Association Analysis

Market-basket analysis

Data

- A large set of **items**: e.g., products sold in a supermarket
- A large set of **baskets**: e.g., each basket represents what a customer bought in one visit to the supermarket

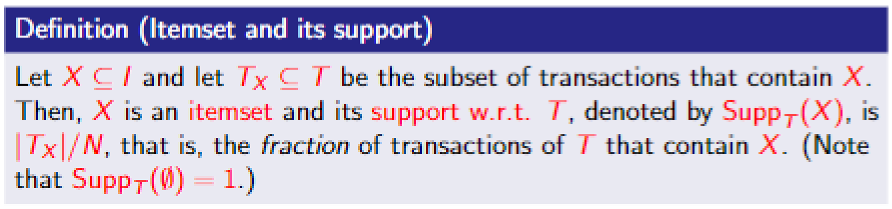
Goal

Analyze data to extract

- **Frequent itemsets**: subsets of items that occure together in a high number of baskets
- **Association rules**: correlation between subsets of items.

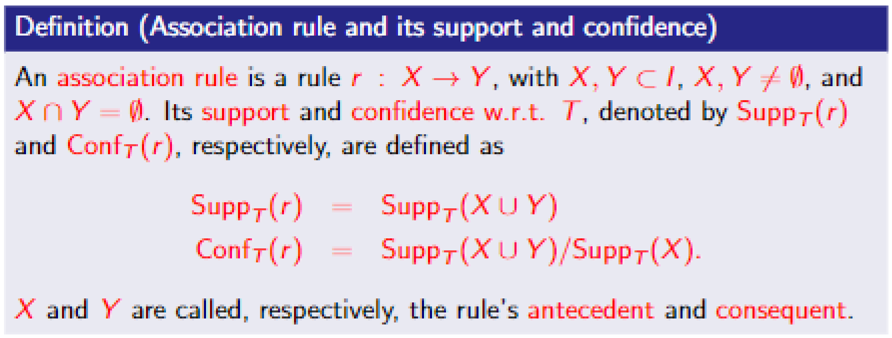
Rigorous formulation of the problem

Dataset T = {t_1,t_2,...,t_N} of *N* transaction (i.e. baskets) over a set *I* of *d* items, with t_i subset of *I*, for 1<=*i*<=*N*



Definition (Itemset and its support)

Let $X \subseteq I$ and let $T_X \subseteq T$ be the subset of transactions that contain X. Then, X is an itemset and its support w.r.t. T, denoted by $\text{Supp}_T(X)$, is $|T_X|/N$, that is, the *fraction* of transactions of T that contain X. (Note that $\text{Supp}_T(\emptyset) = 1$.)



Definition (Association rule and its support and confidence)

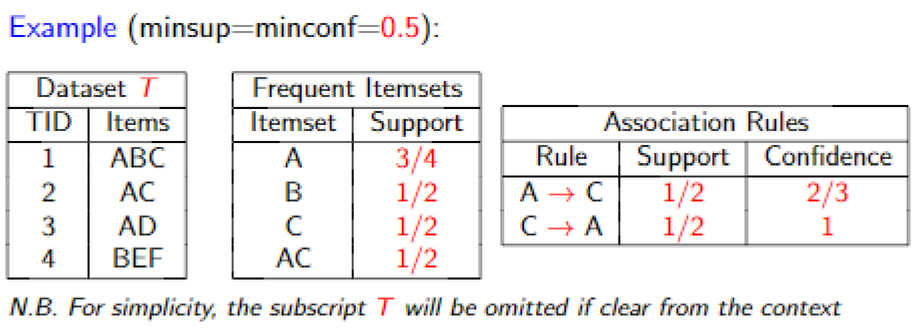
An association rule is a rule $r : X \rightarrow Y$, with $X, Y \subset I$, $X, Y \neq \emptyset$, and $X \cap Y = \emptyset$. Its support and confidence w.r.t. T, denoted by $\text{Supp}_T(r)$ and $\text{Conf}_T(r)$, respectively, are defined as

$$\begin{aligned}\text{Supp}_T(r) &= \text{Supp}_T(X \cup Y)\\ \text{Conf}_T(r) &= \text{Supp}_T(X \cup Y)/\text{Supp}_T(X).\end{aligned}$$

X and Y are called, respectively, the rule's antecedent and consequent.

