TECHNISCHE UNIVERSITÄT DRESDEN

Dresden Database Systems Group

Association Rule Mining

Programming for Data Science

Motivation

Scenario

- In a shop, items are bought together and form a shopping cart (transaction)
- A set of transactions forms a dataset

Goal

- Analysis of items that are frequently bought together
- Establish association rules that represent these strong relationships
- E. g., if a customer buys flour and eggs, he/she is likely to buy butter, too
- Associations rules are characterized with different measures

Applications

- Cross Marketing
- Attached Mailing/Add-on Sales
- Design of catalogues and shop layouts
- Customer segmentation

Terminology

Items I

- $I = \{i_1, \ldots, i_m\}$ is a set of literals
- Each item is uniquely identified by i

Transaction T

- $T = \{i_1, \ldots, i_k\}$ is a set of items (*itemset*) that a bought together

Dataset D

- $D = \{T_1, \ldots, T_n\}$ is a set of transactions
- A dataset D stores the transactions of a shop for a certain period of time

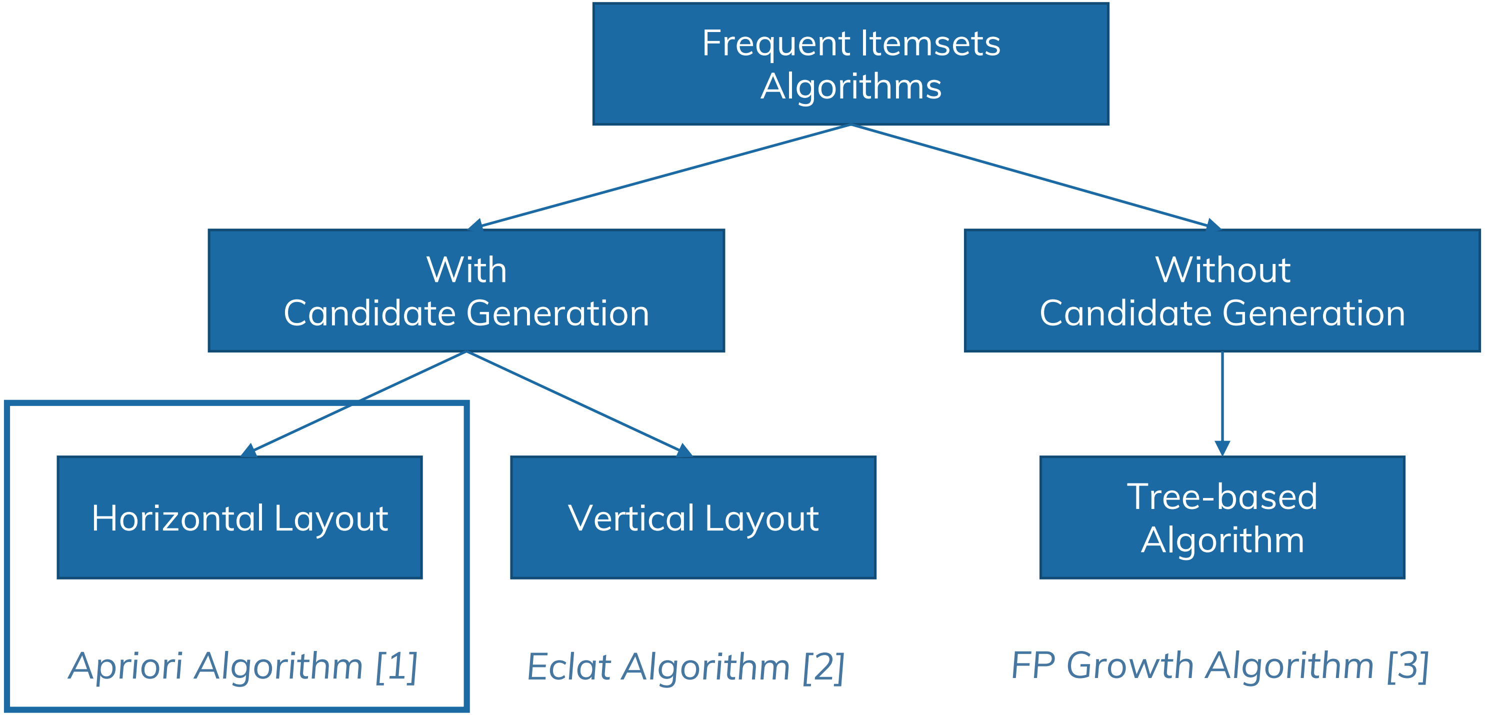
Association rule

- Association rules are rules that imply a relation between itemsets
- Formally: $\mathrm{X} \rightarrow Y$, where $\mathrm{X} \subseteq I$, $\mathrm{Y} \subseteq I$ and $\mathrm{X} \cap Y = \emptyset$

Minimal support

- Percentage of transactions containing itemset i

Classification



[1] Agrawal, Srikant: Fast algorithms for mining association rules in large databases. VLDB ’94.
[2] Zaki et al.: New Algorithms for Fast Discovery of Association Rules. KDD ‘97.
[3] Han: Mining Frequent Patterns Without Candidate Generation. SIGMOD '00.

Apriori Algorithm

Basic idea: Multiple passes over the data

- First pass: Count support of individual items (1-itemset) and determine if they are frequent
- Subsequent passes: e.g. pass k (itemsets with k elements):
  - Frequent itemset of pass $(k-1)$ generates candidate itemsets for pass k
- The support for candidate itemsets is calculated by scanning the database

Remarks

- The candidate itemset is a superset of the set of frequent itemsets
- The set of candidate itemsets is smaller than the set of k-element subsets of I
- Correctness of algorithm can be proved

