Data Mining 2020
Frequent Pattern Mining (1)
Frequent Item Sets and Association Rules

Ad Feelders

Universiteit Utrecht

October 30, 2020

Comparison with Graphical Models

In graphical modelling, we are interested in

- Global models, that is, the associations between all variables simultaneously.
- Associations at the variable level: X and Y are dependent means that the value of X provides information about the value of Y (and vice versa).

In frequent item set mining, we are interested in

- Local patterns, that is, the associations between sets of items.
- Associations at the value level: if all items in the set X have the value 1, then all items in the set Y have the value 1 as well (with a certain support and confidence).

$$\mathit{Diapers} \rightarrow \mathit{Beer},\ \mathit{support} = 20\%,\ \mathit{confidence} = 85\%$$

Association Rules

Table *db* with schema $R = \{I_1, \ldots, I_n\}$, I_i is a binary attribute (*item*).
For $X, Y \subseteq R$, with $X \cap Y = \emptyset$, let:

(Transactions in the database)

- $s(X)$ denote the *support* of X, i.e., the number of tuples that have value 1 for all items in X.
- for an *association rule* $X \rightarrow Y$, define
  - the support is $s(X \cup Y)$
  - the confidence is $s(X \cup Y)/s(X)$

Task: find all association rules with support $\geq t_1$ and confidence $\geq t_2$.

Association Rule Algorithm: sketch

There are two thresholds we have to satisfy:

1. Find all sets Z whose support exceeds the minimal threshold. These sets are called *frequent*.
2. Test for all non-empty subsets X of frequent sets Z whether the rule $X \rightarrow Y$ (with $Y = Z \setminus X$) holds with sufficient confidence.

Y = Z - X

All items in set Z minus all the items in set X, it is the difference between the sets

Finding Frequent Sets

The first problem is then: how do we find the frequent sets?

Suppose we simply check all subsets of R. Then we would have to count

$$|\mathcal{P}(R)| = 2^{|R|}$$

subsets on the data base.

For example, if we can check 1024 sets/sec. then:

- For 10 items, we are done in 1 second;
- For 20 items, we need 1024 seconds, or 17 minutes;
- For 100 items, we need (roughly) 4×10^{18} years, which (far) exceeds the age of the universe!

The Apriori Property

Theorem

X *is frequent* $\Rightarrow \forall Y \subseteq X : Y$ *is frequent.*

Proof

$Y \subseteq X \Rightarrow s(Y) \geq s(X)$

Therefore, if $Y \subseteq X$ and $s(X) \geq t_1$, then $s(Y) \geq t_1$.

Conversely, if $Y \subseteq X$ and $s(Y) < t_1$, then $s(X) < t_1$.
In other words, we can search *levelwise* for the frequent sets. The level is the number of items in the set:

A set X is a candidate frequent set *iff all its subsets are frequent.*

Denote by $C(k)$ the sets of k items that are potentially frequent (the candidate sets) and by $F(k)$ the frequent sets of k items.

Apriori Pseudocode

Algorithm 1 Apriori(t_1, R, db)

$C(1) \leftarrow R$
$k \leftarrow 1$
while $C(k) \neq \emptyset$ **do** First level (k) candidates
  $F(k) \leftarrow \emptyset$
  for all $X \in C(k)$ **do**
    if $s(X) \geq t_1$ **then**
      $F(k) \leftarrow F(k) \cup \{X\}$ {Here you have to scan the database!}
    end if
  end for
  $C(k+1) \leftarrow \emptyset$ Nextt level (k+1) candidates
  for all $X \in F(k)$ **do**
    for all $Y \in F(k)$ that share $k-1$ items with X **do**
      if All $Z \subset X \cup Y$ of k items are frequent **then**
        $C(k+1) \leftarrow C(k+1) \cup \{X \cup Y\}$
      end if
    end for
  end for
  $k \leftarrow k+1$
end while

Example: the data

Note that we switch to a more convenient representation of the transactions.

tid	Items
1	ABE
2	BD
3	BC
4	ABD
5	AC
6	BC
7	AC
8	ABCE
9	ABC

Items ABE were bought

We want to find items that were present in two transactions

Minimum support = 2

Example: Level 1

tid	Items
1	ABE
2	BD
3	BC
4	ABD
5	AC
6	BC
7	AC
8	ABCE
9	ABC

Level 1 candidates

Candidate	Support	Frequent?
A	6	✓
B	7	✓
C	6	✓
D	2	✓
E	2	✓

Example: Level 2

tid	Items
1	ABE
2	BD
3	BC
4	ABD
5	AC
6	BC
7	AC
8	ABCE
9	ABC

Candidates for level 2

All possible combinations from single items because all were frequent

Candidate	Support	Frequent?
AB	4	✓
AC	4	✓
AD	1	✗
AE	2	✓
BC	4	✓
BD	2	✓
BE	2	✓
CD	0	✗
CE	1	✗
DE	0	✗