Given the dataset *T* of *N* transactions over *I*, and given a support threshold minsup in (0,1], and a confidence threshold minconf in (0,1], The following two objectives can be pursued:

1. Compute all frequent itemsets, that is, the set of itemsets *X* such that Supp_T(*X*)>=minsup. We denote this set F_{T,minsup}
2. Compute all association rules *r* such that Supp_T(*r*)>=minsup and Conf_T(*r*)>=minconf.


Example (minsup=minconf=0.5):

Dataset T	
TID	Items
1	ABC
2	AC
3	AD
4	BEF

Frequent Itemsets	
Itemset	Support
A	3/4
B	1/2
C	1/2
AC	1/2

Association Rules		
Rule	Support	Confidence
$A \rightarrow C$	1/2	2/3
$C \rightarrow A$	1/2	1

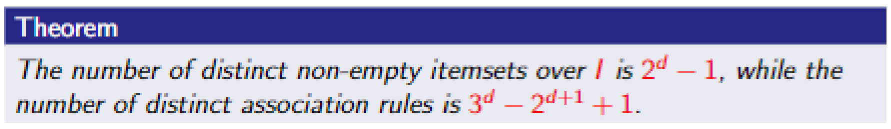
N.B. For simplicity, the subscript T will be omitted if clear from the context

Observations:

- **Support and confidence measure the interestingness** of a pattern. In particular, the threshold *minsup* and *minconf* define which patterns must be regarded as **interesting**.
- Ideally, we would like that the support and confidence of the returned patterns be **unlikely to be seen in a random dataset**. However, what is a random dataset?
- The choice of minsup and minconf is crucial since it directly influences
  - **Output size**: low thresholds may yield too many patterns which become hard to exploit effectively
  - **False positive/negatives**: low thresholds may yield a lot of uninteresting patterns, while high thresholds may miss some interesting patterns

Potential output explosion

Let *I* be a set of *d* items.


Theorem

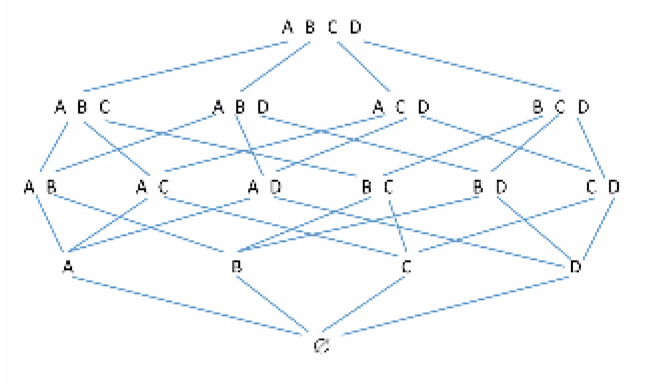
The number of distinct non-empty itemsets over I is $2^d - 1$, while the number of distinct association rules is $3^d - 2^{d+1} + 1$.

- Strategies that enumerate of all itemsets/rules in order to find the interesting ones are out of question een for ground sets *I* of small size
- As a first approximation, we consider efficient strategies that require time/space polynomial in both the input and the output sizes.

Lattice of Itemsets

- The family of itemsets under subset forms a **lattice**, namely a partially ordered set where for each two elements *X, Y* there exists a unique least upper bound (*X union Y*) and a unique greater lower bound (*X intersect Y*)

- The lattice can be represented through the **Hasse diagram**



Anti-monotonicity of Support

The support function for itemsets exhibits the following property, referred to as **anti-monotonicity**. For every *X, Y subset of I*

$$X \subseteq Y \Rightarrow \mathrm{Supp}(X) \geq \mathrm{Supp}(Y).$$

Immediate consequence. For a given support threshold, we have

1. *X* is frequent => for all *W* subset of *X*, *W* is frequent
2. *X* is not frequent => for all *W* subset of *X*, *W* is not frequent

This implies that, in the lattice, frequent itemsets form a sublattice closed downwards

