Friedrich-Alexander-Universität
Technische Fakultät

6. Mining Frequent Patterns

Knowledge Discovery in Databases with Exercises

Dominik Probst, dominik.probst@fau.de
Computer Science 6 (Data Management), Friedrich-Alexander-Universität Erlangen-Nürnberg
Summer semester 2025

Outline

Basic Concepts

What is Frequent-pattern Analysis?

Frequent Pattern

A pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a dataset.

- **Finding inherent regularities in data:**
  - What products are often purchased together? Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - Who bought this has often also bought . . .
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?

What is Frequent-pattern Analysis?

Frequent Pattern

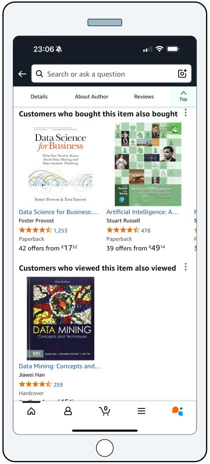
A pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a dataset.

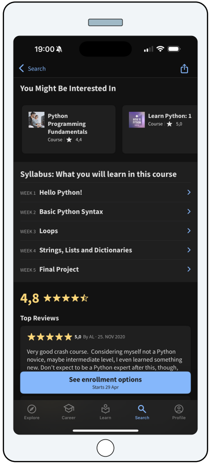
- **Finding inherent regularities in data:**
  - What products are often purchased together? Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - Who bought this has often also bought . . .
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?

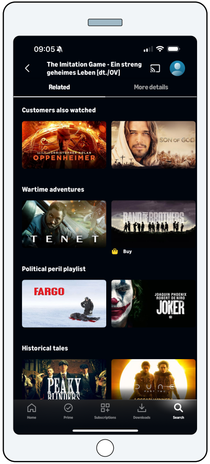
Intrinsic and Important

A frequent pattern is an intrinsic and important property of a dataset.

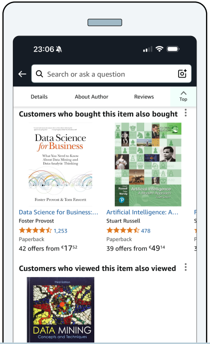
Some Real World Examples

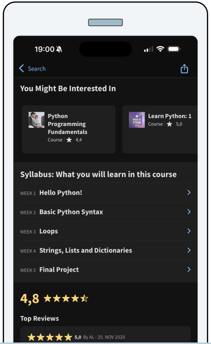


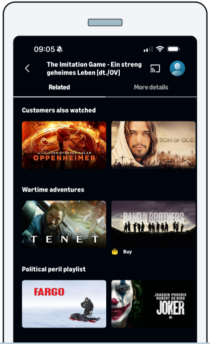




Some Real World Examples







Recommendation Systems

While frequent pattern analysis often serves as the **foundation** of recommendation systems, such systems typically consist of multiple distinct components.

Why is Frequent-pattern Mining Important?

- **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[1]
  - Broad applications.

[1] H. Jagadish et al., "Semantic compression and pattern extraction with fascicles," in *VLDB*, vol. 99, 1999, pp. 186–97

Basic Concepts: Frequent Itemsets

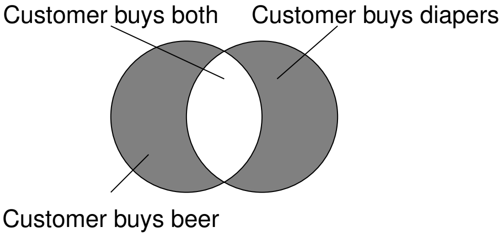
- **Itemset:**
  - A set of one or more items.
  - k-itemset $X = \{x_1, x_2, \dots, x_k\}$.
- **Support:**
  - **Absolute Support** s/**Support Count** of X:
    - Frequency or occurrence count of X.
  - **Relative Support** s:
    - The fraction of the transactions that contain X.
    - I.e. the **probability** that a transaction contains X.

Frequent Itemset

An itemset X is frequent, if X's support is no less than a `min_sup` threshold.

Frequent Itemsets - Example

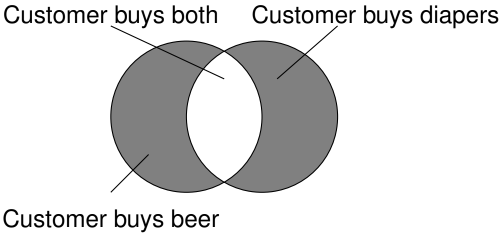
TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk



- **Minimum (absolute) support threshold:**
  - Set by the user.
  - In this example: $min\_sup = 3$.
- **Frequent Itemsets:**
  - **1-itemsets:**
    -
  - **2-itemsets:**
    -
  - **3-itemsets:**
    -
  - **4-itemsets:**
    -
  - **5-itemsets:**
    -

Frequent Itemsets - Example

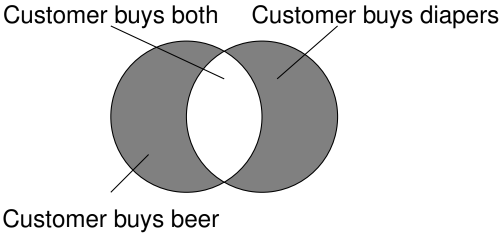
TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk



- **Minimum (absolute) support threshold:**
  - Set by the user.
  - In this example: $min\_sup = 3$.
- **Frequent Itemsets:**
  - **1-itemsets:**
    - {Beer}: 3, {Nuts}: 3, {Diapers}: 4, {Eggs}: 3.
  - **2-itemsets:**
    -
  - **3-itemsets:**
    -
  - **4-itemsets:**
    -
  - **5-itemsets:**
    -

Frequent Itemsets - Example

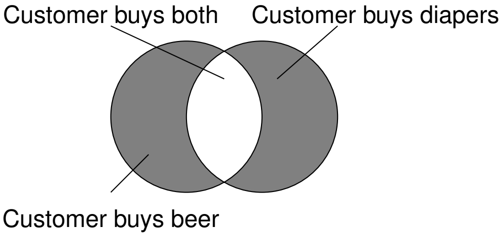
TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk



- **Minimum (absolute) support threshold:**
  - Set by the user.
  - In this example: $min\_sup = 3$.
- **Frequent Itemsets:**
  - **1-itemsets:**
    - {Beer}: 3, {Nuts}: 3, {Diapers}: 4, {Eggs}: 3.
  - **2-itemsets:**
    - {Beer, Diapers}: 3
  - **3-itemsets:**
    -
  - **4-itemsets:**
    -
  - **5-itemsets:**
    -

Frequent Itemsets - Example

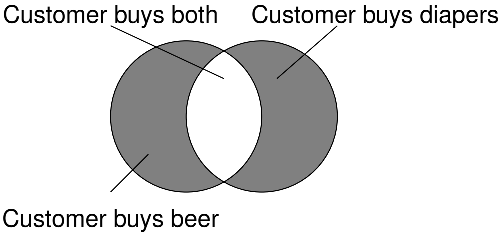
TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk



- **Minimum (absolute) support threshold:**
  - Set by the user.
  - In this example: $min\_sup = 3$.
- **Frequent Itemsets:**
  - **1-itemsets:**
    - {Beer}: 3, {Nuts}: 3, {Diapers}: 4, {Eggs}: 3.
  - **2-itemsets:**
    - {Beer, Diapers}: 3
  - **3-itemsets:**
    - *None*
  - **4-itemsets:**
    -
  - **5-itemsets:**
    -

Frequent Itemsets - Example

TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk

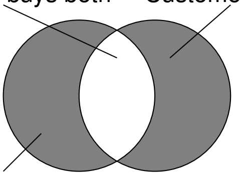


- **Minimum (absolute) support threshold:**
  - Set by the user.
  - In this example: $min\_sup = 3$.
- **Frequent Itemsets:**
  - **1-itemsets:**
    - {Beer}: 3, {Nuts}: 3, {Diapers}: 4, {Eggs}: 3.
  - **2-itemsets:**
    - {Beer, Diapers}: 3
  - **3-itemsets:**
    - *None*
  - **4-itemsets:**
    - *None*
  - **5-itemsets:**
    -

Frequent Itemsets - Example

Friedrich-Alexander-Universität
Technische Fakultät

TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk



- **Minimum (absolute) support threshold:**
  - Set by the user.
  - In this example: $min\_sup = 3$.
- **Frequent Itemsets:**
  - **1-itemsets:**
    - {Beer}: 3, {Nuts}: 3, {Diapers}: 4, {Eggs}: 3.
  - **2-itemsets:**
    - {Beer, Diapers}: 3
  - **3-itemsets:**
    - *None*
  - **4-itemsets:**
    - *None*
  - **5-itemsets:**
    - *None*

Basic Concepts: Association Rules

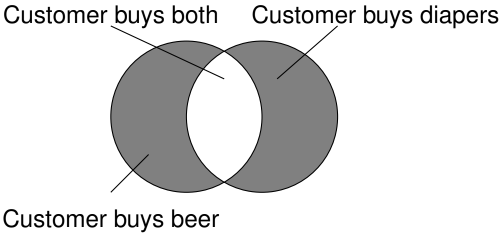
- **Implication of the form $A \implies B$:**
  - where $A \neq \emptyset$, $B \neq \emptyset$ and $A \cap B = \emptyset$.
- **Strong rule:**
  - Satisfies both min_sup and min_conf

$$\begin{aligned}
\text{support}(A \implies B) &= P(A \cup B), \\
\text{confidence}(A \implies B) &= P(B|A) \\
&= \frac{\text{support}(A \cup B)}{\text{support}(A)}.
\end{aligned}$$

  - I.e. confidence of rule can be easily derived from the support counts of A and $A \cup B$.
- **Association-rule mining:**
  - Find all frequent itemsets (with a length of at least 2).
  - Generate strong association rules from the frequent itemsets.

