Frequent Itemset Mining

prof. dr Arno Siebes

Algorithmic Data Analysis Group
Department of Information and Computing Sciences
Universiteit Utrecht

Monotone Monomials

Recall the class of monotone monomials C_n^+

- the set of conjunctions over n boolean variables
- no negations!

In the lecture on examples of the VC dimension we proved that

- $|C_n^+| = 2^n$ (a variable is in or not)
- $VC\left(C_n^+\right) = n$

And in lecture on PAC learning we saw an easy algorithm to learn a C_n^+ formula ϕ

- each positive example tells you which variables should (at most) be in ϕ
- return the conjunction of all variables that have to be in ϕ

And thus we know that we need

- $\left\lceil \frac{n \log(2/\delta)}{\epsilon} \right\rceil$ examples to PAC learn C_n^+

Why Learn Monotone Monomials?

It is all very nice that we can learn C_n^+

- easily and with nice properties

but why would you want to learn C_n^+

- in a practical application?

The boolean variables could, e.g., stand for

- items that people can buy in a store
- links that visitors of your website can click on
- binary features of molecules

And then, the inferred monomial ϕ would be

- the set of (conjunction of) the articles all people buy
- the set of the links all your visitors click on
- the set of features shared by all molecules

Note that in these examples, all entries in the database

- are most likely positive examples only!

Wait A Minute

These are potentially interesting examples

- but it seems unlikely that there are items in a store that all customers buy
  - at least for stores with a reasonably large collection to choose from
- properties shared by *all* molecules seem rather trivial
  - they might very well embody the definition of what it is to be a molecule

It seems much more likely that there are

- sets of items that are bought by certain classes of customers
  - young parents buying Nappies and Baby Milk
  - professors buying Champagne and Caviare
- sets of features that are shared by certain classes of molecules
  - carcinogenic molecules have benzene rings

That is, it seems much more likely that

- the database is a *mixture* of examples of various monotone monomials

Mixtures of Monotone Monomials

What do we mean when we say that

- the database is a *mixture* of examples of various monotone monomials

We mean that there are two types of mixing going on

1. there is a collection $\phi_1, \ldots, \phi_n$
2. and each entry in the database is a *positive* example of *one or more* of the ϕ_i

We already argued that having the database represent just one monotone monomial ϕ is rather unlikely in any practical application

- but why is it useful if each entry in the database could be an example of multiple ϕ_i?

The reason is actually rather simple

- your customers/visitors/... often fit multiple categories
  - there are professors that are also young parents
  - they buy Nappies and Baby Milk in one role and Champagne and Caviare in the other role *at the same time*

Transaction Databases

The problem of inferring *all* ϕ_i has first been introduced as frequent itemset mining. The setup is as follows

- we have a set of *items* $\mathcal{I} = \{i_1, \ldots, i_n\}$
  - representing, e.g., the items for sale in your store
- a *transaction* t is simply a subset of $\mathcal{I}$, i.e., $t \subseteq \mathcal{I}$
  - or, more precisely, a pair (tid, t) in which $tid \in \mathbb{N}$ is the (unique) tuple id and t is a transaction in the sense above
- Note that there is *no* count of how many copies of i_j the customer bought, just a record of the fact whether or not you bought i_j
  - you can easily represent that in the same scheme if you want
- A database D is a set of transactions
  - all with a unique tid, of course
  - if you don't want to bother with tid's, D is simply a bag of transactions

Frequent Itemsets

Let D be a transaction database over $\mathcal{I}$

- an *itemset* I is a set of items (duh), $I \subseteq \mathcal{I}$
- itemset I *occurs* in transaction (tid, t) if $I \subseteq t$
- the *support* of an itemset in D is the number of transaction it occurs in

$$supp_D(I) = |\{(tid, t) \in D \mid I \subseteq t\}|$$

- note that sometimes the *relative* form of support is used, i.e.,

$$supp_D(I) = \frac{|\{(tid, t) \in D \mid I \subseteq t\}|}{|D|}$$

- An itemset I is called *frequent* if its support is equal or larger than some user defined minimal threshold θ

$$I \text{ is frequent in } D \Leftrightarrow supp_D(I) \geq \theta$$

Frequent Itemset Mining

The problem of frequent itemset mining is given by

> *Given a transaction database D over a set of items $\mathcal{I}$, find all itemsets that are frequent in D given the minimal support threshold θ.*

