Data mining

Frequent itemsets
Association&decision rule mining

University of Szeged

What frequent itemsets could be used for?

- Features/observations frequently co-occurring in some database can gain us useful insights
- How a marketing person can make use of it? What about a data scientist?

Transaction ID	Items
1	{milk, bread, salami}
2	{beer, diapers}
3	{beer, wurst}
4	{beer, baby food, diapers}
5	{diapers, coke, bread}

Possible forms – Horizontal

Transaction ID	Items
1	{milk, bread, salami}
2	{beer, diapers}
3	{beer, wurst}
4	{beer, baby food, diapers}
5	{diapers, coke, bread}

Possible forms – Vertical/inverted

Item	Basket
milk	{1}
bread	{1, 5}
salami	{1}
beer	{2,3,4}
diapers	{2,4,5}
wurst	{3}
baby food	{4}
coke	{5}

Possible forms – Relational

Basket	Item
1	milk
1	bread
1	salami
2	beer
2	diapers
3	beer
3	wurst
4	beer
4	baby food
4	diapers
5	diapers
5	coke
5	bread

Collecting association rules

- Goal: find item sets of the transactional database with high **support** and **confidence**
- Support of set X: $s(X) = |\{t_i | t_i \in T \wedge X \subseteq t_i\}|$
- Normalized support: $s(X)$ normalized by the number of transactions
- Confidence of rule $A \rightarrow B$: $c(A \rightarrow B) = \frac{s(A \cup B)}{s(A)} \approx P(B|A)$
- The meaning of a rule $A \rightarrow B$: given that items included in set A are put in the basket, chances are high that the items included in B are also in some basket

Interestingness of association rules

- Are all rules with high support and confidence equally interesting?
- Confidence of a rule can simply be high due to the fact the items on its right side are frequently purchased independently of the items on the left side. Any example?
- Interest score of rule $A \to B$: $I(A \to B) = c(A \to B) - P(B)$
- What is the interpretation of this score? (Interesting rules will have high absolute values. Why?)
- Choose threshold so that the number of rules defined is manageable
- There are plenty other scores for measuring interestingness: χ^2, κ, ...

What frequent itemsets can be utilized for? – Plagiarism detection

- Somewhat counter-intuitively let the sentences be the baskets and documents the items
- Conclusion
  - Sometimes we need to be flexible about the concept of „items" comprising „baskets"
  - Our goal is to examine the relation of items to each other, and not that of baskets (we had that one before)
  - How frequent itemsets could be interpreted if items and baskets were determined vice versa?

What else frequent itemsets could be used for? – Data mining

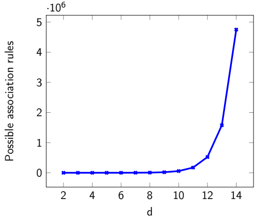
- We can use them to build a simple classifier
- Missing feature values can be estimated knowing how features typically co-occur with each other
- We can merge features if they show highly similar behavior in the database

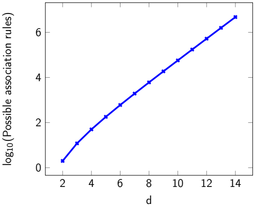
The general schema of producing associational rules

- Collect the set of frequent items F having a (normalized) support surpassing some threshold t
- Partition F into non-empty, disjunct subsets and calculate the confidence of the rules determined

Why naive approach fails?

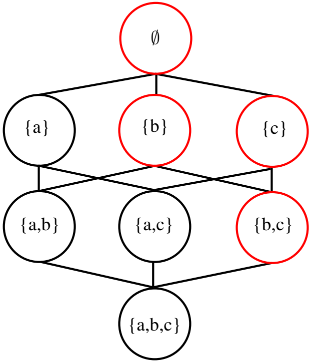
- How a naive approach would look like?
- d items $\Rightarrow 3^d - 2^{d+1} + 1$ possible rules (e.g. $d = 9 \Rightarrow 18660$ possibilities)
- Proof (hint: $(1+x)^d = \sum_{j=1}^{d} \binom{d}{j} x^{d-j} + x^d$)
  - $\sum_{i=1}^{d} \binom{d}{i} \sum_{j=1}^{d-i} \binom{d-i}{j} = \sum_{i=1}^{d} \binom{d}{i} (2^{d-i} - 1) = \sum_{i=1}^{d} \binom{d}{i} 2^{d-i} - \sum_{i=1}^{d} \binom{d}{i} = (3^d - 2^d) - (2^d - 1)$