Anti-Monotonicity

- If $(k-1)$–itemset is frequent k-itemset may be also frequent
- If $(k-1)$–itemset is not frequent k-itemset can never be frequent

Apriori Algorithm (2)

Dresden Database Systems Group

Notation

- A candidate k –itemset $\mathrm{c} = (\mathrm{c}_1, \mathrm{c}_2, \dots, \mathrm{c}_k)$
  - Items are kept in lexicographic order
  - $c.count$ is the support of c

- L_k: Set of frequent k-itemset
  - Have minimum support
  - Each member consists of two fields: the itemset and the support count $c.count$

- C_k: Set of candidate k - itemsets (potentially frequent itemsets)
  - Each member consists of two fields: the itemset and the support count $c.count$

Apriori Algorithm (3)

```
Apriori(I, D, minsupp)
        L_1 := {frequent 1-itemsets of I};
        k := 2;
        while L_{k-1} ≠ ∅ do
                C_k := AprioriGen(L_{k - 1});
                forall transaction T ∈ D do
                        CT := subset(C_k, T) // All candidates of C_k that are in T
                        forall candidate c ∈ CT do c.count++;
                L_k := {c ∈ C_k | (c.count / |D|) ≥ minsup};    // Prune step
                k++;
        return ⋃_k L_k;

AprioriGen(L_{k - 1})
        insert into C_k // Join step
        select p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}
        from L_{k - 1} p, L_{k - 1} q
        where (p.item_1 = q.item_1), ...,(p.item_{k-2} = q.item_{k-2}), (p.item_{k-1} < q.item_{k-1});

        forall itemset c ∈ C_k do
                forall (k-1)-itemset s of c do
                if s ∉ L_{k - 1} then
                        delete c from C_k; // Prune step
```

Apriori Algorithm (4)

Dresden Database Systems Group

Example AprioriGen

$$L_3 = \{(1\ 2\ 3), (1\ 2\ 4), (1\ 3\ 4), (1\ 3\ 5), (2\ 3\ 4)\}$$

- Join step

$$p \in L_3 = (1\ 2\ 3)$$

$$q \in L_3 = (1\ 2\ 4)$$

$$c \in C_4 = (1\ 2\ 3\ 4)$$

- After Join

$$C_4 = \{(1\ 2\ 3\ 4), (1\ 3\ 4\ 5)\}$$

- Pruning checks if all k-1 subsets of a candidate are in L_{k-1}
  - $(1\ 2\ 3\ 4) \rightarrow (1\ 2\ 3), (1\ 2\ 4), (1\ 3\ 4), (2\ 3\ 4) \rightarrow$ ok !
  - $(1\ 3\ 4\ 5) \rightarrow (1\ 3\ 4), (1\ 3\ 5), (1\ 4\ 5), (3\ 4\ 5) \rightarrow$ prune!