The original motivation for frequent itemset mining comes from *association rule* mining

- an association rule is given by a pair of disjoint itemsets X and Y ($X \cap Y = \emptyset$), it is denoted by

$$X \to Y$$

- where $P(XY) \geq \theta_1$, is the (relative) support of the rule
  - i.e., the relative $supp_D(X \cup Y) = supp_D(XY) \geq \theta_1$
- and $P(Y|X) \geq \theta_2$ is the confidence of the rule
  - i.e., $\frac{supp_D(XY)}{supp_D(X)} \geq \theta_2$

Association Rules

The motivation of association rule mining is simply the observation that

- people that buy X also tend to buy Y
  - for suitable thresholds θ_1, θ_2
- which may be valuable information for sales and discounts

But then you might think

- *correlation is no causation*
- all you see is correlation

And you are completely right

- but why would the supermarket manager care?
- if he sees that ice cream and swimming gear are positively correlated
- he knows that if sales of the one goes up, so will (likely) the sales of the other
- whether or not there is a causal relation or both are caused by an external factor like nice weather.

Discovering Association Rules

Given that there are two thresholds, discovering association rules is usually a two step procedure

- first discover all frequent itemsets wrt θ_1
- for each such frequent itemset I consider all partitions of I to check whether or not that partition satisfies the second condition
  - actually one should be a bit careful so that you don't consider partitions that cannot satisfy the second requirement
  - which is very similar to the considerations in discovering the frequent itemsets

The upshot is that the difficult part is

- discovering the frequent itemsets

Hence, most of the algorithmic effort has been put

- in exactly that task

Later on it transpired that frequent itemsets

- or, more general, frequent patterns

have a more general use, we will come back to that, briefly, later

Discovering Frequent Itemsets

Obviously, simply checking all possible itemsets to see whether or not they are frequent is not doable

- $2^{|\mathcal{I}|} - 1$ is rather big, even for small stores

Fortunately, there is the A Priori property

$$I_1 \subseteq I_2 \Rightarrow supp_D(I_1) \geq supp_D(I_2)$$

Proof

$$\begin{aligned} \{(tid, t) \in D \mid I_1 \subseteq t\} &= \{(tid, t) \in D \mid I_2 \subseteq I_1 \subseteq t\} \\ &\supseteq \{(tid, t) \in D \mid I_2 \subseteq t\} \end{aligned}$$

since $I_2 \subseteq I_1 \subseteq t$ is a stronger requirement than $I_2 \subseteq t$. So, we have

$$\begin{aligned} supp_D(I_2) &= |\{(tid, t) \in D \mid I_2 \subseteq t\}| \\ &\leq |\{(tid, t) \in D \mid I_1 \subseteq t\}| = supp_D(I_1) \end{aligned}$$

If I_1 is *not* frequent in D, neither is I_2

Levelwise Search

Hence, we know that:

if $Y \subseteq X$ and $supp_D(X) \geq t_1$, then $supp_D(Y) \geq t_1$.
and conversely, if $Y \subseteq X$ and $supp_S(Y) < t_1$, then $supp_D(X) < t_1$.

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(θ, $\mathcal{I}$, D)

$C(1) \leftarrow \mathcal{I}$
$k \leftarrow 1$
while $C(k) \neq \emptyset$ **do**
  $F(k) \leftarrow \emptyset$
  for all $X \in C(k)$ **do**
    if $supp_D(X) \geq \theta$ **then**
      $F(k) \leftarrow F(k) \cup \{X\}$
    end if
  end for
  $C(k+1) \leftarrow \emptyset$
  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

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 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

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.