A priori principle

- Itemset I is frequent $\Rightarrow \forall J \subseteq I$ itemsets are frequent
- What can we say if itemset I is *not* frequent?
- Anti-monotone property: function f is said to be anti-monotone if $\forall X, Y \in \mathcal{P}(U) : X \subseteq Y \Rightarrow f(X) \geq f(Y)$



The A priori principle in action

- Let assume a frequency threshold of 3

Item	Frequency
beer	3
bread	4
coke	2
diapers	4
milk	4
wurst	1

The A priori principle in action

- Let the frequency threshold be 3

Item	Frequency
beer	3
bread	4
coke	2
diapers	4
milk	4
wurst	1

Item	Frequency
{beer, bread}	2
{beer, diapers}	3
{beer, milk}	2
{bread, diapers}	3
{bread, milk}	3
{diapers, milk}	3

Calculating frequent itemsets

1. Algorithm Pseudocode for calculation of frequent itemsets

Input: set of possible items U, transaction database T, frequency threshold t

Output: frequent itemsets

$C_1 := \mathcal{U}$
Calculate the support of C_1
$F_1 := \{x | x \in C_1 \wedge s(x) \geq t\}$
for $(k = 2; k < |\mathcal{U}| \&\& F_{k-1} \neq \emptyset; k++)$ **do**
    Determine C_k based on F_{k-1}
    Calculate the support of C_k
    $F_k := \{X | X \in C_k \wedge s(X) \geq t\}$
end for
return $\cup_{i=1}^{k} F_i$

Possible ways of determining C_k

- Based on the elements in F_1. Avoid this!
- Combining the elements of F_{k-1} and F_1. Somewhat better
- Combining the elements of F_{k-1} and F_{k-1} itself. Generates less candidates but does not miss any candidate which has the chance to be a frequent itemset. Combine two itemsets if $\exists A = \{a_1, a_2, \ldots, a_{k-1}\}, B = \{b_1, b_2, \ldots, b_{k-1}\} \in F_{k-1}$: $\forall a_i = b_i, 1 \leq i \leq k-2 \wedge a_{k-1} \neq b_{k-1}$
- We need an ordering over the elements of F (e.g. by converting them to integers) and store them in that form

Possible ways of calculating the frequency for C_k

- $\binom{|F_{k-1}|}{2}$ frequencies might need to be stored $\rightarrow$ memory limitations (for brevity, let $f = |F_{k-1}|$)
- Depending on the ratio of non-zero elements, we can either store them *proactively* or store them *reactively*, in the form of (i, j, c) triplets, storing by c the co-occurrence of items i and j
- In a *one-dimensional triangular matrix*, the frequency of item pair $(i, j), i < j$ is stored at index $(j - i) + \sum_{r=f-i+1}^{f-1} r \Rightarrow$ no need to explicitly store index values i and j for each counter
- If the number of zero frequencies surpasses $\frac{\binom{f}{2}}{3}$ representation using triplets pays off (i, j, c)

Calculating C_k – example

- Suppose there are 10^7 baskets, 10 items/baskets
- There are 10^5 different items in total
- Using the triangular matrix representation $\approx 5 * 10^9$ integer is required
- In worst case there are at most $10^7\binom{10}{2}$ different pairs of items in the transaction dataset $\rightarrow$ *max*. $\approx 3 * 4.5 * 10^8 = 1.35 * 10^9$ integer suffices to store the non.zero frequencies of the purchase of item pairs

Compressing frequent itemsets

- Maximal frequent itemsets: $\{I | I \text{ frequent } \wedge \nexists \text{ frequent } J \supset I\}$
- Closed itemsets: $\{I | \nexists J \supset I : s(J) = s(I)\}$
- Closed frequent itemsets: closed itemset having a support above some frequency threshold
- Can a maximal frequent itemset be non-closed?
- Can a closed itemset be non-maximal?

