al., VLDB'05.
[4]Xin et al., Extracting Redundancy-Aware Top-K Patterns, KDD'06.

4.6.2 Robustness of Colossal Patterns

Definition. For a frequent pattern α, a subpattern β is a **τ-core pattern** of α if β shares a similar support set with α, i.e.,

$$\frac{|D_\alpha|}{|D_\beta|} \geqslant \tau, 0 < \tau \leqslant 1,$$

where τ is called the **core ratio**.

Definition. **(d, τ)-robustness**[5]: a pattern α is (d, τ)-robust if d is the maximum number of items that can be removed from α for the resulting pattern to remain a τ-core pattern of α:

$$d = \max_\beta\{|\alpha| - |\beta| \mid \beta \subseteq \alpha, \text{ and } \beta \text{ is a } \tau\text{-core pattern of } \alpha\}$$

For a pattern α let C_α be the set of all its core patterns for a specified τ:

$$C_\alpha = \{\beta \mid \beta \subseteq \alpha, \frac{|D_\alpha|}{|D_\beta|} \geqslant \tau\}$$

Theorem. *For a (d, τ)-robust pattern α:*

$$|C_\alpha| \geqslant 2^d$$

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

$$Dist(\alpha, \beta) = 1 - \frac{|D_\alpha \cap D_\beta|}{|D_\alpha \cup D_\beta|}$$

Theorem. *For two patterns $\beta_1, \beta_2 \in C_\alpha$*

$$Dist(\beta_1, \beta_2) \leqslant r(\tau), \textit{ where } r(\tau) = 1 - \frac{1}{2/\tau - 1}$$

4.6.3 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

[5]Robustness - прочность

4.7 Recommended Readings

- R. Srikant and R. Agrawal, «Mining generalized association rules», VLDB'95
- Y. Aumann and Y. Lindell, «A Statistical Theory for Quantitative Association Rules», KDD'99
- D. Xin, J. Han, X. Yan and H. Cheng, «On Compressing Frequent Patterns», Knowledge and Data Engineering, 60(1): 5-29, 2007
- D. Xin, H. Cheng, X. Yan, and J. Han, «Extracting Redundancy-Aware Top-K Patterns», KDD'06
- F. Zhu, X. Yan, J. Han, P. S. Yu, and H. Cheng, «Mining Colossal Frequent Patterns by Core Pattern Fusion», ICDE'07
- J. Han, H. Cheng, D. Xin, and X. Yan, «Frequent Pattern Mining: Current Status and Future Directions», Data Mining and Knowledge Discovery, 15(1): 55-86, 2007

5 Constraint-Based Pattern Mining

5.1 Meta-Rule Guided Mining

In general, (meta) rules can be in the form of

$$P_1 \wedge P_2 \wedge \dots \wedge P_l \Rightarrow Q_1 \wedge Q_2 \wedge \dots \wedge Q_r$$

Method to find meta-rules:

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

5.2 Kinds of Constraints

- Pattern space pruning constraints
  - Anti-monotonic: If constraint c is violated, its further mining can be terminated
  - Monotonic: If c is satisfied, no need to check c again
  - Succinct[6]: if the constraint c can be enforced by directly manipulating the data
  - Convertible: 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 does not satisfy c, then t can be pruned to reduce data processing effort

Anti-monotonic constraints have more pruning power than monotonic constraints.

[6] Succinct - краткий

5.2.1 Pattern space pruning constraints

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. For example, constraint $\sup(S) \geqslant \sigma$ is anti-monotone.

A constraint c is **monotone**: if an itemset S satisfies the constraint **c**, so does any of its superset. That is, we do not need to check **c** in subsequent mining. For example, constraints $\mathrm{sum}(S.price) \geqslant v$ or $\min(S.price) \leqslant v$ are monotone.

5.2.2 Data space pruning constraints

A constraint **c** is **data anti-monotone**: if a data entry **t** cannot satisfy a pattern **p** under constraint **c**, **t** cannot satisfy **p**'s superset either. That's why, data entry **t** can be pruned.

Succinctness: if the constraint **c** can be enforced by directly manipulating the data.

Convertible constraints: convert tough[7] constraints into (anti-)monotone by proper ordering of items in transactions. For example, ordering items in value-descending order makes the constraint $\mathrm{avg}(S.profit) > 20$ anti-monotone *if the patterns grow in the right order*.