Efficient mining of *F.I.* and *A.R.*

Key objectives

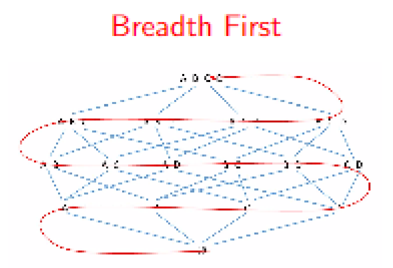
- Careful exploration of the lattice of itemsets exploiting anti-monotonicity of support
- Time/space complexities polynomial in the input and output size

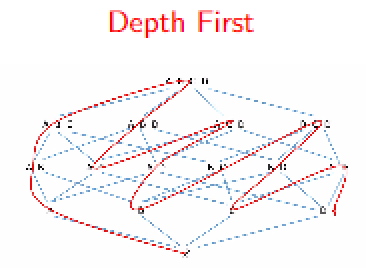
Two phases

1. Compute the set *F* of all frequent itemsets w.r.t. minsup
2. For each itemset *Z* in *F*, compute all rules *r : X->Y* with *X union Y* = *Z* and confidence at least minconf.

Observation: Phase 1 is, usually, the most demanding, computationally

Two main approaches





F.I. mining: A-Priori algorithm

Uses the breadth-first approach

For every itemset *X subset of I*, define its absolute support

$\sigma(X) = \text{Supp}(X) \cdot N$

MAIN ALGORITHM

Input Dataset T of N transactions over I, minsup

Output $F_{T,\text{minsup}} = \{(X, \text{Supp}(X)) \ : \ X \subseteq I \wedge \text{Supp}(X) \geq \text{minsup}\}$

```
$k \leftarrow 1$
Compute $F_1 = \{i \in I \, ; \, \texttt{Supp}(\{i\}) \geq \texttt{minsup}\}$
Compute $O_1 = \{(X, \texttt{Supp}(X)) \, ; \, X \in F_1\}$
repeat
   $k \leftarrow k + 1$
   $C_k \leftarrow$ APRIORI-GEN$(F_{k-1})$ /* Candidates */
   for each $c \in C_k$ do $\sigma(c) \leftarrow 0$
   for each $t \in T$ do
      for each $c \in C_k$ do
         if $c \subseteq t$ then $\sigma(c) \leftarrow \sigma(c) + 1$
   $F_k \leftarrow \{c \in C_k \ : \ \sigma(c) \geq N \cdot \texttt{minsup}\}$;
   $O_k \leftarrow \{(X, \sigma(X)/N) \, ; \, X \in F_k\}$
until $F_k = \emptyset$
return $\bigcup_{k \geq 1} O_k$
```

APRIORI-GEN(F)

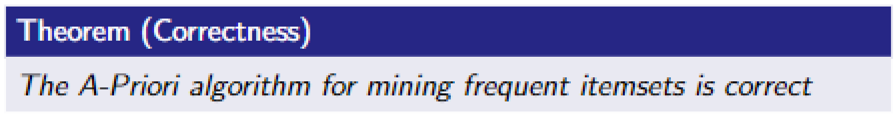
```
Let $\ell - 1$ be the size of each itemset in $F$
$\Phi \leftarrow \emptyset$
/* Candidate Generation */
for each $X, Y \in F$ s.t. $X \neq Y \wedge X[1 \ldots \ell-2] = Y[1 \ldots \ell-2]$ do
   add $X \cup Y$ to $\Phi$
/* Candidate Pruning */
for each $Z \in \Phi$ do
   for each $Y \subset Z$ s.t. $|Y| = \ell - 1$ do
      if $(Y \notin F)$ then {remove $Z$ from $\Phi$; exit inner loop}
return $\Phi$
```

Observations:

1. Candidate generation ensures that no itemset is generated twice
2. Candidate pruning removes itemsets that can be deemed apriori as not being frequent. This explains the name of the algorithm

Correctness of A-priori

We assume the existence of a total odering of the items, and assume the transaction/itemsets are represented as sorted vectors