Example: Level 3

Candidate	Support	Frequent?
AB	4	✓
AC	4	✓
AD	1	✗
AE	2	✓
BC	4	✓
BD	2	✓
BE	2	✓
CD	0	✗
CE	1	✗
DE	0	✗

Candidates for level 3

Candidate	Support	Frequent?
ABC	2	✓
ABE	2	✓

Level 3: For example, ABD and BCD are not level 3 candidates.
Level 4: There are no level 4 candidates.

Order, order

Lines 10-11 of the algorithm leads to multiple generations of the set $X \cup Y$.

For example, the candidate ABC is generated 3 times   No point in doing things 3 times

1. by combining AB with AC
2. by combining AB with BC
3. by combining AC with BC

Order, order

The solution is to place an order on the items.

---

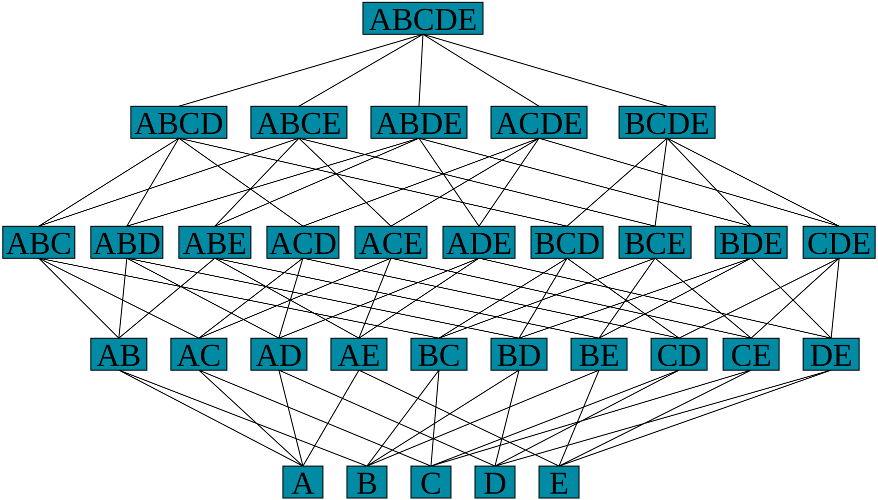
for all $X \in F(k)$ **do**
  for all $Y \in F(k)$ that share <u>the first</u> $k-1$ items with X **do**
    if All $Z \subset X \cup Y$ of k items are frequent **then**
      $C(k+1) \leftarrow C(k+1) \cup \{X \cup Y\}$
    end if
  end for
end for

Now the candidate ABC is generated just once, by combining AB with AC.

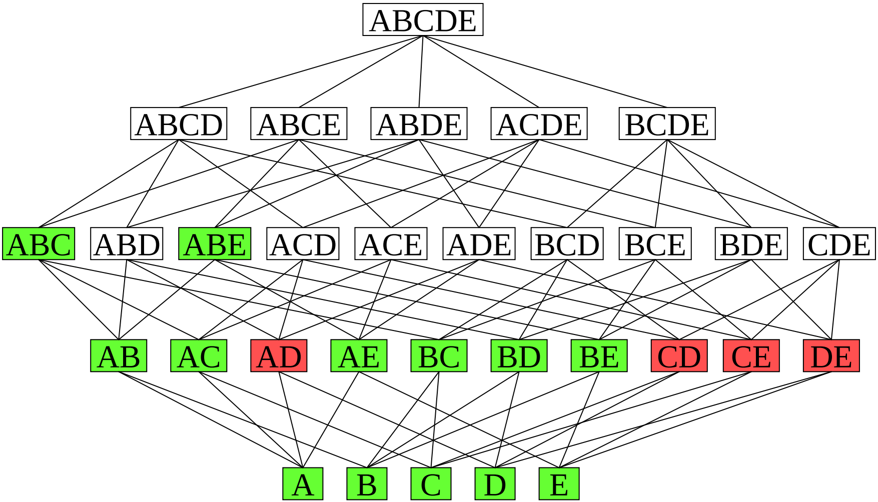
The order itself is arbitrary, as long as it is applied consistently.