Association Rules - Example

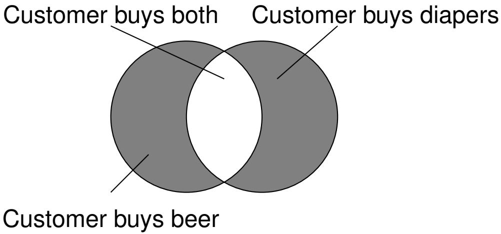
TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk



- **Thresholds:**
  - Set by the user.
  - In this example:
    - $min\_sup = 3$.
    - $min\_conf = 50\%$.
- **Reminder:**
  - Frequent itemset(s) with length > 2:
    - {Beer, Diapers}: 3
  - Already satisfy the `min_sup` threshold.

Association Rules - Example

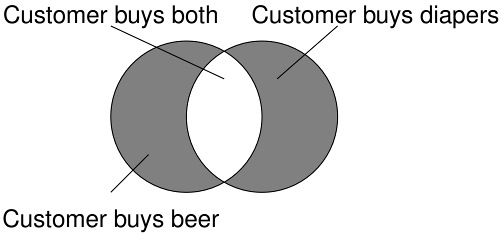
TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk



- **Thresholds:**
  - Set by the user.
  - In this example:
    - $min\_sup = 3$.
    - $min\_conf = 50\%$.
- **Reminder:**
  - Frequent itemset(s) with length > 2:
    - {Beer, Diapers}: 3
  - Already satisfy the `min_sup` threshold.
- **Association Rules:**
  - Beer $\Longrightarrow$ Diapers:
    -
  - Diapers $\Longrightarrow$ Beer:
    -

Association Rules - Example

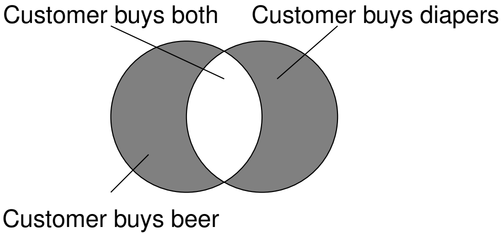
TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk



- **Thresholds:**
  - Set by the user.
  - In this example:
    - $min\_sup = 3$.
    - $min\_conf = 50\%$.
- **Reminder:**
  - Frequent itemset(s) with length > 2:
    - {Beer, Diapers}: 3
  - Already satisfy the `min_sup` threshold.
- **Association Rules:**
  - Beer $\Longrightarrow$ Diapers:
    - $P(Diapers|Beer) = \frac{3}{3} = 100\%$.
  - Diapers $\Longrightarrow$ Beer:
    -

Association Rules - Example

TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk



- **Thresholds:**
  - Set by the user.
  - In this example:
    - $min\_sup = 3$.
    - $min\_conf = 50\%$.
- **Reminder:**
  - Frequent itemset(s) with length > 2:
    - {Beer, Diapers}: 3
  - Already satisfy the `min_sup` threshold.
- **Association Rules:**
  - Beer $\implies$ Diapers:
    - $P(Diapers|Beer) = \frac{3}{3} = 100\%$.
  - Diapers $\implies$ Beer:
    - $P(Beer|Diapers) = \frac{3}{4} = 75\%$.

Closed Itemsets and Max-Itemsets

- **A long itemset contains a combinatorial number of sub-itemsets.**
  - E.g. $\{a_1, a_2, \ldots, a_{100}\}$ contains

$$\binom{100}{1} + \binom{100}{2} + \cdots + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \cdot 10^{30} \text{ sub-itemsets!}$$

  - **Solution:**
    - Mine closed itemsets and max-itemsets instead.

Closed Itemsets[2]

An itemset X is closed, if X is frequent and there exists no super-itemset $X \subset Y$ with the same support.

Max-Itemsets[3]

An itemset X is a max-itemset, if X is frequent and there exists no frequent super-itemset $X \subset Y$.

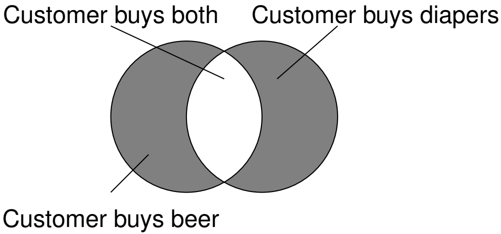
---

[2] N. Pasquier et al., "Discovering frequent closed itemsets for association rules," in *Proceedings of the 7th International Conference on Database Theory*, ser. ICDT '99, Berlin, Heidelberg: Springer-Verlag, 1999, pp. 398–416, ISBN: 3540654526

[3] R. J. Bayardo, "Efficiently mining long patterns from databases," *SIGMOD Rec.*, vol. 27, no. 2, pp. 85–93, Jun. 1998, ISSN: 0163-5808. DOI: 10.1145/276305.276313. [Online]. Available: https://doi.org/10.1145/276305.276313

Closed Itemsets and Max-Itemsets - Example

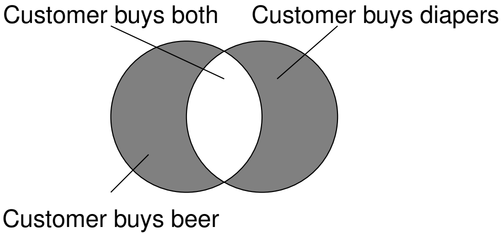
TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk



- **Reminder:**
  - **Frequent Itemsets:**
    - **1-itemsets:**
      {Beer}: 3, {Nuts}: 3, {Diapers}: 4, {Eggs}: 3
    - **2-itemsets:**
      {Beer, Diapers}: 3
- **Closed Itemsets:**
  - **1-itemsets:**
    -
  - **2-itemsets:**
    -
- **Max-Itemsets:**
  - **1-itemsets:**
    -
  - **2-itemsets:**
    -

Closed Itemsets and Max-Itemsets - Example

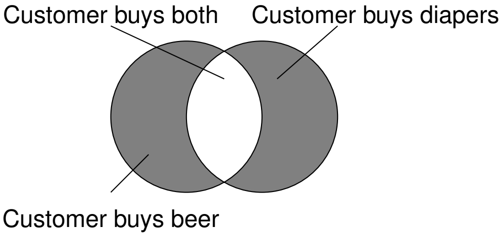
TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk



- **Reminder:**
  - **Frequent Itemsets:**
    - **1-itemsets:**
      {Beer}: 3, {Nuts}: 3, {Diapers}: 4, {Eggs}: 3
    - **2-itemsets:**
      {Beer, Diapers}: 3
- **Closed Itemsets:**
  - **1-itemsets:**
    - {Nuts}: 3, {Diapers}: 4, {Eggs}: 3
  - **2-itemsets:**
    -
- **Max-Itemsets:**
  - **1-itemsets:**
    -
  - **2-itemsets:**
    -

Closed Itemsets and Max-Itemsets - Example

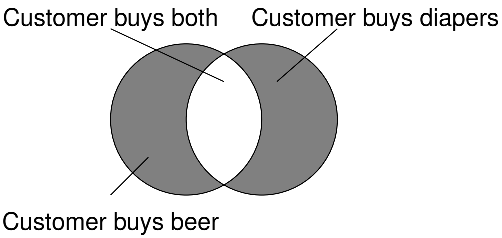
TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk



- **Reminder:**
  - **Frequent Itemsets:**
    - **1-itemsets:**
      {Beer}: 3, {Nuts}: 3, {Diapers}: 4, {Eggs}: 3
    - **2-itemsets:**
      {Beer, Diapers}: 3
- **Closed Itemsets:**
  - **1-itemsets:**
    - {Nuts}: 3, {Diapers}: 4, {Eggs}: 3
  - **2-itemsets:**
    - {Beer, Diapers}: 3
- **Max-Itemsets:**
  - **1-itemsets:**
    -
  - **2-itemsets:**
    -

Closed Itemsets and Max-Itemsets - Example

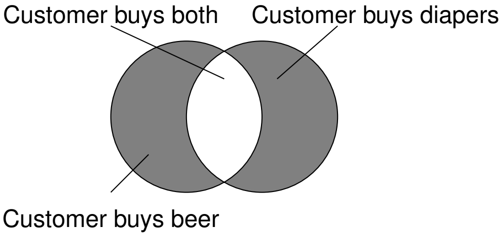
TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk



- **Reminder:**
  - **Frequent Itemsets:**
    - **1-itemsets:**
      {Beer}: 3, {Nuts}: 3, {Diapers}: 4, {Eggs}: 3
    - **2-itemsets:**
      {Beer, Diapers}: 3
- **Closed Itemsets:**
  - **1-itemsets:**
    - {Nuts}: 3, {Diapers}: 4, {Eggs}: 3
  - **2-itemsets:**
    - {Beer, Diapers}: 3
- **Max-Itemsets:**
  - **1-itemsets:**
    - {Nuts}: 3, {Eggs}: 3
  - **2-itemsets:**
    -

Closed Itemsets and Max-Itemsets - Example

TID	Items bought
10	Beer, Nuts, Diapers
20	Beer, Coffee, Diapers
30	Beer, Diapers, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diapers, Eggs, Milk



- **Reminder:**
  - **Frequent Itemsets:**
    - **1-itemsets:**
      {Beer}: 3, {Nuts}: 3, {Diapers}: 4, {Eggs}: 3
    - **2-itemsets:**
      {Beer, Diapers}: 3
- **Closed Itemsets:**
  - **1-itemsets:**
    - {Nuts}: 3, {Diapers}: 4, {Eggs}: 3
  - **2-itemsets:**
    - {Beer, Diapers}: 3
- **Max-Itemsets:**
  - **1-itemsets:**
    - {Nuts}: 3, {Eggs}: 3
  - **2-itemsets:**
    - {Beer, Diapers}: 3

Scalable Frequent-itemset Mining Methods

Friedrich-Alexander-Universität
Technische Fakultät

The Downward-closure Property

Any subset of a frequent itemset must also be frequent.