5.3 Recommended Readings

- R. Srikant, Q. Vu, and R. Agrawal, «Mining association rules with item constraints», KDD'97
- R. Ng, L.V.S. Lakshmanan, J. Han & A. Pang, Exploratory mining and pruning optimizations of constrained association rules», SIGMOD'98
- G. Grahne, L. Lakshmanan, and X. Wang, «Efficient mining of constrained correlated sets», ICDE'00
- J. Pei, J. Han, and L. V. S. Lakshmanan, «Mining Frequent Itemsets with Convertible Constraints», ICDE'01
- J. Pei, J. Han, and W. Wang, «Mining Sequential Patterns with Constraints in Large Databases», CIKM'02
- F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi, «ExAnte: Anticipated Data Reduction in Constrained Pattern Mining», PKDD'03
- F. Zhu, X. Yan, J. Han, and P. S. Yu, «gPrune: A Constraint Pushing Framework for Graph Pattern Mining», PAKDD'07

6 Sequential Pattern Mining

6.1 Sequential Pattern

Sequence $\rightarrow$ Element $\rightarrow$ Item or Event (items within an element are unordered)

[7]Tough - жесткий

The Apriori property still holds: if a subsequence s_1 is infrequent, none of s_1's super-sequences can be frequent.

Algorithms:

- Generalized Sequential Patterns: **GSP**
- Vertical format-based mining: **SPADE**
- Pattern-growth methods: **PrefixSpan**
- Mining closed sequential patterns: **CloSpan**

6.2 GSP: Apriori-Based Sequential Pattern Mining

Algorithm 4 GSP

```
k = 1
repeat
   find length=k frequent sequences
   Apriori: remove candidates with sup < min_sup
   length=k frequent sequences ⇒ length=(k+1) candidate sequences
   k = k + 1
until no frequent sequences or candidates
```

6.3 SPADE: Sequential Pattern Mining in Vertical Data Format

SPADE = **S**equential **P**attern **D**iscovery using **E**quivalent Class

SID	Sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

min_sup = 2

Figure 4: A sequence database

SID	EID	Items
1	1	a
1	2	abc
1	3	ac
1	4	d
1	5	cf
2	1	ad
2	2	c
2	3	bc
2	4	ae
3	1	ef
3	2	ab
3	3	df
3	4	c
3	5	b
4	1	e
4	2	g
4	3	af
4	4	c
4	5	b
4	6	c

a		b		...
SID	EID	SID	EID	...
1	1	1	2	
1	2	2	3	
1	3	3	2	
2	1	3	5	
2	4	4	5	
3	2			
4	3			

ab			ba		
SID	EID (a)	EID(b)	SID	EID (b)	EID(a)
1	1	2	1	2	3
2	1	3	2	3	4
3	2	5			
4	3	5			

aba				...
SID	EID (a)	EID(b)	EID(a)	...
1	1	2	3	
2	1	3	4	

Figure 5: SPADE algorithm

6.4 PrefixSpan: A Pattern-Growth Approach

PrefixSpan = Prefix-projected Sequential pattern mining

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

Prefix	Suffix (Projection)
<a>	<(abc)(ac)d(cf)>
<aa>	<(_bc)(ac)d(cf)>
<ab>	<(_c)(ac)d(cf)>

Figure 6: SPADE algorithm

PrefixSpan Mining: Prefix Projections

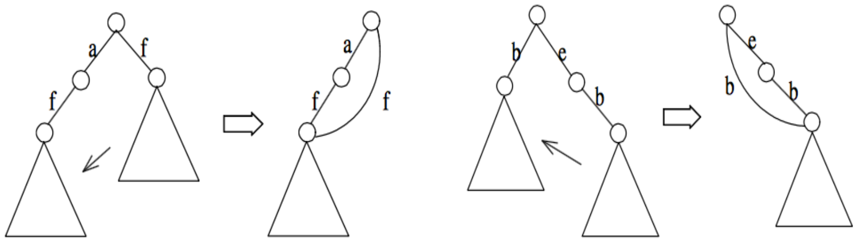
- Step 1: Find length-1 sequential patterns: <a>, <b>, etc.
- Step 2: Divide search space and mine each projected DB: <a>-projected DB, <b>-projected DB, etc.