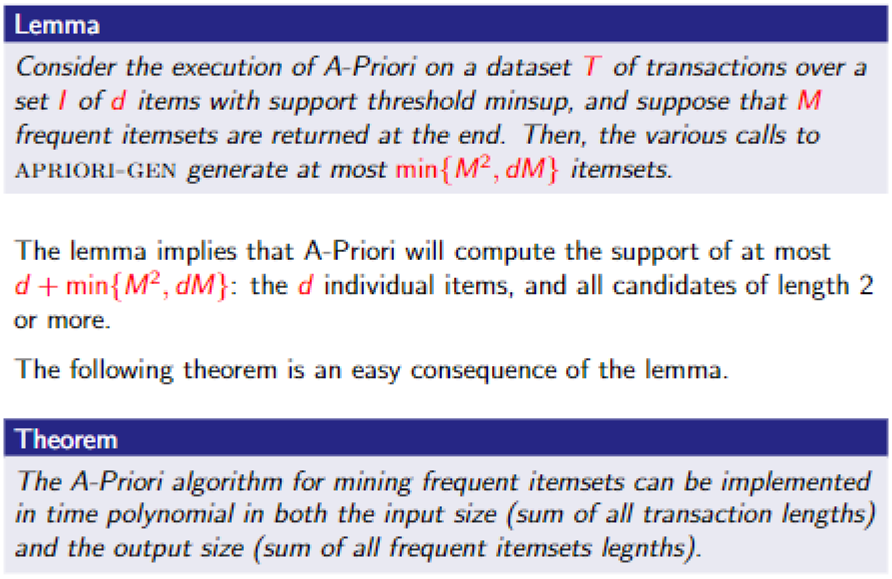

Theorem (Correctness)

The A-Priori algorithm for mining frequent itemsets is correct

Efficiency of A-priori

A-priori owes its popularity to a number of features that yield efficient running times especially when there aren't many frequent itemsets

- A few passes over the dataset are needed: namely k_{max} +1 passes, where k_{max} is the length of the longest frequent itemset. Note that if the number of frequent itemsets is small, k_{max} must also be small
- Exploiting the antimonotonicity of support, candidate generation and pruning ensure that only a few non-frequent itemsets are ever considered and their support computed
- Computing the supports of the candidates, usually the most time-consuming step, can be optimized is many ways


Lemma

Consider the execution of A-Priori on a dataset T of transactions over a set I of d items with support threshold minsup, and suppose that M frequent itemsets are returned at the end. Then, the various calls to APRIORI-GEN *generate at most* $\min\{M^2, dM\}$ *itemsets.*

The lemma implies that A-Priori will compute the support of at most $d + \min\{M^2, dM\}$: the d individual items, and all candidates of length 2 or more.

The following theorem is an easy consequence of the lemma.

Theorem

The A-Priori algorithm for mining frequent itemsets can be implemented in time polynomial in both the input size (sum of all transaction lengths) and the output size (sum of all frequent itemsets legnths).

Optimizations of A-priori: frequent pairs

- The support counting for the candidates in C_k, for k >= 2, is typically the most time-consuming step because:
    1. requires a pass over the entire dataset
    2. may use much space
- In practice, the issue of space may become critical for C_2, which contains all pairs of frequent items. As *k* grows larger, the cardinality of F_{k-1}, hence of C_k drops

GOAL

Compute F_1 efficiently and, at the same time, gather statistics for filtering out infrequent pairs. Consider an istancewith *N* transactions, *d* items, and threshold minsup

- Let *h* be a hash function that maps pairs of items to integers in [0,*K-1*], for a suitable value *K*

- Use *d* + *K* counters: one counter γ_i for each *i in I* and a counter δ_j, for every 0<=*j*<=*K*. Counters are initialized to 0.

- For every transaction *t* do

    - For each item *i in t*, increment γ_i
    - For each pair of items *i_1,i_2 in t* increment δ_{h(i_1,i_2)}

- **Key remark**: only pairs of items *i_1,i_2* such that δ_{h(i_1,i_2)} >= N*minsup have a chance to be frequent

- Compute F_1 = {*i in I : γ_i >= N*minsup*}

- Compute C_2 as the set of pairs *i_1,i_2* such that:

$$(i_1, i_2 \in F_1) \quad \text{AND} \quad (\delta_{h(i_1,i_2)} \geq N \cdot \text{minsup})$$

Observations:

- THe first condition yields the same set of candidates as APRIORI-GEN, while the second condition aims at filtering out some of these candidates
- if *K* is chosen sufficiently large, hence many pair counters are used, then filtering out of infrequent pairs become quite effective

Other optimizations

A large body of litterature has investigated several additional strategies to optimize the mining of frequent itemsets.