- **Example:**
  - If {Beer, Diapers, Nuts} is frequent, so is {Beer, Diapers}.
  - I.e. every transaction having {Beer, Diapers, Nuts} also contains {Beer, Diapers}.
- **Utilized by the major frequent-itemset mining algorithms:**
  - Apriori[4]
  - Frequent-pattern growth (FP-growth)[5]
  - etc ...

[4] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499, ISBN: 1558601538

[5] J. Han et al., "Mining frequent patterns without candidate generation," *SIGMOD Rec.*, vol. 29, no. 2, pp. 1–12, May 2000, ISSN: 0163-5808. DOI: 10.1145/335191.335372. [Online]. Available: https://doi.org/10.1145/335191.335372

Scalable Frequent-itemset Mining Methods

Apriori

The Apriori Pruning Principle[6][7]

If there is any itemset which is infrequent, its supersets should not be generated/tested!

- **The Apriori Algorithm - A Candidate Generation Approach:**[8]
  - Initially, scan DB once to get frequent 1-itemsets.
  - Generate length-$(k+1)$ candidate itemsets from length-k frequent itemsets.
  - Test the candidates against DB, discard those that are infrequent.
  - Terminate when no further candidate or frequent itemset can be generated.

[6] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499, ISBN: 1558601538

[7] H. Mannila et al., "Efficient algorithms for discovering association rules," in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, ser. AAAIWS'94, Seattle, WA: AAAI Press, 1994, pp. 181–192

[8] A complete pseudo-code can be found in the appendix.

Apriori Algorithm - Example

Friedrich-Alexander-Universität
Technische Fakultät

Database

TID	Items
10	A,C,D
20	B,C,E
30	A,B,C,E
40	B,E

$min\_sup = 2$

Apriori Algorithm - Example

Friedrich-Alexander-Universität
Technische Fakultät

Database

TID	Items
10	A,C,D
20	B,C,E
30	A,B,C,E
40	B,E

1st scan $\rightarrow$

C_1

Itemset	sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{D\}$	1
$\{E\}$	3

$min\_sup = 2$

Apriori Algorithm - Example

Database

TID	Items
10	A,C,D
20	B,C,E
30	A,B,C,E
40	B,E

1^{st} scan →

C_1

Itemset	sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{D\}$	1
$\{E\}$	3

prune →

L_1

Itemset	sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{E\}$	3

$min\_sup = 2$

Apriori Algorithm - Example

Database

TID	Items
10	A,C,D
20	B,C,E
30	A,B,C,E
40	B,E

1^{st} scan →

C_1

Itemset	sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{D\}$	1
$\{E\}$	3

prune →

L_1

Itemset	sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{E\}$	3

combine →

C_2

Itemset
$\{A, B\}$
$\{A, C\}$
$\{A, E\}$
$\{B, C\}$
$\{B, E\}$
$\{C, E\}$

$min\_sup = 2$

Apriori Algorithm - Example

Database

TID	Items
10	A,C,D
20	B,C,E
30	A,B,C,E
40	B,E

1^{st} scan →

C_1

Itemset	sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{D\}$	1
$\{E\}$	3

prune →

L_1

Itemset	sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{E\}$	3

combine →

C_2

Itemset
$\{A, B\}$
$\{A, C\}$
$\{A, E\}$
$\{B, C\}$
$\{B, E\}$
$\{C, E\}$

↓ 2^{nd} scan

C_2

Itemset	sup
$\{A, B\}$	1
$\{A, C\}$	2
$\{A, E\}$	1
$\{B, C\}$	2
$\{B, E\}$	3
$\{C, E\}$	2

$min\_sup = 2$

Apriori Algorithm - Example

Database

TID	Items
10	A,C,D
20	B,C,E
30	A,B,C,E
40	B,E

$min\_sup = 2$

$\xrightarrow{1^{st} \text{ scan}}$

C_1

Itemset	sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{D\}$	1
$\{E\}$	3

$\xrightarrow{\text{prune}}$

L_1

Itemset	sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{E\}$	3

$\xrightarrow{\text{combine}}$

C_2

Itemset
$\{A, B\}$
$\{A, C\}$
$\{A, E\}$
$\{B, C\}$
$\{B, E\}$
$\{C, E\}$

$\xrightarrow{2^{nd} \text{ scan}}$

C_2

Itemset	sup
$\{A, B\}$	1
$\{A, C\}$	2
$\{A, E\}$	1
$\{B, C\}$	2
$\{B, E\}$	3
$\{C, E\}$	2

$\xrightarrow{\text{prune}}$

L_2

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

Apriori Algorithm - Example

Database

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

prune →

L_1

Itemset	sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{E\}$	3

combine →

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

prune

L_2

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

combine ←

C_3

Itemset
$\{B, C, E\}$

$min\_sup = 2$

Apriori Algorithm - Example

Database

TID	Items
10	A,C,D
20	B,C,E
30	A,B,C,E
40	B,E

1^{st} scan →

C_1

Itemset	sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{D\}$	1
$\{E\}$	3

prune →

L_1

Itemset	sup
$\{A\}$	2
$\{B\}$	3
$\{C\}$	3
$\{E\}$	3

combine →

C_2

Itemset
$\{A, B\}$
$\{A, C\}$
$\{A, E\}$
$\{B, C\}$
$\{B, E\}$
$\{C, E\}$

2^{nd} scan ↓

C_2

Itemset	sup
$\{A, B\}$	1
$\{A, C\}$	2
$\{A, E\}$	1
$\{B, C\}$	2
$\{B, E\}$	3
$\{C, E\}$	2

prune

L_2

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

← combine

C_3

Itemset
$\{B, C, E\}$

← 3^{rd} scan

L_3

Itemset	sup
$\{B, C, E\}$	2

$min\_sup = 2$

Apriori Algorithm - Candidate Generation

Friedrich-Alexander-Universität
Technische Fakultät

Follow the Apriori Pruning Principle!

If **any** subset of an itemset you wish to generate is infrequent, it is **not** a valid candidate!

- **Example**[9]**:**
  - The itemset $\{A, B, C\}$ is **not** a valid candidate:
    - Frequent Subsets: $\{A\}, \{B\}, \{C\}, \{A, C\}, \{B, C\}$
    - Infrequent Subset: $\{A, B\}$
- **How to generate candidates?**
  - Step 1: Join all frequent k-itemsets that have $k - 1$ items in common.
    - E.g. $\{A, B\}$ and $\{A, C\}$ can be joined to form $\{A, B, C\}$.
  - Step 2: Prune all combinations that have infrequent subsets.
    - E.g. $\{A, B, C\}$ has to be pruned, because $\{A, B\}$ is infrequent.

[9] Based on the previous slide/example

Improvements

- **Apriori is pretty inefficient:**
  - Multiple scans of transaction database.
  - Huge number of candidates.
  - Support counting for candidates is laborious.
- **Many improvements have been proposed.**
- **Some examples:**
  - Reducing the passes of database scans:
    - Partitioning[10]
    - Dynamic itemset counting[11]
  - Shrinking the number of candidates:
    - Hashing[12]

[10] e.g. A. Savasere et al., "An efficient algorithm for mining association rules in large databases," in *Proceedings of the 21th International Conference on Very Large Data Bases*, ser. VLDB '95, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 432–444, ISBN: 1558603794

[11] e.g. S. Brin et al., "Dynamic itemset counting and implication rules for market basket data," *SIGMOD Rec.*, vol. 26, no. 2, pp. 255–264, Jun. 1997, ISSN: 0163-5808. DOI: `10.1145/253262.253325`. [Online]. Available: `https://doi.org/10.1145/253262.253325`

[12] e.g. J. S. Park et al., "An effective hash-based algorithm for mining association rules," *SIGMOD Rec.*, vol. 24, no. 2, pp. 175–186, May 1995, ISSN: 0163-5808. DOI: `10.1145/568271.223813`. [Online]. Available: `https://doi.org/10.1145/568271.223813`

Improvements - Partitioning

Partitioning: The Basic Idea

Any itemset that is potentially frequent in the whole database must be frequent in at least one of the partitions of the database.