6.5 CloSpan: Mining Closed Sequential Patterns

Definition. A closed sequential pattern α: there exists no superpattern β such that β and α have the same support:

$$CS = \{\alpha \mid \alpha \in FS \text{ and } \nexists \beta \in FS\text{, such that } \alpha \subseteq \beta \text{ and } sup(\alpha) = sup(\beta)\}$$

CloSpan is based on this property: if $s \supset s_1$ then s is closed only if two project DBs have the same size. So redundant search space can be pruned using **Backward Subpattern** and **Backward Superpattern** pruning.



(a) backward sub-pattern (b) backward super-pattern

Figure 7: CloSpan pruning algorithm

6.6 Constraint-Based Sequential-Pattern Mining

- **Anti-monotonic**: If S violates c, the super-sequences of S also violate c
- **Monotonic**: If S satisfies c, the super-sequences of S also do so
- **Data anti-monotonic**: If a sequence s1 with respect to S violates c3, s1 can be removed
- **Succinct**: Enforce constraint c by explicitly manipulating data
- **Convertible**: Projection based on the sorted value not in sequence order

6.6.1 Timing-Based Constraints

- **Order constraint**: Some items must happen before the other. Anti-monotonic: constraint-violating sub-patterns pruned
- **Min-gap/max-gap constraint**: Confines two elements in a pattern. Succinct: enforced directly during pattern growth
- **Max-span constraint**: Maximum allowed time difference between the 1st and the last elements in the pattern. Succinct: enforced directly when the 1st element is determined
- **Window size constraint**: Time window allows a group of consecutive elements of a data-sequence to be merged and treated as a single element as long as their timestamps are within the user-specified window-size.

6.7 Recommended Readings

- M. N. Garofalakis, R. Rastogi, K. Shim: Mining Sequential Patterns with Regular Expression Constraints. IEEE Trans. Knowl. Data Eng. 14(3), 2002

- H. Mannila, H. Toivonen, and A. I. Verkamo, "Discovery of frequent episodes in event sequences", Data Mining and Knowledge Discovery, 1997
- J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach", IEEE TKDE, 16(10), 2004
- J. Pei, J. Han, and W. Wang, "Constraint-based sequential pattern mining: the pattern-growth methods", J. Int. Inf. Sys., 28(2), 2007
- R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements", EDBT'96
- X. Yan, J. Han, and R. Afshar, "CloSpan: Mining Closed Sequential Patterns in Large Datasets", SDM'03
- M. Zaki, "SPADE: An Efficient Algorithm for Mining Frequent Sequences", Machine Learning, 2001

7 Lecture 8. Graph Pattern Mining

7.1 Frequent (Sub)Graph Patterns

Given a labeled graph dataset $D = \{G_1, G_2, ..., G_n\}$, the supporting graph set of a subgraph g is $D_g = \{G_i \mid g \subseteq G_i, G_i \in D\}$:

$$support(g) = |D_g| / |D|$$

A (sub)graph g is frequent if $support(g) \geqslant min\_sup$.