The search space

Full item set



Item sets counted by Apriori



The Complexity of Levelwise Search

We rejected the naive algorithm because its complexity was $O(2^{|R|})$. So, what is the complexity of level wise search?

Take a database with just 1 tuple consisting completely of 1's and set minimum support to 1. Then, all subsets of R are frequent! Hence, the worst case complexity of level wise search is $O(2^{|R|})$.

In supermarket you only buy a fraction of products available

However, suppose that *db* is *sparse* (by far the most values are 0), then we *expect* that the frequent sets have some maximal size m with $m << |R|$.

If that expectation is met, we have a worst case complexity of:

$$O\left(\sum_{j=1}^{m+1} \binom{|R|}{j}\right) = O(|R|^{m+1}) << O(2^{|R|}).$$

Generating Association Rules

Generating association rules from the frequent sets is done as follows:

Generate Association Rules

For each frequent set Z **do**

  For all non-empty $X \subset Z$ **do**

    If $s(Z)/s(X) \geq t_2$ **then**

      Output $X \rightarrow Y$ where $Y = Z \setminus X$

Continuing the Example

One found frequent item set ABE

One of the frequent sets is Z = ABE. This generates:

X	$X \rightarrow Y$	Confidence
AB	AB $\rightarrow$ E	$2/4 = 50\%$
AE	AE $\rightarrow$ B	$2/2 = 100\%$
BE	BE $\rightarrow$ A	$2/2 = 100\%$
A	A $\rightarrow$ BE	$2/6 = 33\%$
B	B $\rightarrow$ AE	$2/7 = 29\%$
E	E $\rightarrow$ AB	$2/2 = 100\%$

Suppose $t_2 = 0.75$. Do we need to check A $\rightarrow$ BE and B $\rightarrow$ AE?

Complexity of the Generation

Clearly, this algorithms is again exponential. For every Z, we consider all $(2^{|Z|} - 2)$ non-empty proper subsets X of Z. However:

- $|Z| \leq m << |R|$
- Quite often one generates only those association rules with a singleton Y. This makes the generation algorithm linear.

Drowning in Association Rules: Example

One frequent set may induce many association rules.

ABE generates:

Itemset	Rule	Confidence
AB	AB $\rightarrow$ E	$2/4 = 50\%$
AE	AE $\rightarrow$ B	$2/2 = 100\%$
BE	BE $\rightarrow$ A	$2/2 = 100\%$
A	A $\rightarrow$ BE	$2/6 = 33\%$
B	B $\rightarrow$ AE	$2/7 = 29\%$
E	E $\rightarrow$ AB	$2/2 = 100\%$

Drowning in Association Rules

Mining for association rules has its own dilemma:

and not very interesting

- High confidence and high support rules are probably already known.
- Low confidence and/or low support thresholds lead to a flood of results (could be more than the original database!)

Moreover, not all discovered rules will be interesting: suppose you discover that 60% of the people that buy bread also buy cheese. How interesting is this if you know that 60% of all people buy cheese?

Managing the Flood

report only the top results of “how many user would like to see”

- Rank or (partially) order the results on support and confidence.
- Filter for interesting rules (what is interesting?)
- Mine for less rules (condensed representations)

An Interestingness Measure: Lift

second point from last slide

The *lift* of an association rule tells us how much the rule increases the probability of the consequent:

$$\text{lift}(X \rightarrow Y) = \frac{P(Y|X)}{P(Y)} = \frac{P(X,Y)}{P(X)P(Y)}$$

For example, if a rule has a confidence $P(Y|X)$ of 0.9 while $P(Y) = 0.2$, then the lift of the rule is 4.5

Note the slight abuse of notation: X and Y are not random variables, but item sets.

Generating Less Results

third point from two slides back

Condensed representations:

- Maximal Frequent Itemsets
- Closed Frequent Itemsets

Maximal Frequent Itemset

An itemset I is maximal frequent iff

- I is frequent and
- no proper superset of I is frequent

Clearly, each frequent itemset is a subset of at least one maximal frequent itemset. Hence, the set of all maximal frequent itemsets is a condensed representation of the set of all frequent itemsets.

But given the maximal frequent item sets and their support, we can not infer the *support* of every frequent item set.

Closed Frequent Itemsets

An item set is closed if every superset has lower support.

More formally:

transactions

- For an itemset I, denote by $\sigma(I)$ the set of tuples in which all items in I are "bought", i.e., $\sigma(I)$ is the set of tuples that *support* I.
- An itemset I is closed iff for all proper supersets J, $\sigma(I)$ is a proper superset of $\sigma(J)$: itemset I can't be extended without decreasing the support.
- An itemset I is a closed frequent itemset iff it is both frequent and closed.