- **Method: Scan the database twice**
  - Scan 1: Partition database and find the local frequent itemsets:
    - $\text{min\_sup}_i = \text{min\_sup}[\%] \cdot |\sigma \text{DB}_i|$.
  - Scan 2: Use the local frequent itemsets to check for global frequent itemsets:
    - Only itemsets that are frequent in at least one partition are checked.

$$\text{DB}_1 + \text{DB}_2 + \cdots + \text{DB}_k = \text{DB}$$

$$\text{sup}(i) \leq |\sigma \text{DB}|$$

$$\text{sup}_1(i) \leq |\sigma \text{DB}_1| \quad \text{sup}_2(i) \leq |\sigma \text{DB}_2| \quad \text{sup}_k(i) \leq |\sigma \text{DB}_k|$$

Improvements - Dynamic Itemset Counting (I)

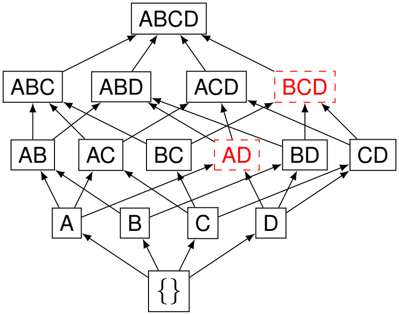
Friedrich-Alexander-Universität
Technische Fakultät

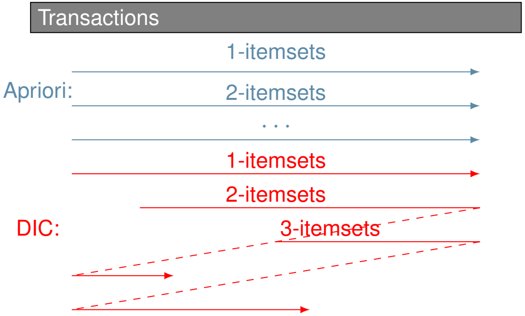
Dynamic Itemset Counting (DIC): The Basic Idea

Itemset frequency counting starts once all subsets are confirmed to be frequent.

- **Candidate itemsets are added at different points during a scan:**
  - New candidate itemsets can be added at any start point during a scan.
    - E.g. if A and B are already found to be frequent, AB are also counted from that starting point on.
  - Uses the count-so-far as the lower bound of the actual count.
  - If count-so-far passes minimum support, itemset is added to frequent-itemset collection.
  - Can then be used to generate even longer candidates.

Improvements - Dynamic Itemset Counting (II)





Hashing: The Basic Idea

Itemsets are hashed into buckets, and during the first scan, only the occurrences of each bucket are counted.

- **A k-itemset whose corresponding hashing-bucket count is below the threshold cannot be frequent.**
  - Candidates: a, b, c, d, e.
  - While scanning DB for frequent 1-itemsets, create hash entries for 2-itemsets:
    - $\{ab, ad, ae\}$
    - $\{bd, be, de\}$
    - ...
  - 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.

Hash table:

count	itemsets
35	$\{ab, ad, ae\}$
88	$\{bd, be, de\}$
$\vdots$	$\vdots$
102	$\{yz, qs, wt\}$

Scalable Frequent-itemset Mining Methods

FP-growth

FP-growth

- **Apriori:**
  - Breadth-first (i.e., level-wise) search.
  - Candidate generation and test.
    - Often generates a huge number of candidates.
- **FP-growth:**
  - Depth-first search.
  - Avoid explicit candidate generation.

FP-growth: All Frequent Itemsets in Only Two Scans

FP-growth employs a tree-based structure to identify all frequent itemsets in a dataset using two scans.

FP-growth - Algorithm

- **Steps of FP-growth:**

TID	Items bought
100	f,a,c,d,g,i,m,p
200	a,b,c,f,l,m
300	b,f,h,j,o,w
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).

TID	Items bought
100	f,a,c,d,g,i,m,p
200	a,b,c,f,l,m
300	b,f,h,j,o,w
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).

TID	Items bought
100	f,a,c,d,g,i,m,p
200	a,b,c,f,l,m
300	b,f,h,j,o,w
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

Frequent 1-itemsets:

Itemset	Support
{f}	4
{a}	3
{c}	4
{m}	3
{p}	3
{b}	3

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.

TID	Items bought
100	f,a,c,d,g,i,m,p
200	a,b,c,f,l,m
300	b,f,h,j,o,w
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

Frequent 1-itemsets:

Itemset	Support
{f}	4
{a}	3
{c}	4
{m}	3
{p}	3
{b}	3

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.

TID	Items bought
100	f,a,c,d,g,i,m,p
200	a,b,c,f,l,m
300	b,f,h,j,o,w
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p

Frequent 1-itemsets:

Itemset	Support
{f}	4
{a}	3
{c}	4
{m}	3
{p}	3
{b}	3

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,a,c,d,g,i,m,p
200	a,b,c,f,l,m
300	b,f,h,j,o,w
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p

FP-growth - Algorithm

Friedrich-Alexander-Universität
Technische Fakultät

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,a,c,d,g,i,m,p
200	a,b,c,f,l,m
300	b,f,h,j,o,w
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,c,a,m,p
200	a,b,c,f,l,m
300	b,f,h,j,o,w
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p

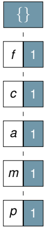
FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,c,a,m,p
200	a,b,c,f,l,m
300	b,f,h,j,o,w
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p



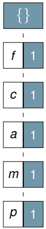
FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,c,a,m,p
200	a,b,c,f,l,m
300	b,f,h,j,o,w
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p



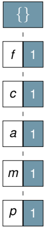
FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,c,a,m,p
200	f,c,a,b,m
300	b,f,h,j,o,w
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p



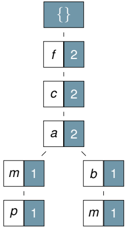
FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,c,a,m,p
200	f,c,a,b,m
300	b,f,h,j,o,w
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p



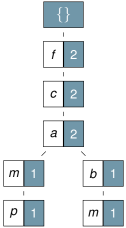
FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,c,a,m,p
200	f,c,a,b,m
300	b,f,h,j,o,w
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p



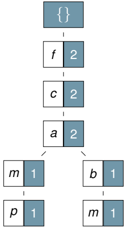
Friedrich-Alexander-Universität
Technische Fakultät

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,c,a,m,p
200	f,c,a,b,m
300	f,b
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p



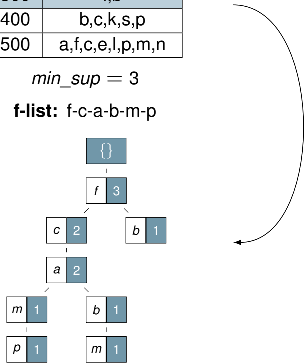
FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,c,a,m,p
200	f,c,a,b,m
300	f,b
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p



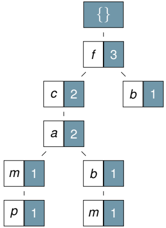
FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,c,a,m,p
200	f,c,a,b,m
300	f,b
400	b,c,k,s,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p



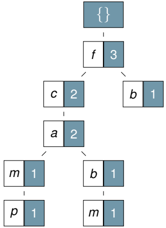
FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,c,a,m,p
200	f,c,a,b,m
300	f,b
400	c,b,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p



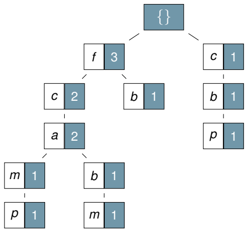
FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,c,a,m,p
200	f,c,a,b,m
300	f,b
400	c,b,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p



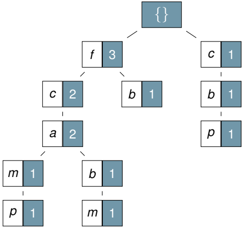
FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,c,a,m,p
200	f,c,a,b,m
300	f,b
400	c,b,p
500	a,f,c,e,l,p,m,n

$min\_sup = 3$

f-list: f-c-a-b-m-p



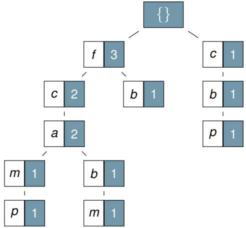
FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,c,a,m,p
200	f,c,a,b,m
300	f,b
400	c,b,p
500	f,c,a,m,p

$min\_sup = 3$

f-list: f-c-a-b-m-p



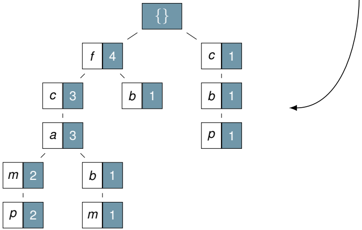
FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.