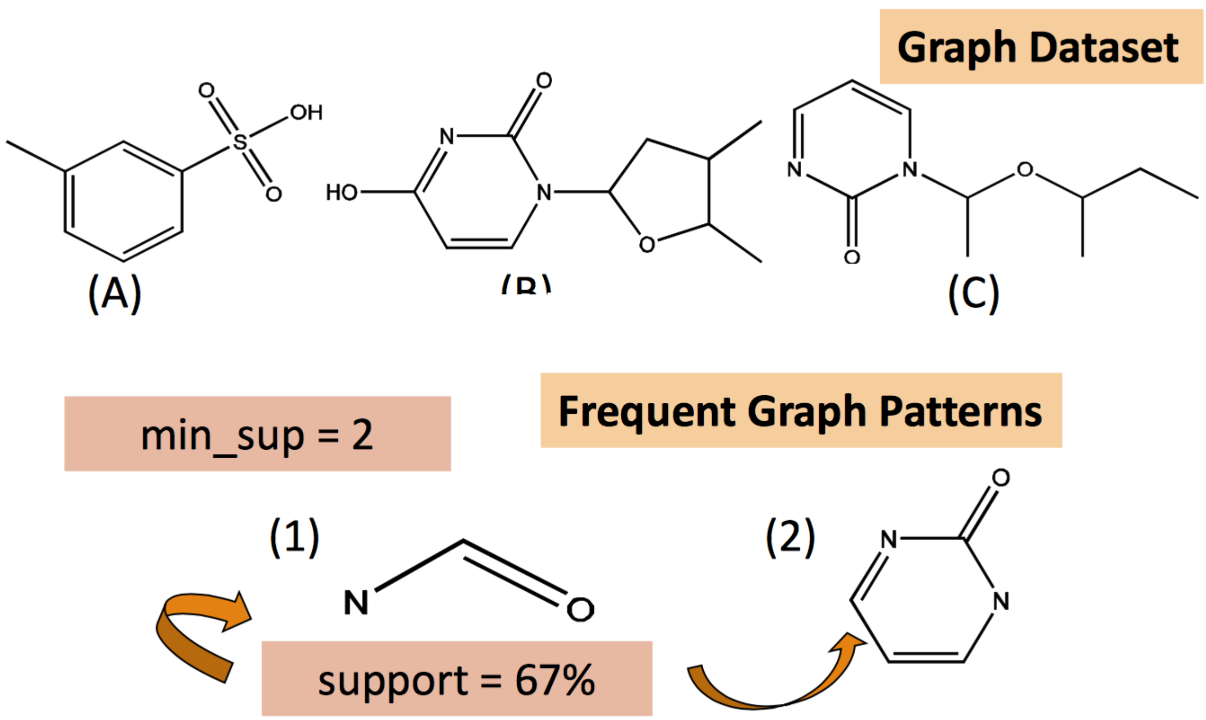


Figure 8: Example: Chemical structures

7.2 Apriori-Based Approach

The Apriori property (anti-monotonicity): a size-k subgraph is frequent if and only if all of its subgraphs are frequent.

Candidate generation: a candidate size-(k+1) edge/vertex subgraph is generated if its corresponding two k-edge/vertex subgraphs are frequent:

- AGM - Generating new graphs with one more vertex
- FSG - Generating new graphs with one more edge (more efficient)

Iterative mining process: Candidate-generation $\rightarrow$ candidate pruning $\rightarrow$ support counting $\rightarrow$ candidate elimination.

7.3 gSPAN: Graph Pattern Growth

Depth-first growth of subgraphs from k-edge to (k+1)-edge, then (k+2)-edge subgraphs generates many duplicate subgraphs.

Right-most path extension in subgraph pattern growth reduces generation of duplicate subgraphs: *take the path from root to the right-most leaf (choose the vertex with the smallest index at each step)*. The Enumeration of graphs using right-most path extension is complete.

DFS Code: flatten a graph into a sequence using depth-first search

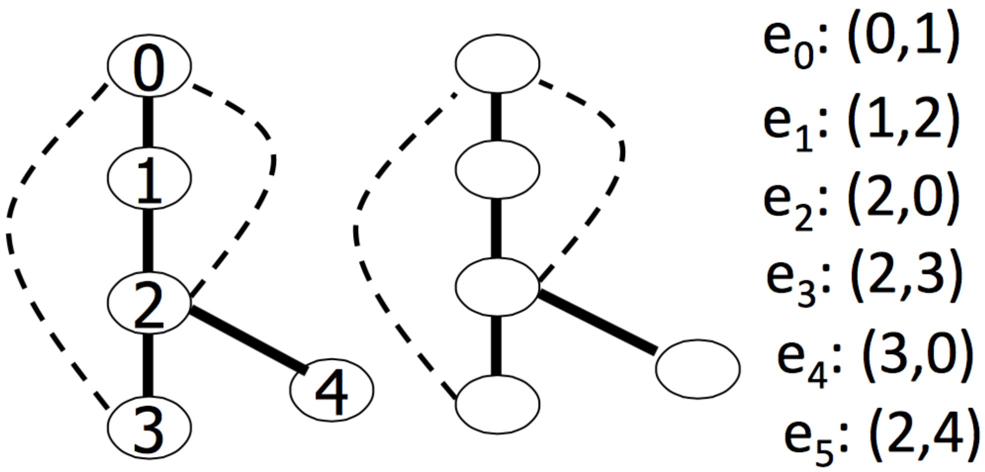


Figure 9: gSPAN

7.4 Mining Closed Graph Patterns

A frequent graph G is closed if there exists no supergraph of G that carries the same support as G.

CloseGraph algorithm: mining closed graph patterns by extending gSpan. Suppose G and G_1 are frequent, and G is a subgraph of G_1. If in any part of the graph in the dataset where G occurs, G_1 also occurs, then we need not grow G (except some special, subtle cases), since *none of G's children will be closed except those of* G_1.