$$C_4 = \{(1\ 2\ 3\ 4)\}$$

Example Apriori Algorithm

Dresden Database Systems Group

TID	Items
1	A, C, D, E
2	C, D, E
3	A, B, C, E
4	D, E
5	A, D
6	A

minsupp σ= **2** (2/6)

L_k: Set of frequent k- itemset
C_k: Set of candidate k - itemsets

C_1

1-Itemset	σ
A	4
B	1
C	3
D	4
E	4

L_1

1-Itemset	σ
A	4
C	3
D	4
E	4

Delete {B} because $\sigma(\{B\}) < minsupp$

Example Apriori Algorithm (2)

Dresden Database Systems Group

TID	Items
1	A, C, D, E
2	C, D, E
3	A, B, C, E
4	D, E
5	A, D
6	A

minsupp σ= **2** (2/6)

L_1

1-Itemset	σ
A	4
C	3
D	4
E	4

C_2

2-Itemset	σ
{A, C}	2
{A, D}	2
{A, E}	2
{C, D}	2
{C, E}	3
{D, E}	3

L_2

2-Itemset	σ
{A, C}	2
{A, D}	2
{A, E}	2
{C, D}	2
{C, E}	3
{D, E}	3

Example Apriori Algorithm (3)

TID	Items
1	A, C, D, E
2	C, D, E
3	A, B, C, E
4	D, E
5	A, D
6	A

minsupp σ= **2** (2/6)

L_2

2-Itemset	σ
{A, C}	2
{A, D}	2
{A, E}	2
{C, D}	2
{C, E}	3
{D, E}	3

- Delete {A, C, D} because σ({A, C, D}) < minsupp
- Delete {A,D,E} because σ({A, D, E}) < minsupp

C_3

3-Itemset	σ
{A, C, D}	1
{A, C, E}	2
{A, D, E}	1
{C, D, E}	2

L_3

3-Itemset	σ
{A, C, E}	2
{C, D, E}	2

Example Apriori Algorithm (4)

TID	Items
1	A, C, D, E
2	C, D, E
3	A, B, C, E
4	D, E
5	A, D
6	A

minsupp σ= **2** (2/6)
minconf c = **4** (4/6)