TID	Items bought
100	f,c,a,m,p
200	f,c,a,b,m
300	f,b
400	c,b,p
500	f,c,a,m,p

$min\_sup = 3$

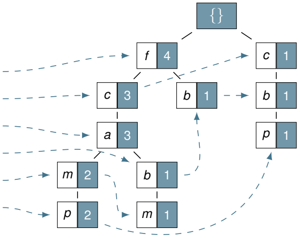
f-list: f-c-a-b-m-p



FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*

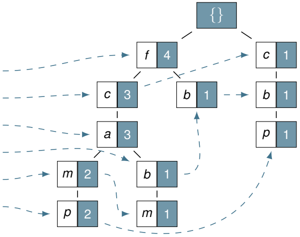
Item	Sum
f	4
c	4
a	3
b	3
m	3
p	3



FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**

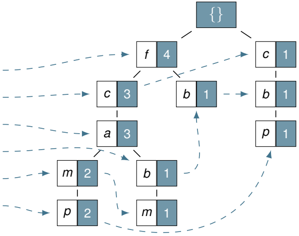
Item	Sum
f	4
c	4
a	3
b	3
m	3
p	3



FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.

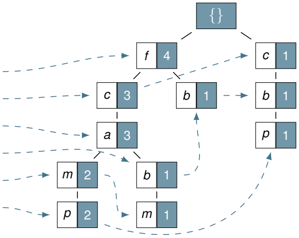
Item	Sum
f	4
c	4
a	3
b	3
m	3
p	3



Friedrich-Alexander-Universität
Technische Fakultät

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.

Item	Sum
f	4
c	4
a	3
b	3
m	3
p	3

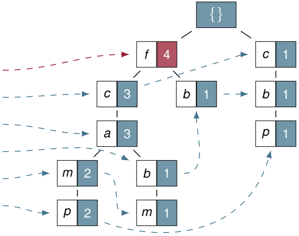


Item	Conditional Pattern Base
f	
c	
a	
b	
m	
p	

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       (Also comes with a ***header table****)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.

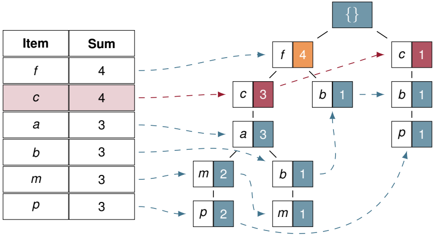
Item	Sum
f	4
c	4
a	3
b	3
m	3
p	3



Item	Conditional Pattern Base
f	-
c	
a	
b	
m	
p	

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.

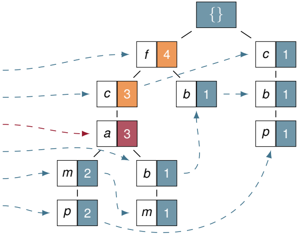


Item	Conditional Pattern Base
f	-
c	f:3
a	
b	
m	
p	

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     ⇒ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**

     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.

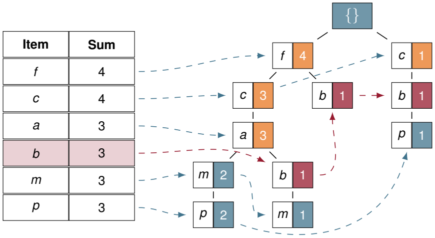
Item	Sum
f	4
c	4
a	3
b	3
m	3
p	3



Item	Conditional Pattern Base
f	-
c	f:3
a	f,c:3
b	
m	
p	

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.



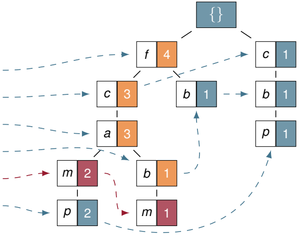
Item	Conditional Pattern Base
f	-
c	f:3
a	f,c:3
b	f,c,a:1, f:1, c:1
m	
p	

FP-growth - Algorithm

Friedrich-Alexander-Universität
Technische Fakultät

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.

Item	Sum
f	4
c	4
a	3
b	3
m	3
p	3

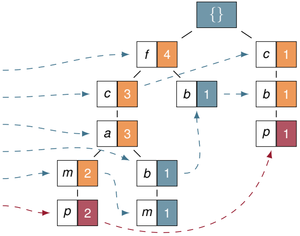


Item	Conditional Pattern Base
f	-
c	f:3
a	f,c:3
b	f,c,a:1, f:1, c:1
m	f,c,a:2, f,c,a,b:1
p	

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.

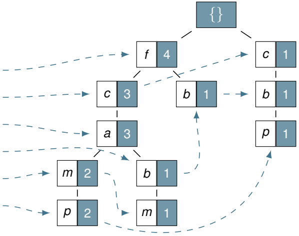
Item	Sum
f	4
c	4
a	3
b	3
m	3
p	3



Item	Conditional Pattern Base
f	-
c	f:3
a	f,c:3
b	f,c,a:1, f:1, c:1
m	f,c,a:2, f,c,a,b:1
p	f,c,a,m:2, c,b:1

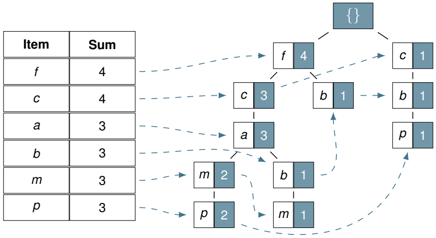
- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     ⇒ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     - 4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     - 4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.

Item	Sum
f	4
c	4
a	3
b	3
m	3
p	3



Item	Conditional Pattern Base
f	-
c	f:3
a	f,c:3
b	f,c,a:1, f:1, c:1
m	f,c,a:2, f,c,a,b:1
p	f,c,a,m:2, c,b:1

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     ⇒ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     - 4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     - 4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.



Item	Conditional Pattern Base
f	-
c	f:3
a	f,c:3
b	f,c,a:1, f:1, c:1
m	f,c,a:2, f,c,a,b:1
p	f,c,a,m:2, c,b:1

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.

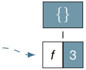
Condition	Pattern Base
c	f:3

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     - 4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     - 4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.

Condition	Pattern Base
c	f:3

Item	Sum
$(c,)f$	3

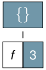


FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
c	f:3

Item	Sum
$(c,)f$	3

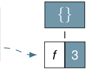


FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
c	f:3

Item	Sum
$(c,)f$	3

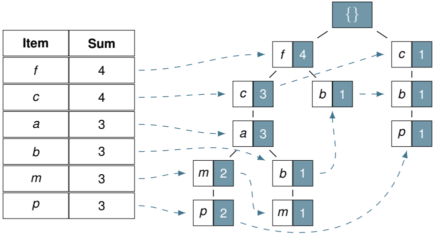


Item	Conditional Pattern Base
(c,) f	-

FP-growth - Algorithm

Friedrich-Alexander-Universität
Technische Fakultät

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.



Item	Conditional Pattern Base
f	-
c	f:3
a	f,c:3
b	f,c,a:1, f:1, c:1
m	f,c,a:2, f,c,a,b:1
p	f,c,a,m:2, c,b:1

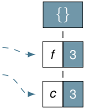
FP-growth - Algorithm

Friedrich-Alexander-Universität
Technische Fakultät

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     - 4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     - 4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     - 4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
a	f,c:3

Item	Sum
$(a,)f$	3
$(a,)c$	3



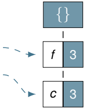
Item	Conditional Pattern Base
(a,) f	-
(a,) c	f:3

Friedrich-Alexander-Universität
Technische Fakultät

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
a	f,c:3

Item	Sum
$(a,)f$	3
$(a,)c$	3



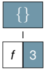
Item	Conditional Pattern Base
(a,) f	-
(a,) c	f:3

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
a,c	f:3

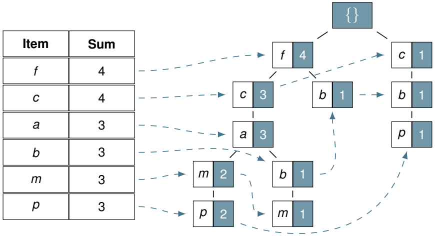
Item	Sum
$(a, c,)f$	3



Item	Conditional Pattern Base
(a,c,) f	-

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     ⇒ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     - 4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     - 4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     - 4.3 Perform **4.** for each cond. FP-tree.



Item	Conditional Pattern Base
f	-
c	f:3
a	f,c:3
b	f,c,a:1, f:1, c:1
m	f,c,a:2, f,c,a,b:1
p	f,c,a,m:2, c,b:1

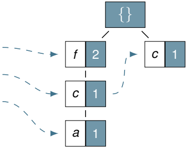
FP-growth - Algorithm

Friedrich-Alexander-Universität
Technische Fakultät

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
b	f,c,a:1, f:1, c:1

Item	Sum
$(b,)f$	2
$(b,)c$	2
$(b,)a$	1



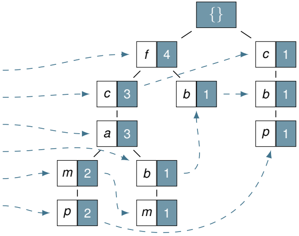
Item	Conditional Pattern Base
No frequent items	

FP-growth - Algorithm

Friedrich-Alexander-Universität
Technische Fakultät

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     - 4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     - 4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     - 4.3 Perform **4.** for each cond. FP-tree.