Closure Operator

Two operators:

sigma of I

$$\sigma(I) = \{t \in db \mid \forall i \in I, i \in t\}$$

that returns

"The set of transactions that contain all items in I".

f of set of transactions T

$$f(T) = \{i \in R \mid \forall t \in T, i \in t\}$$

that returns

"The set of items included in all transactions in T".

Closure Operator

the closure of itemset I

Let $c(I)$ be the set of items that are bought in all transactions in which all items in I are bought, that is

$$c(I) = f(\sigma(I))$$

$c(I)$ is called the closure of I.

An itemset I is closed if and only if $c(I) = I$

Example of Closure Operator

tid	Items
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE

$$c(\{A,B\}) = \{A,B,C,E\}$$

Why?

where both A and B occur

$$\begin{aligned} c(\{A,B\}) &= f(\sigma(\{A,B\})) = f(\{3,5\}) \\ &= \{A,B,C,E\} \end{aligned}$$

Note that $I \subseteq c(I)$ and I has the same support as $c(I)$.

Running Example

tid	Items
1	ACD
2	BCE
$\boxed{3}$	ABCE
4	BE
$\boxed{5}$	ABCE

tid	Items
1	ACD
2	BCE
3	$\boxed{\text{ABCE}}$
4	BE
5	$\boxed{\text{ABCE}}$

apply sigma

$$\sigma(\{A, B\}) = \{3, 5\}$$

apply f

$$f(\{3, 5\}) = \{A, B, C, E\}$$

Example of Closure Operator

tid	Items
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE

$$c(\{A,C\}) = \{A,C\}$$

Why?

$$\begin{aligned} c(\{A,C\}) &= f(\sigma(\{A,C\})) = f(\{1,3,5\}) \\ &= \{A,C\} \end{aligned}$$

$\{A,C\}$ is closed since $c(\{A,C\}) = \{A,C\}$

Running Example

tid	Items
$\boxed{1}$	ACD
2	BCE
$\boxed{3}$	ABCE
4	BE
$\boxed{5}$	ABCE

tid	Items
1	$\boxed{\text{A}}\boxed{\text{C}}$D
2	BCE
3	$\boxed{\text{A}}$B$\boxed{\text{C}}$E
4	BE
5	$\boxed{\text{A}}$B$\boxed{\text{C}}$E

$\sigma(\{A, C\}) = \{1, 3, 5\}$ $\qquad$ $f(\{1, 3, 5\}) = \{A, C\}$

Example Try at home

The closed frequent itemsets for

tid	Items
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE

with minimum support 2 are

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

Closure properties

Theorem

If X is subset Y and X has the same support as Y, then closure of X is closure of Y.

If $X \subset Y$ *and* $s(X) = s(Y)$ *then* $c(X) = c(Y)$.

Proof.

1. Assume $X \subset Y$ and $s(X) = s(Y)$.
2. Since $X \subset Y$, it follows that $\sigma(Y) \subseteq \sigma(X)$.
3. From $s(X) = s(Y)$ it follows that $\sigma(Y) = \sigma(X)$.
4. Hence $c(X) = f(\sigma(X)) = f(\sigma(Y)) = c(Y)$.

$\square$

Closure properties

Theorem

If $c(X) = Z$ then $s(X) = s(Z)$.

Proof.

The closure of an item set X is the set of items $Z \supseteq X$ that is contained in all transactions that contain X. So if $c(X) = Z$, then $\sigma(X) = \sigma(Z)$.

It follows that $s(X) = s(Z)$. $\square$

Closure properties

Theorem

If $c(X) = Z$ then Z is closed.

Proof.

1. Assume $c(X) = Z$.
2. It follows that $\sigma(X) = \sigma(Z)$.
3. So $c(X) = f(\sigma(X)) = f(\sigma(Z)) = c(Z) = Z$

$\square$

A-Close Algorithm

Phase 1: Discover all frequent closed itemsets in *db*.

Phase 2: Derive all frequent itemsets from the frequent closed itemsets found in phase 1.

A-Close: Phase 1

We just want to find the generators in phase 1

Determine a set of *generators* that will produce all frequent closed itemsets by application of the closure operator c.

An itemset Y is a *generator* of a closed itemset Z if $c(Y) = Z$, and there is no $X \subset Y$ with $c(X) = Z$.

A-Close: Phase 1

Levelwise construction: G_{k+1} is constructed using G_k.