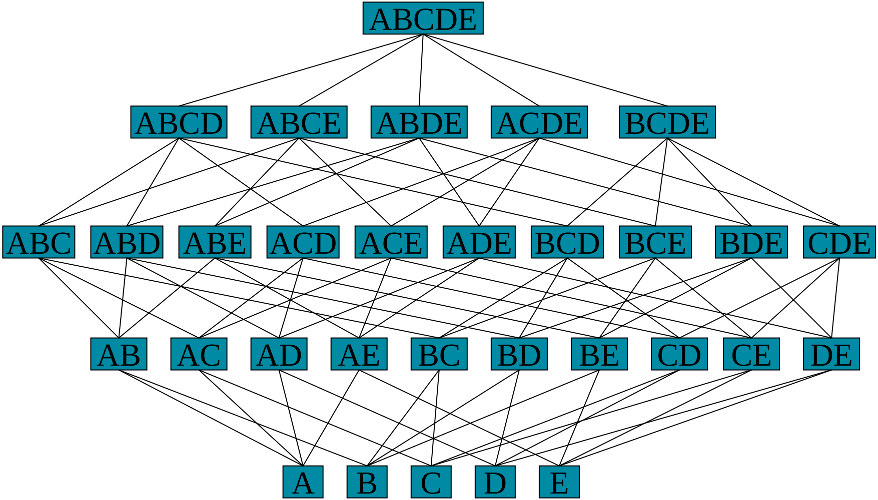
---

for all $X \in F(k)$ **do**
  for all $Y \in F(k)$ that share the first $k-1$ 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

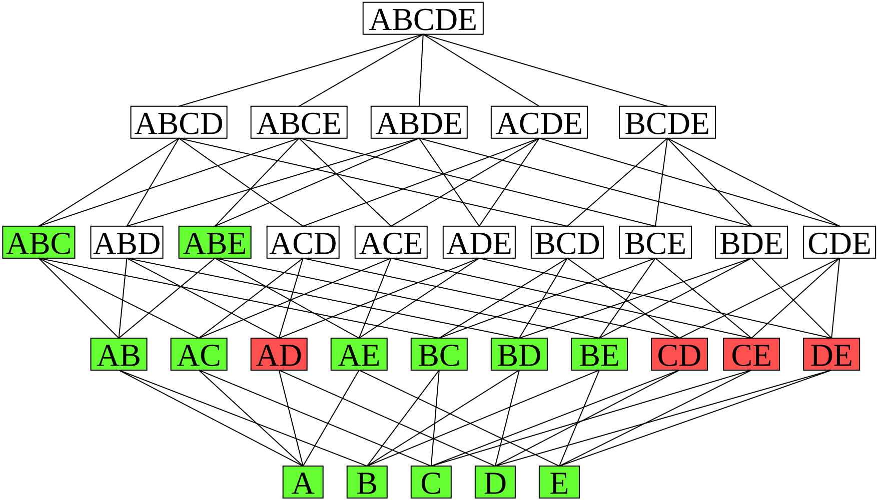
Now the candidate ABC is generated just once, by combining AB with AC.

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

The search space



Item sets counted by Apriori



The Complexity of Apriori

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

However, suppose that D is *sparse* (by far the most values are 0), then we *expect* that the frequent sets have a maximal size m with $m << |\mathcal{I}|$

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

$$O\left(\sum_{j=1}^{m} \binom{|\mathcal{I}|}{j}\right) = O(|\mathcal{I}|^m) << O(2^{|\mathcal{I}|})$$

More General

Apriori is not only a good idea for itemset mining

- it is applicable in pattern mining in general
- provided that some simple conditions are met

To explain this more general setting

- we briefly discuss partial orders
- lattices and
- Galois connections

Partial Orders

A partially ordered set $(X, \preceq)$ consists of

- a set X
- and a partial order $\preceq$ on X
- that is, $\forall x, y, z \in X$:
  1. $x \preceq x$
  2. $x \preceq y \wedge y \preceq x \Rightarrow x = y$
  3. $x \preceq y \wedge y \preceq z \Rightarrow x \preceq z$

An element $x \in X$ is an upperbound of a set $S \subseteq X$ if

$$\forall s \in S : s \preceq x$$

It is the least upperbound, aka join, of S if

$$\forall y \in \{y \in X \mid \forall s \in S : s \preceq y\} : x \preceq y$$

Lowerbounds and greatest lowerbounds, aka meet, are defined dually

Lattices

A partially ordered set $(X, \preceq)$ is a lattice if each two elements subset $\{x, y\} \subseteq X$

- has a join, denoted by $x \vee y$
- and a meet, denoted by $x \wedge y$

If for $S, T \subseteq X$, $\bigvee S, \bigvee T, \bigwedge S$, and $\bigwedge T$ exist, then

- $\bigvee(S \cup T) = (\bigvee S) \vee (\bigvee T)$
- $\bigwedge(S \cup T) = (\bigwedge S) \wedge (\bigwedge T)$

A lattice is bounded if it has a largest element 1, sometimes denoted by $\top$, and a smallest element 0, sometimes denoted by $\bot$:

- $\forall x \in X : 0 \preceq x \preceq 1$

A lattice is complete

- if all its subsets have a join and a meet

Note that it immediately follows that

- each complete lattice is bounded
- each finite lattice is complete

Properties of Lattices

It is easy to see that for any lattice we have that $\vee$ and $\wedge$ are

- idempotent $x \vee x = x \wedge x = x$
- commutative $x \vee y = y \vee x$ and $x \wedge y = y \wedge x$
- associative $x \vee (y \vee z) = (x \vee y) \vee z$ and $x \wedge (y \wedge z) = (x \wedge y) \wedge z$

Moreover, they obey the absorption laws:

- $x \vee (x \wedge y) = x$
- $x \wedge (x \vee y) = x$

Note that idempotency of $\vee$ and $\wedge$ are a direct consequence of the absorption laws

- in fact, they are a special case

Rather than starting from a partial order $\preceq$ one can define lattices algebraically

- with two operators $\vee$ and $\wedge$
- that follow the commutative, associative and absorption laws given above

Two Examples of Lattices

In our discussion of frequent itemset mining we have already met two lattices

1. The itemsets, $(P(\mathcal{I}), \subseteq)$
   - where $\cup$ is the join $\vee$
   - $\cap$ is the meet $\wedge$
   - and the smallest and largest elements are $\emptyset$ and $\mathcal{I}$, respectively
2. Subsets of the database $(P(D), \subseteq)$
   - with the same operators as above
   - and $\emptyset$ and D as minimal and maximal element

Both are finite and complete and we know that they have distributive properties:

- $x \vee (y \wedge z) = x \cup (y \cap z) = (x \cup y) \cap (x \cup z) = (x \vee y) \wedge (x \vee z)$
- $x \wedge (y \vee z) = (a \wedge y) \vee (x \wedge z)$

Both are the nicest type of lattice you can imagine

- as is any subset lattice

Galois Connections

Let $(A, \leq)$ and $(B, \preceq)$ be two partially ordered sets. and let $F : A \to B$ and $G : B \to A$ be two functions

- (F, G) is a monotone Galois connection iff

$$\forall a \in A, b \in B : F(a) \preceq b \Leftrightarrow a \leq G(b)$$

- (F, G) is a anti-monotone (antitone) Galois connection iff

$$\forall a \in A, b \in B : b \preceq F(a) \Leftrightarrow a \leq G(b)$$

In the monotone case we have for the *closure* operators $GF : A \to A$ and $FG : B \to B$ that

- $a \leq GF(a)$ and $FG(b) \preceq b$

While in the anti-monotone case we have for these closure operators that

- $a \leq GF(a)$ and $b \preceq FG(b)$

A Galois Connection

There is an easy Galois connection between the two lattices $(P(\mathcal{I}), \subseteq)$ and $(P(D), \subseteq)$:

- define $F : P(\mathcal{I}) \to P(D)$ by

$$F(I) = \{t \in D \mid I \subseteq t\}$$

- define $G : P(D) \to P(\mathcal{I})$ by

$$\begin{aligned} G(E) &= \{i \in I \mid \forall t \in E : i \in t\} \\ &= \bigcap_{t \in E} t \end{aligned}$$

Now, note that for $I \in P(\mathcal{I})$ and $E' \in P(D)$ we have that

$$[E' \subseteq F(I)] \Leftrightarrow [\forall t \in E' : I \subseteq t] \Leftrightarrow \left[I \subseteq \bigcap_{t \in E'} t\right] \Leftrightarrow [I \subseteq G(E')]$$

That is, the connection is anti-monotone

Closed Itemsets

With these F and G, we have the mapping

$$GF : P(\mathcal{I}) \to P(\mathcal{I})$$

If an itemset $I \in P(\mathcal{I})$ is a fixed point of GF

$$GF(I) = I$$

then I is called a *closed* itemset.

It is easy to see that $I \in P(\mathcal{I})$ is closed iff

- $\forall i \in \mathcal{I} : i \notin I \to supp_D(I \cup \{i\}) < supp_D(I)$

Call an itemset $J \in P(\mathcal{I})$ maximal iff

- J is frequent in D
- $\forall K \in P(\mathcal{I}) : J \subset K \to K$ is *not* frequent

The we have

- maximal itemsets are frequent

A Condensed Representation

Let C be the set of all closed frequent item sets and let $J \in P(\mathcal{I})$.