L_3

3-Itemset	σ
{A, C, E}	2
{C, D, E}	2

C_4

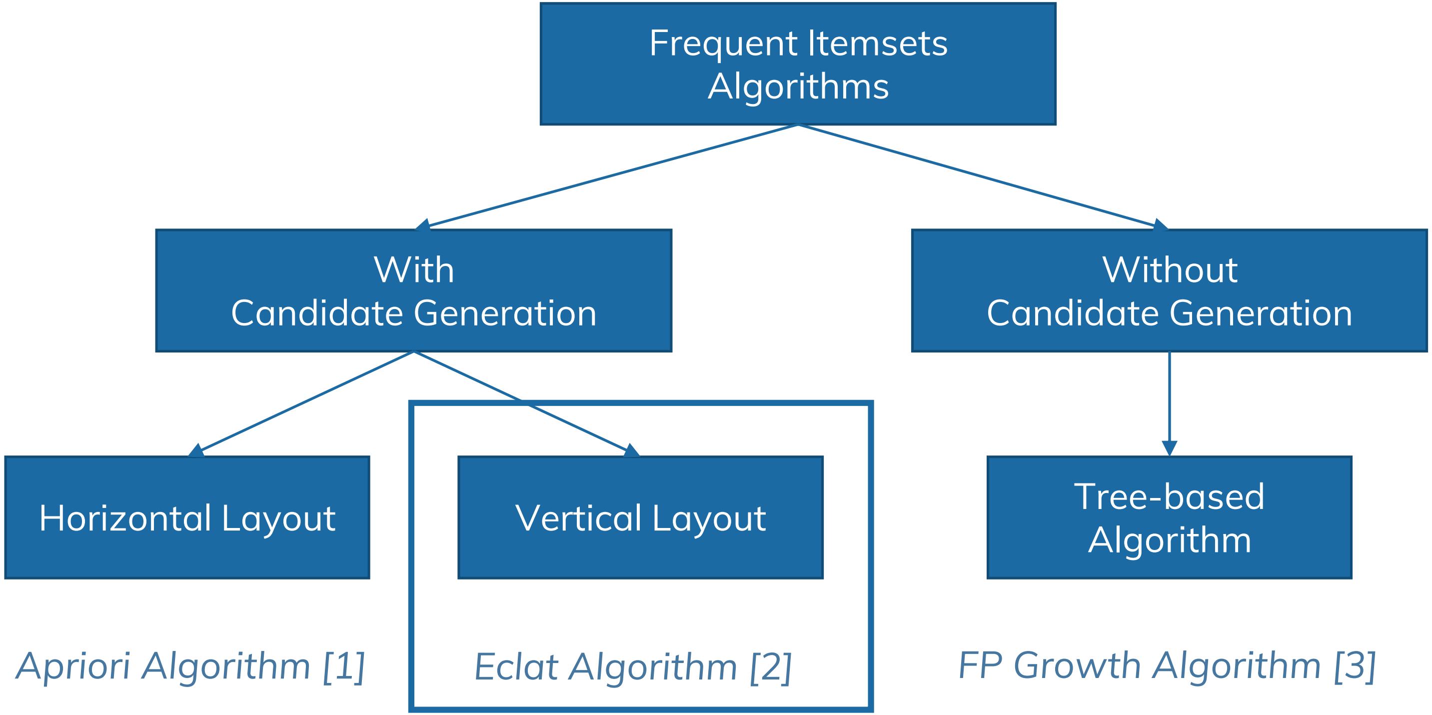
4-Itemset	σ

L_4

4-Itemset	σ

- Join cannot generate candidates
- (A, C, D, E) would be deleted since, e.g., (A, C, D) is not frequent
- No further generation because $L_4 = C_4 = \emptyset$
- Result: $L_1 \cup L_2 \cup L_3$

Classification



[1] Agrawal, Srikant: Fast algorithms for mining association rules in large databases. VLDB ’94.
[2] Zaki et al.: New Algorithms for Fast Discovery of Association Rules. KDD ‘97.
[3] Han: Mining Frequent Patterns Without Candidate Generation. SIGMOD '00.

Eclat Algorithm

Eclat Algorithm

- **E**quivalence **CLA**ss **T**ransformation (ECLAT)
- Vertical data layout

Vertical Layout

- Each item has a list with transactions (TID list)
- Intersection of TID lists leads to frequent itemsets
- Advantage: Frequency of item can be retrieved from list cardinality
- Disadvantage: TID list may be too large for main memory

Equivalence classes

- An equivalence class contains all k-itemsets with the same prefix (lexicographic sorted items)
- Frequent combinations are only possible in equivalence classes
- Advantages: Pruning and Parallelization

Eclat Algorithm (2)