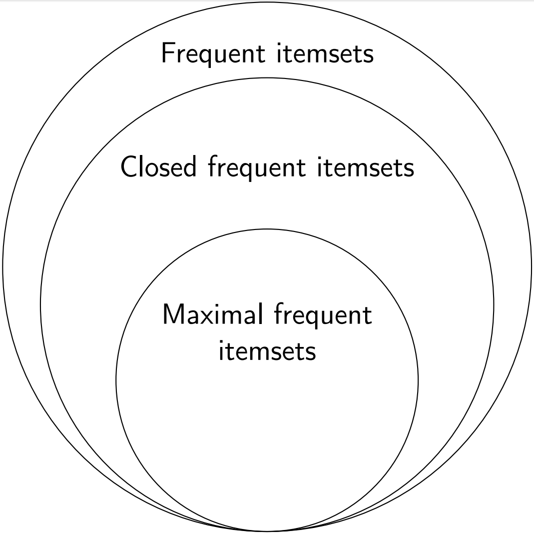
Compressing frequent itemsets — Example (t = 3)

Item	Frequency	Maximal	Closed	Closed frequent
A	4			
B	5			
C	3			
AB	4			
AC	2			
BC	3			
ABC	2			

Compressing frequent itemsets — Example (t = 3)

Item	Frequency	Maximal	Closed	Closed frequent
A	4	-	-	-
B	5	-	+	+
C	3	-	-	-
AB	4	+	+	+
AC	2	-	-	-
BC	3	+	+	+
ABC	2	-	+	-

The relation of different classes to each other

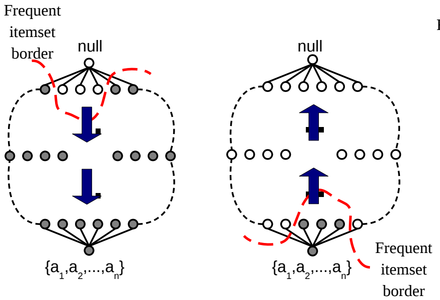


PCY (Park-Chen-Yu) algorithm – an extension to A priori

- Besides counting the frequency of standalone items keep track of the frequency of buckets into which pairs of elements get assigned according to some hash function
- What can be said based on the aggregated frequency counts? What cannot be made for sure?
- Only pairs that consist of frequent items and which got hashed to a frequent bucket have the chance to be indeed frequent in the end

Practical considerations

- We might as well do sampling from the transaction database $\Rightarrow$ correctness and completeness is sacrificed
- Different strategies can be applied for extracting frequent itemsets

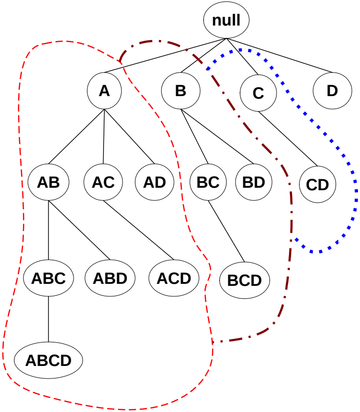


(a) General-to-specific/ Top-down (b) Specific-to-general/ Bottom-up (c) Bidirectional/ Hybrid

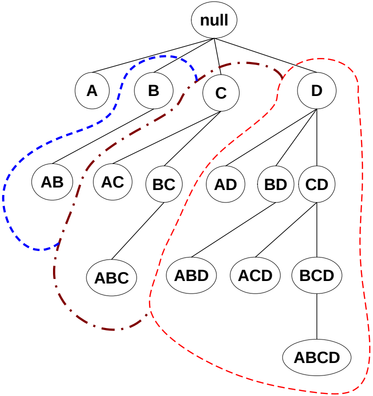
Generalizing A priori

- It is not necessary to do the expansion according to the different "levels" of the item lattice
- We can do expansion according to any equivalent classes
- Originally we defined equivalent classes based on the size of the itemsets
- Alternatively we can define equivalent classes based on the size of the overlap in the prefixes/suffixes

Generalizing A priori – the prefix and suffix view



(a) Prefix tree



(b) Suffix tree

FP trees

- Use a data structure which makes the extraction of frequent datasets easy
- FP trees: alternative, condensed representation
- It can be constructed by processing the rows of the transaction database in one pass
- Market baskets are represented as paths in the tree
- Certain item subsets show up multiple times $\Rightarrow$ overlapping paths $\Rightarrow$ compressability
- Hopefully the whole transaction dataset can be stored in the main memory as a result

Constructing FP trees

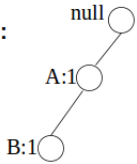
- Useful heuristic: order items according to their decreasing support
- Processing one line of the transaction database
  - Continue paths starting from the root with the same prefix to the currently processed line
    - Increment the frequencies stored for visited nodes
    - Set the frequency of the new nodes to 1
  - Otherwise start a new path from the root node
    - Assign 1 for the frequency value of new nodes
- Include pointers between items of the same kind