Item	Sum
f	4
c	4
a	3
b	3
m	3
p	3

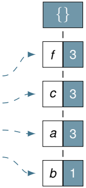


Item	Conditional Pattern Base
f	-
c	f:3
a	f,c:3
b	f,c,a:1, f:1, c:1
m	f,c,a:2, f,c,a,b:1
p	f,c,a,m:2, c,b:1

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
m	f,c,a:2, f,c,a,b:1

Item	Sum
$(m,)f$	3
$(m,)c$	3
$(m,)a$	3
$(m,)b$	1



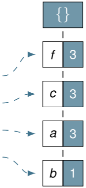
Item	Conditional Pattern Base
(m,) f	-
(m,) c	f:3
(m,) a	f,c:3

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
m	f,c,a:2, f,c,a,b:1

Item	Sum
$(m,)f$	3
$(m,)c$	3
$(m,)a$	3
$(m,)b$	1



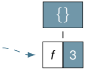
Item	Conditional Pattern Base
(m,) f	-
(m,) c	f:3
(m,) a	f,c:3

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       (Also comes with a ***header table****)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
m,c	f:3

Item	Sum
$(m, c,)f$	3



Item	Conditional Pattern Base
(m,c,) f	-

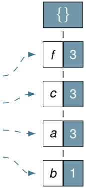
FP-growth - Algorithm

Friedrich-Alexander-Universität
Technische Fakultät

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
m	f,c,a:2, f,c,a,b:1

Item	Sum
$(m,)f$	3
$(m,)c$	3
$(m,)a$	3
$(m,)b$	1



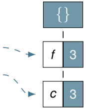
Item	Conditional Pattern Base
(m,) f	-
(m,) c	f:3
(m,) a	f,c:3

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     - 4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     - 4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     - 4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
m,a	f,c:3

Item	Sum
$(m, a,)f$	3
$(m, a,)c$	3



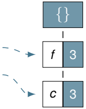
Item	Conditional Pattern Base
(a,) f	-
(m,a,) c	f:3

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
m,a	f,c:3

Item	Sum
$(m, a,)f$	3
$(m, a,)c$	3



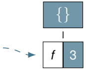
Item	Conditional Pattern Base
(a,) f	-
(m,a,) c	f:3

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
m,a,c	f:3

Item	Sum
$(m, a, c,)f$	3

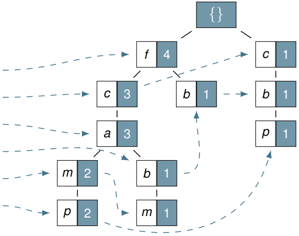


Item	Conditional Pattern Base
(m,a,c,) f	-

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     ⇒ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.

Item	Sum
f	4
c	4
a	3
b	3
m	3
p	3



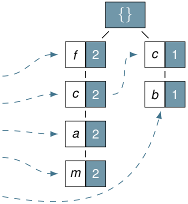
Item	Conditional Pattern Base
f	-
c	f:3
a	f,c:3
b	f,c,a:1, f:1, c:1
m	f,c,a:2, f,c,a,b:1
p	f,c,a,m:2, c,b:1

Friedrich-Alexander-Universität
Technische Fakultät

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     - 4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     - 4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     - 4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
p	f,c,a,m:2, c,b:1

Item	Sum
$(p,)f$	2
$(p,)c$	3
$(p,)a$	2
$(p,)m$	2
$(p,)b$	1



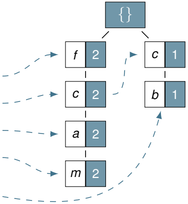
Item	Conditional Pattern Base
(p,) c	f:2

FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     - 4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     - 4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     - 4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
p	f,c,a,m:2, c,b:1

Item	Sum
$(p,)f$	2
$(p,)c$	3
$(p,)a$	2
$(p,)m$	2
$(p,)b$	1



Item	Conditional Pattern Base
(p,) c	f:2

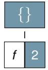
FP-growth - Algorithm

Friedrich-Alexander-Universität
Technische Fakultät

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.

Condition	Pattern Base
p,c	f:2

Item	Sum
$(p, c,)f$	2



Item	Conditional Pattern Base
No frequent items	

FP-growth - Algorithm

Friedrich-Alexander-Universität
Technische Fakultät

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       *(Also comes with a **header table**)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.
  5. Collect the **frequent itemsets**.

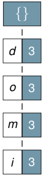
FP-growth - Algorithm

- **Steps of FP-growth:**
  1. Find frequent 1-itemsets (**1st scan**).
  2. Put them in a frequency-descending list.
     $\Rightarrow$ **f-list**.
  3. Perform the **2nd scan:**
     - Sort and filter the items in each tuple.
     - Construct the initial **FP-tree**.
       (Also comes with a ***header table****)*
  4. Start the FP-tree **recursion:**
     4.1 Determine the **conditional pattern base** (prefix paths) for each frequent item in the header table.
     4.2 Build a **conditional FP-tree** for each non-empty conditional pattern base.
     4.3 Perform **4.** for each cond. FP-tree.
  5. Collect the **frequent itemsets**.

Source	Frequent Itemset(s)
Initial FP-tree	{f}, {c}, {a}, {b}, {m}, {p}
c's cond. FP-tree	{c,f}
a's cond. FP-tree	{a,f}, {a,c}
b's cond. FP-tree	-
a,c's cond. FP-tree	{a,c,f}
m's cond. FP-tree	{m,f},{m,c},{m,a}
m,c's cond. FP-tree	{m,c,f}
m,a's cond. FP-tree	{m,a,f},{m,a,c}
m,a,c's cond. FP-tree	{m,a,c,f}
p's cond. FP-tree	{p,c}
p,c's cond. FP-tree	-

FP-growth - Special Case(s) (I)

- **Special Case:** A single branch FP-tree
  - No recursion required.[13]
  - Frequent itemsets can directly be generated in one shot.



$\rightarrow$

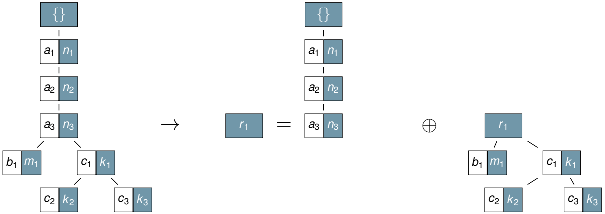
Frequent Itemsets:

- {d}, {o}, {m}, {i}
- {d,o}, {d,m}, {d,i}
- {o,m}, {o,i}
- {m,i}
- {d,o,m}, {d,o,i}, {d,m,i}
- {o,m,i}
- {d,o,m,i}

[13] A simple FP-tree recursion will still work, but is not as efficent as one with a special case optimization.

FP-growth - Special Case(s) (II)

- **Special Case:** A single prefix path in a FP-tree
  - Reduction of the single prefix path into one node.
  - Concatenation of the mining results of the two parts.
  - Both parts can be mined in parallel.



Advantages of the FP-growth Approach

- Can be **parallelized**:
  - Different conditional pattern bases can be mined in parallel.
- **No** candidate generation
- Only **two** scans of the database.
- The **FP-tree** structure is **compact**:
  - Compressed representation of the database.

Scalable Frequent-itemset Mining Methods

Other Approaches

- **Many other approaches exist.**
- **Often with a specialization. E.g.:**
  - **ECLAT**[14]**:** Mining in the vertical data format.
  - **CLOSET**[15]**:** Mining closed itemsets.
  - **MaxMiner**[16]**:** Mining max-itemsets.

[14] M. J. Zaki et al., "Parallel algorithms for discovery of association rules," *Data Min. Knowl. Discov.*, vol. 1, no. 4, pp. 343–373, 1997. DOI: 10.1023/A:1009773317876. [Online]. Available: https://doi.org/10.1023/A:1009773317876

[15] J. Wang et al., "CLOSET+: searching for the best strategies for mining frequent closed itemsets," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, L. Getoor et al., Eds., ACM, 2003, pp. 236–245. DOI: 10.1145/956750.956779. [Online]. Available: https://doi.org/10.1145/956750.956779

[16] R. J. Bayardo, "Efficiently mining long patterns from databases," *SIGMOD Rec.*, vol. 27, no. 2, pp. 85–93, Jun. 1998, ISSN: 0163-5808. DOI: 10.1145/276305.276313. [Online]. Available: https://doi.org/10.1145/276305.276313

ECLAT

- **Vertical format:** $t(AB) = \{T_{11}, T_{25}, \ldots\}$
  - Tid-list: list of transaction ids containing an itemset.
- **Deriving frequent itemsets based on vertical intersections.**
  - $t(X) = t(Y)$: X and Y always happen together.
  - $t(X) \implies t(Y)$: transaction having X always has Y.
- **Using diffset to accelerate mining.**
  - Only keep track of differences of tids.
  - $t(X) = \{T_1, T_2, T_3\}$, $t(XY) = \{T_1, T_3\}$.
  - Diffset $(XY, X) = \{T_2\}$.

CLOSET (I)

- **F-list:** List of all frequent items in **support-ascending**[a] order.
  - f-list: d-a-f-e-c.
- **Divide search space.**
  - Itemsets having d.
  - Itemsets having d but not a, etc.
- **Find closed itemsets recursively.**
  - Every transaction having d also has *cfa* $\Longrightarrow$ *cfad* is a closed itemset.

TID	Items
10	a,c,d,e,f
20	a,b,e
30	c,e,f
40	a,c,d,f
50	c,e,f

[a] Note: this is the exact reverse of the f-list ordering in FP-growth.