Using their support, and the support of their k-subsets in G_k, infrequent candidates and candidates that have the same support as one of their subsets are deleted from G_{k+1}.

Example: G_1

Candidate	Support
A	3
B	4
C	4
D	1
E	4

$\Longrightarrow$

Level 1 item sets that are frequent are generators

Generator	Support
A	3
B	4
C	4
E	4

Minimum support $= 2$

Example: G_2

Candidate	Support
AB	2
AC	3
AE	2
BC	3
BE	4
CE	3

$\xrightarrow{\text{Compute support}}$

Level 2 item sets that are frequent are generators

Generator	Support
AB	2
AE	2
BC	3
CE	3

AC is pruned, because subset A has the same support (and therefore the same closure)
BE is pruned because it has the same support as B (and E).

Level 3 pre-candidate ABE is pruned, because its subset BE is not a level 2 generator.

Why can ABE be pruned?

Precandidate

1. BE is a subset of ABE and BE is not a generator.
2. BE is not a generator, because it has the same support as its subset B.
3. Since BE has the same support as B, it follows that AB has the same support as ABE.
4. Therefore ABE is not a generator and can be pruned.

General Justification

1. Let XA be a candidate generator, where X is an itemset, and A is a single item.
2. Suppose X is not a generator, because there is some $Y \subset X$ with $s(Y) = s(X)$.
3. Then $s(YA) = s(XA)$ and since $YA \subset XA$, it follows that XA is not a generator.

Apriori-Close: computing generators

Algorithm 2 Apriori-Close(t_1, R, db)

$C(1) \leftarrow R$
$k \leftarrow 1$
while $C(k) \neq \emptyset$ **do**
  $G(k) \leftarrow \emptyset$
  for all $X \in C(k)$ **do**
    if $s(X) \geq t_1$ **and** ($k = 1$ **or** for all $Z \subset X$ of $k-1$ items: $s(X) < s(Z)$) **then**
      $G(k) \leftarrow G(k) \cup \{X\}$
    end if
  end for
  $C(k+1) \leftarrow \emptyset$
  for all $X \in G(k)$ **do**
    for all $Y \in G(k)$ that share the first $k-1$ items with X **do**
      if All $Z \subset X \cup Y$ of k items $\in G(k)$ **then**
        $C(k+1) \leftarrow C(k+1) \cup \{X \cup Y\}$
      end if
    end for
  end for
  $k \leftarrow k+1$
end while

Example: Computing Closures

Generator	Closure	Support
A	AC	3
B	BE	4
C	C	4
E	BE	4
AB	ABCE	2
AE	ABCE	2
BC	BCE	3
CE	BCE	3

$\Longrightarrow$

Closure	Support
AC	3
BE	4
C	4
ABCE	2
BCE	3

$$c(I) = \cap t \in db : I \subseteq t$$

Phase 2

To determine all frequent itemsets and their support, we use two properties:

- All maximal frequent itemsets are closed (proof?)
- The support of an itemset equals the support of the smallest closed itemset in which it is contained (its closure).

Select maximal itemsets, generate all their subsets, and determine their support.

Example: Phase 2

Closure	Support
AC	3
BE	4
C	4
ABCE	2
BCE	3

$\{A, B, C, E\}$ is the only maximal frequent itemset.

Subset	Support
ABC	2
ABE	2
ACE	2

+ the generators and closed itemsets themselves.

Example

Transaction	Items
1	ABCD
2	ABCD
3	ABCD
4	ABCD
5	ABCD
6	BCDE
7	BCDE
8	BCDE
9	BCDE
10	BCDE

Minsup $= 4$. Use A-close to find all closed frequent itemsets and their support.
How does it compare to Apriori?

Example: G_1 and G_2

Generator	Support
A	5
B	10
C	10
D	10
E	5

Candidate	Support
AB	5
AC	5
AD	5
AE	0
BC	10
BD	10
BE	5
CD	10
CE	5
DE	5

All level 2 candidates are pruned, AE because it is infrequent, the remaining itemsets because they have a subset with the same support.

Apriori would only prune AE (and its supersets).

Example: Computing Closures

Generator	Closure	Support
A	ABCD	5
B	BCD	10
C	BCD	10
D	BCD	10
E	BCDE	5

Only 3 closed frequent itemsets.

How many frequent itemsets are there?

Comparison of A-Close with Apriori

- Comparable to Apriori on sparse, weakly correlated data (e.g. supermarket basket data).
- Significantly better on dense, correlated data.
- Why?
- For strongly correlated data, the difference between the number of frequent itemsets, and the number of *closed* frequent itemsets is larger.