FP tree – example

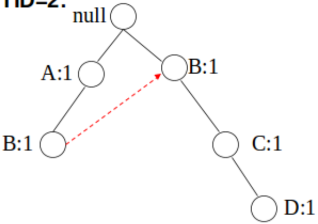
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

[1]

After reading TID=1:



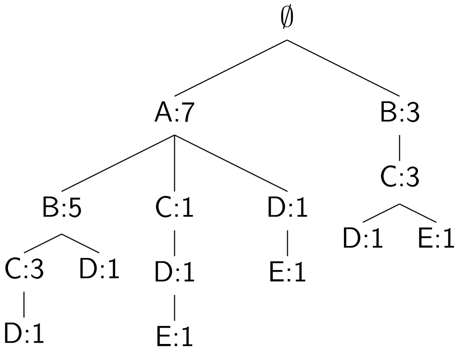
After reading TID=2:



[1] based on the slides of Tan, Steinbach, Kumar: Introduction to Data Mining

FP tree – after processing the last basket

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}



- What trees would we get if items within baskets were ordered according to their increasing/decreasing order of overall support? (`{A:7,B:8,C:7,D:5,E:3}`)

Why to order items based on their support?

- Item i that is ordered at position $r(i)$ among all the items, can add at most $2^{r(i)-1}$ nodes to an FP tree
- Another natural upper bound for the nodes of item i in an FP tree is $s(i)$, hence it has $\leq \min\left(2^{r(i)-1}, s(i)\right)$ presences

i	A	B	C	D	E	Σ
$s(i)$	7	8	7	5	3	30
$r_1(i)$	2	1	3	4	5	—
presences	≤ 2	1	≤ 4	≤ 5	≤ 3	≤ 15
$r_2(i)$	4	5	3	2	1	—
presences	≤ 7	≤ 8	≤ 4	≤ 2	1	≤ 22

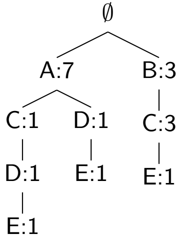
FP-Growth algorithm

- Divide and conquer algorithm working on the FP tree in a bottom-up manner
- If the FP tree reveals that an itemset is frequent, check the support of its supersets
    - Examine FP trees conditioned on some already known frequent itemsets
    - E.g. as `{E}` is frequent, check out the frequency of sets `{A,E}`, `{B,E}`, `{C,E}` and `{D,E}`

FP trees conditioned on some target item(s)

- The part of the FP trees that we would get if we looked at only transactions containing the target item(s)
  - Without building the tree from scratch
    1. Forget about the parts of the tree not related to the target item(s)

TID	Basket
1	~~{A,B}~~
2	~~{B,C,D}~~
3	{A,C,D,E}
4	{A,D,E}
5	~~{A,B,C}~~
6	~~{A,B,C,D}~~
7	~~{B,C}~~
8	~~{A,B,C}~~
9	~~{A,B,D}~~
10	{B,C,E}

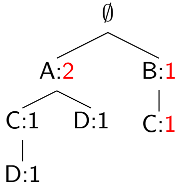


FP tree conditioned on {E}

FP trees conditioned on some target item(s)

- The part of the FP trees that we would get if we looked at only transactions containing the target item(s)
  - Without building the tree from scratch
    2. Let the support of a node be the sum of the **upgraded** supports of its descendants

TID	Basket
⋮	⋮
3	{A,C,D,E}
4	{A,D,E}
⋮	⋮
10	{B,C,E}

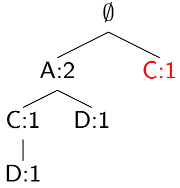


FP tree conditioned on {E}

FP trees conditioned on some target item(s)

- The part of the FP trees that we would get if we looked at only transactions containing the target item(s)
  - Without building the tree from scratch
    3. Eliminate items below frequency threshold (e.g. using a frequency threshold of 2)
    4. Create FP trees conditioned on item pairs (containing item `{E}`), based on which we can determine frequent item triples (e.g. `{D,E}`$\rightarrow$ `{A,D,E}`)

TID	Basket
$\vdots$	$\vdots$
3	{A,C,D,E}
4	{A,D,E}
$\vdots$	$\vdots$
10	{B,C,E}



FP tree conditioned on `{E}`