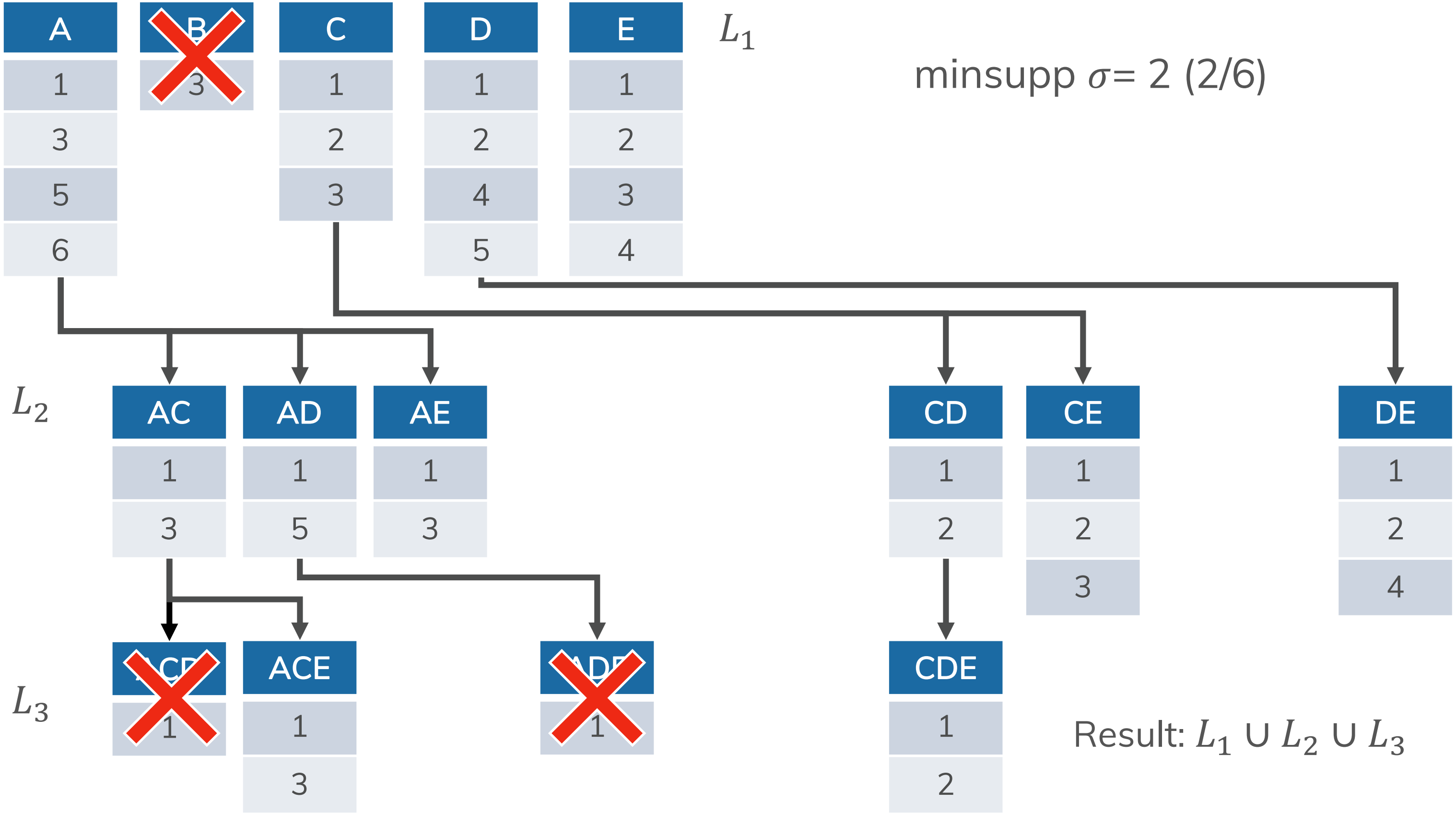
Horizontal layout

TID	Items
1	A, C, D, E
2	C, D, E
3	A, B, C, E
4	D, E
5	A, D
6	A

Vertical layout

A	B	C	D	E
1	3	1	1	1
3		2	2	2
5		3	4	3
6			5	4

Eclat Algorithm (3)



minsupp σ= 2 (2/6)

Result: $L_1 \cup L_2 \cup L_3$

Association Rule Mining

Problem

- Find association rules with minimum confidence among frequent itemsets

Procedure

- Given itemsets X, Y with $\mathrm{X} \subset Y$, then

$$\mathrm{c}\big((\mathrm{X} \setminus \mathrm{Y}) \rightarrow \mathrm{Y}\big) > minconf \Rightarrow \mathrm{c}\big((\mathrm{X} \setminus \mathrm{Y}') \rightarrow \mathrm{Y}'\big) > minconf \ \forall (Y' \subseteq Y)$$

