Data Mining 2013
Frequent Pattern Mining (1)

Ad Feelders

Universiteit Utrecht

October 8, 2013

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 $X \cap Y = \varnothing$, let:

- $s(X)$ 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 $s(X \cup Y)/s(X)$

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.

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

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

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

Levelwise Search

Find frequent sets

```
$C(1) := R$
$i := 1$
**While** $C(i) \neq \varnothing$ **do**
    $F(i) := \varnothing$
    **For** each $X \in C(i)$ **do**
        **If** $s(X) \geq t_1$ **then** $F(i) := F(i) \cup \{X\}$
    $i := i + 1$
    $C(i) := \varnothing$
    **For** each $X \in F(i-1)$ **do**
        **For** each $Y \in F(i-1)$ that shares $\overbrace{i-2}^{\text{all but one}}$ items with $X$ **do**
            **If** All $Z \subset X \cup Y$ of $i-1$ items are frequent **then**
                $C(i) := C(i) \cup \{X \cup Y\}$
```

Example: the data

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

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

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

Example: Level 2

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

Candidate	Support	Frequent?
AB	4	Yes
AC	4	Yes
AD	1	No
AE	2	Yes
BC	4	Yes
BD	2	Yes
BE	2	Yes
CD	0	No
CE	1	No
DE	0	No

Example: Level 3

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

Candidate	Support	Frequent?
ABC	2	Yes
ABE	2	Yes

Level 4: BCE is not frequent, so we don't have to check ABCE.

Order, order

The algorithm fragment:

```
For each $X \in F(i-1)$ do
  For each $Y \in F(i-1)$ that shares $\overbrace{i-2}^{\text{all but one}}$ items with $X$ do
```

could lead to multiple generations of the set $X \cup Y$.

For example, the candidate ABC is generated 3 times

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

Order, order

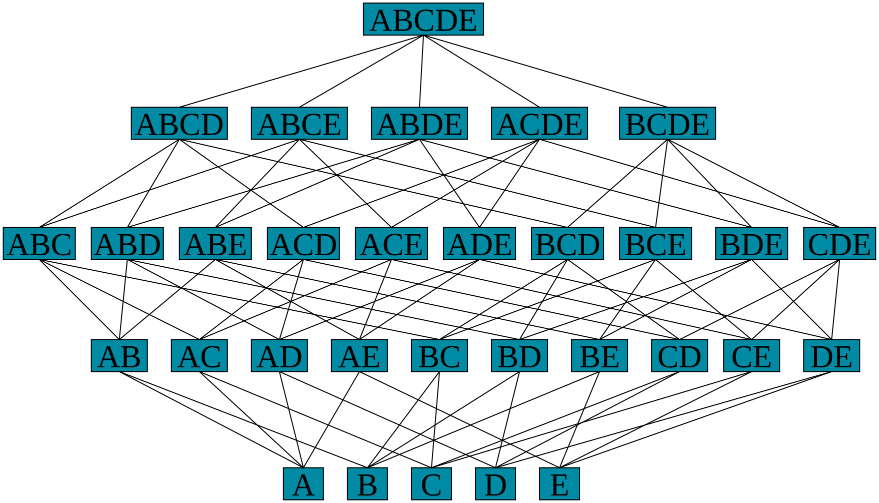
The solution is to place an order on the items.

```
**For** each $X \in F(i-1)$ **do**
    **For** each $Y \in F(i-1)$ that shares $\overbrace{\textit{the first } i-2}^{\text{all but the last one}}$ items with $X$ **do**
```

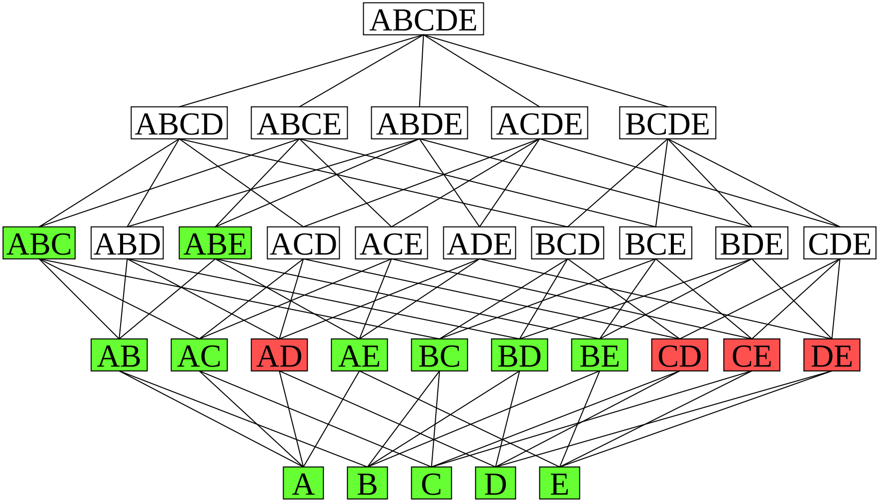
Now the candidate ABC is generated just once, by combining AB with AC.

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

The search space



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 $O(2^{|R|})$!

However, if we *assume* that *db* is *sparse* (by far the most values are 0), then we *expect* that the frequent sets have a maximal size k with $k << |R|$

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

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

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 of the frequent sets is ABE. This 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	$2/2 = 100\%$

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:

- $|Z| \leq k << |R|$
- 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:

- 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

- 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

The *lift* of an association rule tells us how much better the rule predicts the consequent than the random prediction:

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

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

Generating Less Results

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.

Closed Frequent Itemsets

- 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:

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

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

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

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

Closure Operator

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 $c(I) = I$

Running Example

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

Example of Closure Operator

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

Why?

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

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

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

Running Example

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

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

Example of Closure Operator

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

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

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

Running Example

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

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

Example

The closed frequent itemsets for

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

with minimum support 2 are

Example

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 $X \subset Y$ *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 $c(X) = f(\sigma(X)) = f(\sigma(Y)) = c(Y)$.

□

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 $s(X) = s(Z)$. □

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

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

An itemset I is a *generator* of a closed itemset J if it is one of the smallest itemsets with $c(I) = J$.

Example: BC and CE are generators of the closed itemset BCE.

A-Close: Phase 1

Levelwise construction: G_{i+1} is constructed using G_i.

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

The support of BCE is the same as the support of its subsets BC and CE, so they have the same closure.

Example: G_1

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

$\Longrightarrow$

Generator	Support
A	3
B	4
C	4
E	4

Minimum support $= 2$

Example: G_2

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

$\Longrightarrow$

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 candidate ABE is pruned, because its subset BE is not a level 2 generator.

Why can ABE be pruned?

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.

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

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

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

All level 2 generators 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 with Apriori

- Comparable to Apriori on sparse, weakly correlated data (e.g. supermarket basket data).
- Significantly better on dense, correlated data.

Comparison with Apriori

Why more improvement on strongly correlated data?

Comparison with Apriori

Why more improvement on strongly correlated data?

For strongly correlated data, the difference between the number of frequent itemsets, and the number of *closed* frequent itemsets is larger.