- if $\forall I \in C : J \not\subset I$ then J is not frequent
  - there is a maximal $K \in C$ such that $K \subset J$ and thus J is not frequent
- if $\exists I \in C : J \subset I$, then J is frequent and we know its frequency
  - just look at the frequency of all $I \in C : J \subset I$ and take the .... frequency of those. Since that that itemset is frequent, so is J.

In other words C tells you all there is to know about the set of frequent itemsets.

- it is a condensed representation of the set of all frequent itemsets

The Power of Anti-Monotone

The reason that the A Priori principle holds

- and thus that the Apriori algorithm works

is that the Galois connection between $P(\mathcal{I})$ and $P(D)$ is anti-monotone, because that means that

- $I_1 \subseteq I_2 \Rightarrow F(I_1) \supseteq F(I_2)$
- and $supp_D(I) = |F(I)|$

In other words, we can use the Apriori Algorithm on *any* anti-monotone Galois Connection.

We'll explain this in more detail on the following few slides following Manilla and Toivonen, Levelwise Search and Borders of Theories in Knowledge Discovery, DMKD, 1997.

Theory Mining

Given a language $\mathcal{L}$

- for defining subgroups of the database
  - one example is $\mathcal{L} = P(\mathcal{I})$

and a predicate q that

- determines whether or not $\phi \in \mathcal{L}$ describes an interesting subset of D
- i.e., whether or not $q(\phi, D)$ is true or not
  - an example of q is $supp_D(I) \geq \theta$

The task is to compute the theory of D with respect to $\mathcal{L}$ and q. That is, to compute

$$\mathcal{T}\langle(\mathcal{L}, D, q) = \{\phi \in \mathcal{L} \mid q(\phi, D)\}$$

Now, if $\mathcal{L}$ is a finite set with a partial order $\preceq$ such that

$$\psi \preceq \phi \Rightarrow [q(\phi, D) \rightarrow q(\psi, D)]$$

we have the anti-monotonicity to use Apriori

Queries and Consequences

Since $\mathcal{L}$ defines subgroups of the database

- it is essentially a query language

Most query languages naturally have a partial order

- either "syntactically" $D \vdash \phi \rightarrow D \vdash \psi$
- or semantically $D \vDash \phi \rightarrow D \vDash \psi$
- or both can be used (think of monomials)

Furthermore, note that query languages can be defined for many different types of data, e.g.,

- graphs
- data streams
- text

For all these types, and many more, we can define pattern languages

- and compute all frequent patterns using levelwise search.

Complexity: the database perspective

We only looked at the complexity wrt the number of items of our table. But, that is not the only aspect: what about the role of the database?

- If we check each itemset separately, we need as many passes over the database as there are candidate frequent sets.
- If at each level we first generate all candidates and check all of them in one pass, we need as many passes as the size of the largest candidate set.

If the database does not fit in main memory, such passes are costly in terms of I/O.

Can we use sampling? Of course we can!

Toivonen, Sampling Large Databases for Association Rules, VLDB 96

Mining from one Sample

If we mine just one sample for item sets, we will make mistakes:

- we will *find* sets that *do not hold* on the complete data set
- we will *miss* sets that *do hold* on the complete data set

Clearly, the probability of such errors depend on the size of the sample.

Can we say something about this probability and its relation to the size?

Of course we can, remember our old friend Hoeffding.

Binomial Distribution and Hoeffding Bounds

An experiment with two possible outcomes is called a Bernoulli experiment. Let's say that the probability of *success* is p and the probability of *failure* is $q = 1 - p$.
If X is the random variable that denotes the number of successes in n trials of the experiment, then X has a *binomial distribution*:

$$P(X = m) = \binom{n}{m} p^m (1 - p)^{n-m}$$

In n experiments, we expect pn successes, How likely is it that the measured number m is (many) more or less? One way to answer this question is via the *Hoeffding bounds:*

$$P(|pn - m| > \epsilon n) \leq 2e^{-2\epsilon^2 n}$$

Or (divide by n)

$$P(|p - \tfrac{m}{n}| > \epsilon) \leq 2e^{-2\epsilon^2 n}$$

Sampling with replacement

Let

- p denote the support of Z on the database.
- n denote the sample size.
- m denote the number of transactions in the sample that contain all items in Z.

Hence $\hat{p} = \frac{m}{n}$ is our sample-based estimate of the support of Z.