- No further database scans needed
- For association rules one should start with a small Y' and exclude Y with $\mathrm{Y}' \subset \mathrm{Y}$ if the following holds

$$\mathrm{c}\big((\mathrm{X} \setminus \mathrm{Y}') \rightarrow \mathrm{Y}'\big) < minconf$$

- Reason: Support of subset of Y' cannot be smaller than support of Y'

(2) Frequent Itemsets

Combinations		
A (7)	A ∧ B (3)	A ∧B ∧ C (1)
B (5)	A ∧ C (3)	
C (3)	B ∧ C (2)	

(3) Association rules

Rule A ∧ B → C

Task

Dresden Database Systems Group

Step 0

- You will get two tsv files from us. Rows are transactions with purchased items. Load it in your language/environment.
- Use the smaller file (items.tsv, minsupp of 70%) for development and the larger file (retail.tsv, minsupp of 10%) for evaluation.

Step 1

- Implement the Apriori algorithm*.
- Using your implementation, extract frequent item sets from the given datasets.

Step 2

- Implement the ECLAT algorithm*.
- Using your implementation, extract frequent item sets from the given datasets.

Step 3

- Compare the run times of both algorithms on both files.

*use your own implementation

Package suggestions

R

- (data.table)
- microbenchmark

python3

- numpy
- pandas
- Itertools
- timeit
- (os)

Items

1	5-Minute-Noodles
2	Pineapple
3	Applesauce
4	Asia-Snack
5	Cup
6	Beer
7	College pad
8	Canned tomatoes
9	Peas
10	Peas & Carrots
11	Lint roller
12	Putty
13	Mushroom-Spaghetti
14	Salt sticks
15	Mustard
16	Spaghetti
17	Cream
18	Deodorant
19	Disinfection
20	Showergel
21	Shot
22	Gummi bears
23	Hair tie
24	Hazelnut waffle
25	Glue stick
26	Cap bomb
27	Air nozzle

28	Sticky note
29	Maoam
30	Milk bar
31	Mouth wash
32	Shaver
33	Small bowl
34	Chocolate bar
35	Pen
36	Tomato paste
37	Coaster
38	Toothpaste
39	Envelopes
40	Deco balls
41	Scented candles
42	Napkins
43	Cereal
44	Cactus
45	Bend lights
46	Booklet
47	OREO cookies
48	Light bag
49	Puzzle
50	Novels
51	Cards
52	Detergent
53	Tissues

Solution items.tsv (70%)

Frequent Item sets length 1

- Beer (6)
- Shot (21)
- Gummi bears (22)
- Hazelnut waffle (24)
- Milk bar (30)
- Chocolate bar (34)
- Coaster (37)
- Scented candles (41)

Frequent Item sets of length 2

- Shot, Beer (21,6)
- Gummi bears, Beer (22,6)
- Hazelnut waffle, Beer (24,6)
- Milk bar, Beer (30,6)
- Chocolate bar, Beer (34,6)
- Coaster, Beer (37,6)
- Scented candles, Beer (41,6)
- Shot, Chocolate bar (21,34)
- Gummi bears, Chocolate bar (22,34)
- Milk bar, Chocolate bar (30,34)
- Gummi bears, Milk bar (22,30)
- Gummi bears, Hazelnut waffle (22,24)

Frequent Item sets of length 3

- Shot, Chocolate bar, Beer (21,34,6)
- Gummi bears, Chocolate bar, Beer (22,34,6)
- Gummi bears, Milk bar, Beer (22,30,6)
- Gummi bears, Hazelnut waffle, Beer (22,24,6)
- Gummi bears, Chocolate bar, Beer (30,34,6)

Exercise Appointment

Dresden Database Systems Group

We compare and discuss the results

- Tuesday, 10.12.2019,
- Consultation: 05.12.2019,
- Please prepare your solutions! Send us your code!

If you have questions, please mail us:

claudio.hartmann@tu-dresden.de Orga + Code
lucas.woltmann@tu-dresden.de Tasks + Python
lars.kegel@tu-dresden.de Tasks + R