7.5 gIndex: A Graph Indexing Method

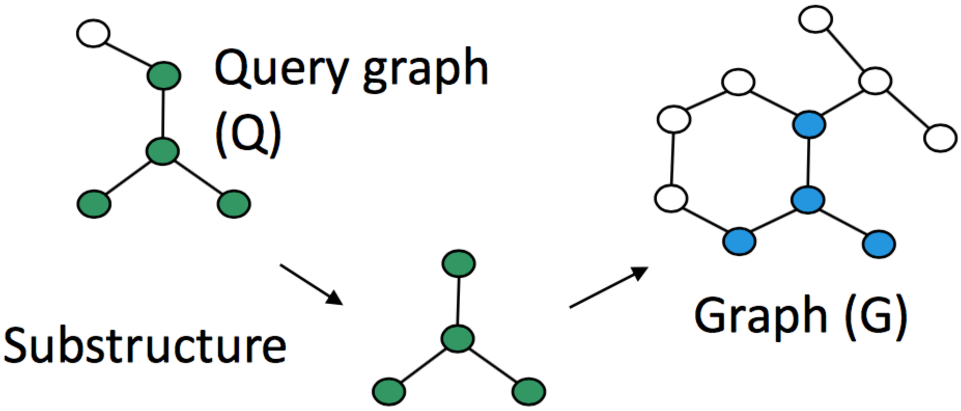


Figure 10: Graph query

- use frequent substructures for indexing
- discriminative substructures: reduce index size by removing similar (not discriminative) substructures from the index

Definition. Fragment x is **discriminative** with respect to feature set F if $D_x \ll \bigcap_{f \in F \wedge f \subseteq x} D_f$, where D_x is the set of graphs containing x.

Selection: Given a set of indexing features $f_1, f_2, ..f_n$, and a new structure x (x should be either redundant or discriminative), the extra indexing power is measured by occurrence probability

$$Pr(x \mid f_1, f_2, ...f_n) = \frac{\left|\bigcap_{f \in F \wedge f \subseteq x} D_f\right|}{|D_x|}$$

When $Pr(x \mid f_1, f_2, ...f_n) \ll 1$, x is a discriminative structure and should be included in the index.

7.6 SpiderMine: Mining Top-K Large Structural Patterns in a Massive Network

SpiderMine: mine top-K largest frequent substructure patterns whose diameter is bounded by $_{Dmax}$ with a probability at least $1 - \varepsilon$. General idea: large patterns are composed of a number of small components («spiders») which will eventually connect together after some rounds of pattern growth.

An r-spider is a frequent graph pattern P such that there exists a vertex u of P, and all other vertices of P are within distance r from u.

The SpiderMine Algorithm

- Mine the set S of all the r-spiders
- Randomly draw M r-spiders

- Grow these M r-spiders for $t = D_{max}/2$ iterations, and merge two patterns whenever possible
- Discard unmerged patterns
- Continue to grow the remaining ones to maximum size
- Return the top-K largest ones in the result

SpiderMine general ideas:

- Small patterns are much less likely to be hit in the random draw
- Even if a small pattern is hit, it is even less likely to be hit multiple times
- The larger the pattern, the greater the chance it is hit and saved

7.7 Recommended Readings

- C. Borgelt and M. R. Berthold, «Mining molecular fragments: Finding relevant substructures of molecules», ICDM'02
- J. Huan, W. Wang, and J. Prins. «Efficient mining of frequent subgraph in the presence of isomorphism», ICDM'03
- A. Inokuchi, T. Washio, and H. Motoda. «An apriori-based algorithm for mining frequent substructures from graph data», PKDD'00
- M. Kuramochi and G. Karypis. «Frequent subgraph discovery», ICDM'01
- S. Nijssen and J. Kok. A quickstart in frequent structure mining can make a difference. KDD'04
- N. Vanetik, E. Gudes, and S. E. Shimony. «Computing frequent graph patterns from semistructured data», ICDM'02
- X. Yan and J. Han, «gSpan: Graph-Based Substructure Pattern Mining», ICDM'02
- X. Yan and J. Han, «CloseGraph: Mining Closed Frequent Graph Patterns», KDD'03
- X. Yan, P. S. Yu, and J. Han, «Graph Indexing: A Frequent Structure-based Approach», SIGMOD'04
- F. Zhu, Q. Qu, D. Lo, X. Yan, J. Han, and P. S. Yu, «Mining Top-K Large Structural Patterns in a Massive Network», VLDB'11