The probability that the difference between the true support p and the estimated support $\hat{p}$ is bigger than ϵ is bounded by

$$P(|p - \hat{p}| > \epsilon) \leq 2e^{-2\epsilon^2 n}$$

The Sample Size and the Error

If we want to have:

$$P(|p - \hat{p}| > \epsilon) < \delta$$

(estimate is probably (δ) approximately (ϵ) correct).

Then, we have to choose n such that:

$$\delta \geq 2e^{-2\epsilon^2 n}$$

Which means that:

$$n \geq \frac{1}{2\epsilon^2} \ln \frac{2}{\delta}$$

Example

To get a feeling for the required sample sizes, consider the following table:

ϵ	δ	n
0.01	0.01	27000
0.01	0.001	38000
0.01	0.0001	50000
0.001	0.01	2700000
0.001	0.001	3800000
0.001	0.0001	5000000

Lowering the Threshold

If we want to have a low probability (say, μ) that we miss item sets on the sample, we can mine with a lower threshold t'. How much lower should we set it for a given sample size?

$$P(p - \hat{p} > \epsilon) \leq e^{-2\epsilon^2 n}$$

Thus, if we want $P(\hat{p} < t') \leq \mu$, we have:

$$\begin{aligned} P(\hat{p} < t') &= P(p - \hat{p} > \overbrace{p - t'}^{\epsilon}) \\ &\leq e^{-2(p-t')^2 n} = \mu \end{aligned}$$

Which means that

$$t' = p - \sqrt{\frac{1}{2n} \ln \frac{1}{\mu}}$$

In other words, we should lower the threshold by $\sqrt{\frac{1}{2n} \ln \frac{1}{\mu}}$

Mining Using a Sample

The main idea is:

- Draw (with replacement) a sample of sufficient size
- Compute the set FS of all frequent sets on this sample, using the lowered threshold.
- Check the support of the elements of FS on the complete database

This means that we have to scan the complete database only once. Although, taking the random sample may require a complete database scan also!

Did we miss any results?

There is the possibility that we miss frequent sets. Can we check whether we are missing results in the same database scan?
If $\{A\}$ and $\{B\}$ are frequent sets, we have to check the frequency of $\{A, B\}$ in the next level of level-wise search.
This gives rise to the idea of the *border* of a set of frequent sets:

Definition

Let $S \subseteq \mathcal{P}(R)$ be closed with respect to set inclusion. The border $Bd(S)$ consists of the minimal itemsets $X \subseteq R$ which are not in S.

Example: Let $R = \{A, B, C\}$. Then

$$\mathrm{Bd}(\{\{A\}, \{B\}, \{C\}, \{A, B\}, \{A, C\}\}) = \{\{B, C\}\}$$

The set of frequent itemsets is obviously closed with respect to set inclusion.

On the Border

Theorem Let FS be the set of all frequent sets on the sample (with or without the lowered threshold). If there are frequent sets on the database that are not in FS, then at least one of the sets in Bd(FS) is frequent.

Proof Every set not in FS is a superset of one of the border elements of FS. So if some set not in FS is frequent, then by the A Priori property, one of the border elements must be frequent as well.

So, if we check not only FS for frequency, but FS $\cup$ Bd(FS) and warn when an element of Bd(FS) turns out to be frequent, we know that we might have missed frequent sets.

Finding Frequent Itemsets

Algorithm 2 Sampling-Border Algorithm

```
FS ← set of frequent itemsets on the sample
PF ← FS ∪ Bd(FS) {Perform first scan of database}
F^(0) ← {I : I ∈ PF and I frequent on D}
NF ← PF \ F^(0) {Create candidates for second scan}
if F^(0) ∩ Bd(FS) ≠ ∅ then
    repeat
        F^(i) ← F^(i−1) ∪ (Bd(F^(i−1)) \ NF)
    until no change to F^(i)
end if {Perform second scan}
F ← F^(0) ∪ {I : I ∈ F^(i) \ F^(0) and I frequent on D}
return F
```

Can We Do Better?

We computed the

- errors, sample sizes, and so on

directly using Hoeffding's bound

That must be a bit disappointing to you

- we spend weeks building up a complete theoretical framework
- quite often using this same Hoeffding bound
- and at the first opportunity, we don't use any of it at all!

So, the $64000 question is

Can we do better?

Next time we show we can, using our PAC learning framework