- A data structure trie-like was defined by the original developers of A-priori, to be used for storing the set of candidates C_k so to speed up their support counting. In essence, for each transaction, the hash tree quickly provides a subset of the candidates to be checked for inclusion
- Several implementations of depth-first mining strategies have been devised and tested. Their goal is to avoid several passes over the entire dataset of transactions, which may be huge, and to confine the supportcounting of longer itemsets to suitable projections of the dataset, typically much smaller than the original one.

Mining association rules

- Once the frequent itemsets and their supports have been computed all association rules which are based ont these itemsets and satisfy the given confidence requirement can be determined

- Let minconf be the given confidence threshold. For each frequent itemset *Z*, we must determine the set:

$$\{r \ : \ Z - Y \rightarrow Y \ \text{ s.t. } \ \emptyset \neq Y \subset Z \ \wedge \ \text{Conf}(r) \geq \text{minconf}\}$$

- Note that each rule in the above set has the same support as *Z*, hence it automatically satisfies the support constraints since *Z* is frequent. Conversely, rules derived from itemsets which are not frequent need not to be checked, since they would not satisfy the support constraint

- Checking all non-empty subsets *Y subset Z* as RHS of possible rules with confidence at least minconf may be too costly. We exploit a sort of anti-monotonicity properly for rules, as we did for frequent itemsets.
- Anti-monotonicity property for rules. For *emptyset != Y' subset Y subset Z* we have

$$\frac{\mathsf{Supp}(Z)}{\mathsf{Supp}(Z-Y)} \leq \frac{\mathsf{Supp}(Z)}{\mathsf{Supp}(Z-Y')}$$

- Immediate consequence

$$\mathsf{Conf}(Z-Y' \to Y') < \mathsf{minconf} \Rightarrow \mathsf{Conf}(Z-Y \to Y) < \mathsf{minconf}.$$

  Thus for each itemset *Z* it is convenient to check rules with RHS of progressively increasing size

Algorithm for mining association rules

Let *O* = set of frequent itemset and their supports

```
Input $O$, minconf
Output $\{(r, \mathtt{Supp}(r), \mathtt{Conf}(r)) \ : \ \mathtt{Supp}(r) \geq \mathtt{minsup} \wedge \mathtt{Conf}(r) \geq \mathtt{minconf}\}$

$R \leftarrow \emptyset$
for each $Z$ s.t. $|Z| > 1 \wedge (Z, \mathtt{support}(Z)) \in O$ do
    $R \leftarrow R \cup$ AP-GENRULES$(Z)$
return $R$
```

```
$m \leftarrow 1$
$H_{Z,1} \leftarrow \{Y \subset Z \ : \ |Y| = 1 \wedge \mathtt{Supp}(Z)/\mathtt{Supp}(Z-Y) \geq \mathtt{minconf}\}$
$R_{Z,1} \leftarrow \{(r, \mathtt{Supp}(r), \mathtt{Conf}(r)) \ : \ r : Z-Y \to Y, \ \mathtt{with} \ Y \in H_{Z,1}\}$
repeat
    if $(m+1 = |Z|)$ then break
    $m \leftarrow m+1$
    $H_{Z,m} \leftarrow$ APRIORI-GEN$(H_{Z,m-1})$
    $R_{Z,m} \leftarrow \emptyset$
    for each $Y \in H_{Z,m}$ do
        if $(\mathtt{Supp}(Z)/\mathtt{Supp}(Z-Y) \geq \mathtt{minconf})$
            then add $(r : Z-Y \to Y, \mathtt{Supp}(r), \mathtt{Conf}(r))$ to $R_{Z,m}$
            else remove $Y$ from $H_{Z,m}$
until $H_{Z,m} = \emptyset$
return $\bigcup_{m \geq 1} R_{Z,m}$
```

Efficiency of the A.R. algorithm

- The algorithm does not require access to the dataset *T* but only to the frequent itemsets and their supports. If the frequent itemsets are not too many, as one qould hope when the support threshold is properly chosen, avoiding the access to *T* may yield substantial performance gains.