CLOSET (II)

- **Itemset merging:.**
  - If Y appears in each occurrence of X, then Y is merged with X.
- **Sub-itemset pruning:**
  - If $X \subset Y$ and $\sup(X) = \sup(Y)$, X and all of X's descendants in the set enumeration tree can be pruned.
- **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

- **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
- **Potential max-itemsets: ABCDE, BCDE, CDE.**
- **Since BCDE is a max-itemset, no need to check BCD, BDE, CDE in later scan.**

TID	Items
10	A,B,C,D,E
20	B,C,D,E
30	A,C,D,F

Generating Association Rules

Generating Association Rules

- **Once frequent itemsets from transactions in database D found:**
  - Generate strong association rules from them,
    Where "strong" = satisfying both minimum support and minimum confidence.

$$\text{confidence}(A \implies B) = P(B|A) = \frac{\text{support}(A \implies B)}{\text{support}(A)}$$

- **For each frequent itemset l:**
  - Generate all **nonempty subsets** of l.
- **For every s in l:**
  - Output the rule $s \implies (l - s)$, if
  - min_sup is satisfied, because only frequent itemsets used.

Which Patterns are Interesting?

Interesting Patterns - Example

Friedrich-Alexander-Universität
Technische Fakultät

TID	Items
1	Apple, Cereal
2	Bread, Mango, Cereal
3	Cereal, Bread
4	Bread
...	...

Interesting Patterns - Example

TID	Items
1	Apple, Cereal
2	Bread, Mango, Cereal
3	Cereal, Bread
4	Bread
...	...

$\rightarrow$

	Bread	No Bread	Sum (Row)
Cereal	2000	1750	3750
No Cereal	1000	250	1250
Sum (Col.)	3000	2000	5000

Interesting Patterns - Example

TID	Items
1	Apple, Cereal
2	Bread, Mango, Cereal
3	Cereal, Bread
4	Bread
...	...

$\rightarrow$

	Bread	No Bread	Sum (Row)
Cereal	2000	1750	3750
No Cereal	1000	250	1250
Sum (Col.)	3000	2000	5000

Is Bread $\Longrightarrow$ Cereal a good rule?

Interesting Patterns - Example

TID	Items
1	Apple, Cereal
2	Bread, Mango, Cereal
3	Cereal, Bread
4	Bread
...	...

$\rightarrow$

	Bread	No Bread	Sum (Row)
Cereal	2000	1750	3750
No Cereal	1000	250	1250
Sum (Col.)	3000	2000	5000

Is Bread $\implies$ Cereal a good rule?

- **Support:** $2000/5000 = 40\%$.
- **Confidence:** $2000/3000 = 66.7\%$.

Interesting Patterns - Example

Friedrich-Alexander-Universität
Technische Fakultät

TID	Items
1	Apple, Cereal
2	Bread, Mango, Cereal
3	Cereal, Bread
4	Bread
...	...

$\rightarrow$

	Bread	No Bread	Sum (Row)
Cereal	2000	1750	3750
No Cereal	1000	250	1250
Sum (Col.)	3000	2000	5000

Is Bread $\Longrightarrow$ Cereal a good rule?

- **Support:** $2000/5000 = 40\%$.
- **Confidence:** $2000/3000 = 66.7\%$.
- **Problem:**
  - Overall 75% of transactions contain cereal.
    $\Rightarrow$ If bread is present, the likelihood of cereal is actually lower (66.7%).
  - **Misleading due to negative correlation.**

Interesting Patterns - Lift

- **Idea:** Check association rules for positive correlation.
- **Interestingness measure:** Lift

$$\mathrm{Lift}(A, B) = \frac{P(A \cup B)}{P(A)P(B)}$$

  - **Independence:** $\mathrm{Lift}(A, B) = 1$.
  - **Positive correlation:** $\mathrm{Lift}(A, B) > 1$.
  - **Negative correlation:** $\mathrm{Lift}(A, B) < 1$.
- **In our example:**

$$\mathrm{Lift}(\mathrm{Bread}, \mathrm{Cereal}) = \frac{2000/5000}{3000/5000 \cdot 3750/5000} = 0.89$$

Interesting Patterns - χ^2-Test

- With a small trick, we can also use the χ^2**-test**[17].
- **In our example:**

	Bread	No Bread	Sum (Row)
Cereal	2000 (2250)	1750 (1500)	3750
No Cereal	1000 (750)	250 (500)	1250
Sum (Col.)	3000	2000	5000

$$\chi^2 = \frac{(2000-2250)^2}{2250} + \frac{(1750-1500)^2}{1500} + \frac{(1000-750)^2}{750} + \frac{(250-500)^2}{500} = 277.78$$

Interpretation

- Lookup in the χ^2-table with *df* $= (2-1)(2-1) = 1$ and $\alpha = 0.005$ gives 7.879
  $\Rightarrow$ Bread and Cereal **are correlated**.
- The observed value of Bread and Cereal is 2000, while the expected value is 2250.
  $\Rightarrow$ Hints at a **negative** correlation.

[17] Known from KDDmUe - Lecture 4: Data Preprocessing.

Interesting Patterns - Null-Invariance

Null-Transaction

- A transaction that does not contain any of the itemsets being examined.
- Can outweigh the number of individual itemsets.

Null-Invariance

- A measure is null-invariant, if its value is free from the influence of null-transactions.

- We also want interestingness measures that are **null-invariant**.
  - Lift and χ^2 are **not** null-invariant.
  - We will take a closer look at the **Kulczynski measure** (Kulc) and the **Imbalance Ratio** (IR) as examples for null-invariant measures[18].

[18] The appendix also contains a list of 20+ measures (some null-invariant, some not). This list is not exam relevant.

Interesting Patterns - Kulczynski Measure

- **Kulczynski Measure:**

$$\mathrm{Kulc}(A, B) = \frac{\sup(AB)}{2}\left(\frac{1}{\sup(A)} + \frac{1}{\sup(B)}\right)$$

  - **Interesting rule:** $\mathrm{Kulc}(A, B)$ close to 0 or 1.
  - **(Potentially) not very interesting rule:** $\mathrm{Kulc}(A, B)$ close to 0.5.

- **In our example:**

$$\mathrm{Kulc}(\mathrm{Bread}, \mathrm{Cereal}) = \frac{2000}{2}\left(\frac{1}{3000} + \frac{1}{3750}\right) = 0.6$$

Interesting Patterns - Imbalance Ratio

- **Imbalance Ratio:**

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

  - **(Very) balanced rule:** $\mathrm{IR}(A, B)$ close to 0.
  - **(Very) unbalanced rule:** $\mathrm{IR}(A, B)$ close to 1.

- **In our example:**

$$\mathrm{IR}(\text{Bread}, \text{Cereal}) = \frac{|3000 - 3750|}{3000 + 3750 - 2000} \approx 0.16$$

Summary

Summary

Friedrich-Alexander-Universität
Technische Fakultät

- **Basic concepts:**
  - Association rules.
  - Support-confidence framework.
  - Closed and max-itemsets.
- **Scalable frequent-itemset-mining methods:**
  - Apriori:
    - Candidate generation & test.
  - FP-growth:
    - Only two scans of the database.
  - Other approaches:
    - ECLAT, CLOSET, . . .
- **Association rules generated from frequent itemsets.**
- **Which patterns are interesting?**
  - Pattern-evaluation methods.

Any questions about this chapter?

Ask them now or ask them later in our forum:

https://www.studon.fau.de/studon/goto.php?target=lcode_0LYeD79h

Appendix

Apriori Algorithm - Pseudo Code

C_k: candidate itemsets of size k
L_k: frequent itemsets of size k

$L_1 = \{\text{frequent items}\};$

for $(k = 1; L_k \neq \emptyset; k{+}{+})$ **do begin**
    $C_{k+1} =$ 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;
    $L_{k+1} =$ candidates in C_{k+1} with min_sup;
end;
return $\bigcup_k L_k$;

Other Improved Mining Methods

- **AFOPT**[19]
  - A "push-right" method for mining condensed frequent-pattern (CFP) tree.
- **Carpenter**[20]
  - Mine datasets with small rows but numerous columns.
  - Construct a row-enumeration tree for efficient mining.
- **FP-growth+**[21]
  - Efficiently using prefix-trees in mining frequent itemsets.
- **TD-Close**[22]

[19] G. Liu et al., “Afopt: An efficient implementation of pattern growth approach.,” in *FIMI*, 2003, pp. 1–10
[20] F. Pan et al., “Carpenter: Finding closed patterns in long biological datasets,” in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 637–642
[21] G. Grahne and J. Zhu, “Efficiently using prefix-trees in mining frequent itemsets,” in *FIMI ’03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA*, B. Goethals and M. J. Zaki, Eds., ser. CEUR Workshop Proceedings, vol. 90, CEUR-WS.org, 2003. [Online]. Available: http://ceur-ws.org/Vol-90/grahne.pdf
[22] H. Liu et al., “Mining interesting patterns from very high dimensional data: A top-down row enumeration approach,” in *Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA*, J. Ghosh et al., Eds., SIAM, 2006, pp. 282–293. DOI: 10.1137/1.9781611972764.25. [Online]. Available: https://doi.org/10.1137/1.9781611972764.25

Extension of Pattern-growth Mining (I)

- **Mining closed frequent itemsets and max-patterns.**
  - FPmax* and FPclose[23]
- **Mining sequential patterns.**
  - PrefixSpan[24], CloSpan[25], BIDE[26]
- **Mining graph patterns.**
  - gSpan[27]
- **Constraint-based mining of frequent patterns.**
  - gPrune[28]

[23] G. Grahne and J. Zhu, "Reducing the main memory consumptions of fpmax*and fpclose," in *Proc. Workshop Frequent Item Set Mining Implementations (FIMI 2004, Brighton, UK), Aachen, Germany*, 2004, p. 75

[24] J. Han et al., "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *proceedings of the 17th international conference on data engineering*, IEEE Piscataway, NJ, USA, 2001, pp. 215–224

[25] X. Yan et al., "Clospan: Mining: Closed sequential patterns in large datasets," in *Proceedings of the 2003 SIAM international conference on data mining*, SIAM, 2003, pp. 166–177

[26] J. Wang and J. Han, "Bide: Efficient mining of frequent closed sequences," in *Proceedings. 20th international conference on data engineering*, IEEE, 2004, pp. 79–90

[27] X. Yan and J. Han, "Gspan: Graph-based substructure pattern mining," in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, IEEE, 2002, pp. 721–724

[28] F. Zhu et al., "Gprune: A constraint pushing framework for graph pattern mining," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2007, pp. 388–400

Extension of Pattern-growth Mining (II)

- **Computing iceberg data cubes with complex measures.**
  - Star-cubing[29]
- **Pattern-growth-based clustering.**
  - MaPle[30]
- **Pattern-growth-based classification.**
  - Mining frequent and discriminative patterns[31]

[29] D. Xin et al., "Star-cubing: Computing iceberg cubes by top-down and bottom-up integration," in *Proceedings 2003 VLDB Conference*, Elsevier, 2003, pp. 476–487
[30] J. Pei et al., "Maple: A fast algorithm for maximal pattern-based clustering," in *Third ieee international conference on data mining*, IEEE, 2003, pp. 259–266
[31] H. Cheng et al., "Discriminative frequent pattern analysis for effective classification," in *2007 IEEE 23rd international conference on data engineering*, IEEE, 2006, pp. 716–725

Interestingness Measures - List (I)

- Over 20 **interestingness measures** have been proposed[32]:

symbol	name	range	formula
ψ	ψ-coefficient	$[-1, 1]$	$\frac{P(A,B)-P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
Q	Yule's Q	$[-1, 1]$	$\frac{P(A,B)P(\neg A,\neg B)-P(A,\neg B)P(\neg A,B)}{P(A,B)P(\neg A,\neg B)+P(A,\neg B)P(\neg A,B)}$
Y	Yule's Y	$[-1, 1]$	$\frac{\sqrt{P(A,B)P(\neg A,\neq B)}-\sqrt{P(A,\neg B)P(\neg A,B)}}{\sqrt{P(A,B)P(\neg A,\neg B)}+\sqrt{P(A,\neg B)P(\neg A,B)}}$
k	Cohen's k	$[-1, 1]$	$\frac{P(A,B)+P(\neg A,\neg B)-P(A)P(B)-P(\neg A)P(\neg B)}{1-P(A)P(B)-P(\neg A)P(\neg B)}$
PS	Patetsky-Shapiro's	$[-0.25, 0.25]$	$P(A, B) - P(A)P(B)$
F	Certainty factor	$[-1, 1]$	$\max(\frac{P(B\mid A)-P(B)}{1-P(B)}, \frac{P(A\mid B)-P(A)}{1-P(A)})$
AV	Added Value	$[-0.5, 1]$	$\max(P(B\mid A) - P(B), P(A\mid B) - P(A))$
K	Klosgen's Q	$[-0.33, 0.38]$	$\sqrt{P(A, B)}\max(P(B\mid A) - P(B), P(A\mid B) - P(A))$
g	Goodman-kruskal's	$[0, 1]$	$\frac{\sum_j \max_k P(A_j,B_k)+\sum_k \max_j P(A_j,B_k)-\max_j P(A_j)-\max_k P(B_k)}{2-\max_j P(A_j)-\max_k P(B_k)}$
M	Mutual information	$[0, 1]$	$\frac{\sum_i \sum_j P(A_i,B_j)\log\frac{P(A_i,B_j)}{P(A_i)P(B_j)}}{\min(-\sum_i P(A_i)\log P(A_i)\log P(A_i), -\sum_i P(B_i)\log P(B_i)\log P(B_i))}$

[32] P. Tan et al., "Selecting the right interestingness measure for association patterns," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, ACM, 2002, pp. 32–41. DOI: 10.1145/775047.775053. [Online]. Available: https://doi.org/10.1145/775047.775053

Interestingness Measures - List (II)

symbol	name	range	formula
J	J-Measure	$[0,1]$	$\max(P(A,B)\log\frac{P(B\mid A)}{P(B)} + P(\neg A,B)\log\frac{P(\neg A,B)}{P(\neg A)},$ $P(A,B)\log\frac{P(B\mid A)}{P(A)} + P(\neg A,B)\log\frac{P(\neg A\mid B)}{P(\neg B)})$
G	Gini index	$[0,1]$	$\max(P(A)[P(B\mid A)^2 + P(\neg B\mid A)^2]+$ $P(\neg A)[P(B\mid\neg A)^2 + P(\neg B\mid\neg A)^2]P(B)^2 - P(\neg B)^2,$ $P(B)[P(A\mid B)^2 + P(\neg A\mid B)^2]+$ $P(\neg B)[P(A\mid\neg B)^2 + P(\neg A\mid\neg B)^2] - P(A)^2 - P(\neg A)^2)$
s	Support	$[0,1]$	$P(A,B)$
c	Confidence	$[0,1]$	$\max(P(B\mid A), P(A\mid B))$
L	Laplace	$[0,1]$	$\max(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2})$
cos	Cosine	$[0,1]$	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
γ	coherence(Jaccard)	$[0,1]$	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
α	all_confidence	$[0,1]$	$\frac{P(A,B)}{\max(P(A),P(B))}$
o	Odds ratio	$[0,\infty)$	$\frac{P(A,B)P(\neg A,\neg B)}{P(\neg A,B)P(A,\neg B)}$
V	Conviction	$[0.5,\infty)$	$\max(\frac{P(A)P(\neg B)}{P(A,\neg B)}, \frac{P(B)P(\neg A)}{P(B,\neg A)})$
λ	Lift	$[0,\infty)$	$\frac{P(A,B)}{P(A)P(B)}$
S	Collective strength	$[0,\infty)$	$\frac{P(A,B)+P(\neg A,\neg B)}{P(A)P(B)+P(\neg A)P(\neg B)} \cdot \frac{1-P(A)P(B)-P(\neg A)P(\neg B)}{1-P(A,B)-P(\neg A,\neg B)}$
χ^2	χ^2	$[0,\infty)$	$\sum_i \frac{(P(A_i)-E_i)^2}{E_i}$

Interestingness Measures - Properties

Symbol	Measure	Range	O1	O2	O3	O3'	O4
φ	φ-coefficient	$[-1, 1]$	Y	N	Y	Y	N
λ	Goodman-Kruskal's	$[0, 1]$	Y	N	N*	Y	N
α	Odds ratio	$[0, \infty)$	Y	Y	Y*	Y	N
Q	Yule's Q	$[-1, 1]$	Y	Y	Y	Y	N
Y	Yule's Y	$[-1, 1]$	Y	Y	Y	Y	N
κ	Cohen's	$[-1, 1]$	Y	N	N	Y	N
M	Mutual information	$[0, 1]$	N**	N	N*	Y	N
J	J-Measure	$[0, 1]$	N**	N	N	N	N
G	Gini index	$[0, 1]$	N**	N	N*	Y	N
s	Support	$[0, 1]$	Y	N	N	N	N
c	Confidence	$[0, 1]$	N**	N	N	N	Y
L	Laplace	$[0, 1]$	N**	N	N	Y	N
V	Conviction	$[0.5, \infty)$	N**	N	N	Y	N
I	Interest	$[0, \infty)$	Y	N	N	N	N
cos	Cosine	$[0, 1]$	Y	N	N	N	Y
PS	Piatetsky-Shapiro's	$[-0.25, 0.25]$	Y	N	Y	Y	N
F	Certainty factor	$[-1, 1]$	N**	N	N	Y	N
AV	Added value	$[-0.5, 1]$	N**	N	N	N	N
S	Collective strength	$[0, \infty]$	Y	N	Y*	Y	N
θ	Jaccard	$[0, 1]$	Y	N	N	N	Y
K	Klosgen's	$[(\frac{2}{\sqrt{3}} - 1)^{\frac{1}{2}}[2 - \sqrt{3} - \frac{1}{\sqrt{3}}], \frac{2}{3\sqrt{3}}]$	N**	N	N	N	N

O1: Symmetry under variable permutation.

O2: Row and column scaling invariance.

O3: Antisymmetry under row or column permutation.

O3': Inversion invariance

O4: Null invariance.

Y*: Yes if measure is normalized.

N*: Symmetry under row or column permutation.

N:** No unless the measure is symmetrized by taking $\max(M(A, B), M(B, A))$.