- The use of APRIORI-GEN avoids that for any frequent itemset the confidence of an extremely large number of rules be checked
- It can be easily shown that the algorithm can be implemented in time polynomial in both the input size and output size

Frequent itemsets mining for big data

When the dataset *T* is very large one can follow two approaches

1. **Partition-based approach**: avoid multiple passes over the dataset by **partitioning *T* into subsets**, mining frequent itemsets independently in each subset, and combining the results
2. **Sampling approach**: compute the frequent itemsets from a small sample of *T* and show that they provide a *suitable approximation* to the exact set.

Partition-based approach

1. Partition *T* arbitrarily into *K* subsets of _O(N/K) transactions each, and compute the set of frequent itemsets with regard to minsup independently in each subset
2. Gather al frequent itemsets computed in round 1 and eliminate duplicates. Call Φ the resulting set of itemsets
3. For every 0 < j < *K* independently do the following:

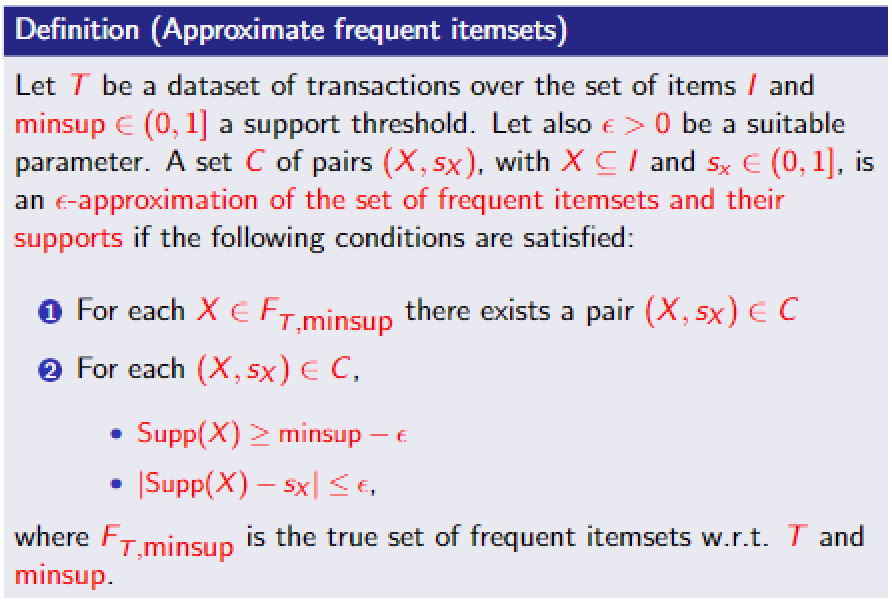
gather a copy of Φ and *T_j* and compute, for each *X in Φ*, the number of transactions of *T_j* that contain *X* and call it *σ(X,j)* 4. For each *X in Φ*, gather all *σ(X,j)*, compute the final support and output *X* if Supp(X)>=minsup

Partition-based approach: analysis

- **Correctness**: it follows from the fact that each itemset frequent in *T* must be frequent in some *T_j*. In other words, Φ contains all final frequent itemsets, although it may contain many more
- **Number of rounds**: 4
- **Space requirements**: they mainly depend on the size of Φ, which cannot be easily predictedm and the algorithm used to extract the frequent itemsets in the first round

Remark: while the algorithm may work well in practice, it does not feature strong theoretical guarantees

Sampling-based approach



Definition (Approximate frequent itemsets)

Let T be a dataset of transactions over the set of items I and minsup $\in (0,1]$ a support threshold. Let also $\epsilon > 0$ be a suitable parameter. A set C of pairs (X, s_X), with $X \subseteq I$ and $s_x \in (0,1]$, is an ϵ-approximation of the set of frequent itemsets and their supports if the following conditions are satisfied:

1. For each $X \in F_{T,\text{minsup}}$ there exists a pair $(X, s_X) \in C$
2. For each $(X, s_X) \in C$,
   - $\text{Supp}(X) \geq \text{minsup} - \epsilon$
   - $|\text{Supp}(X) - s_X| \leq \epsilon$,

where $F_{T,\text{minsup}}$ is the true set of frequent itemsets w.r.t. T and minsup.

- Condition (1) ensures that the approximate set C comprises all true frequent itemsets
- Condition (2) ensures that: (a) C does not contain intemsets of very low support; and (b) for each itemset *X* such that *(X,s_X) in* C, *s_X* is a good estimate of its support.

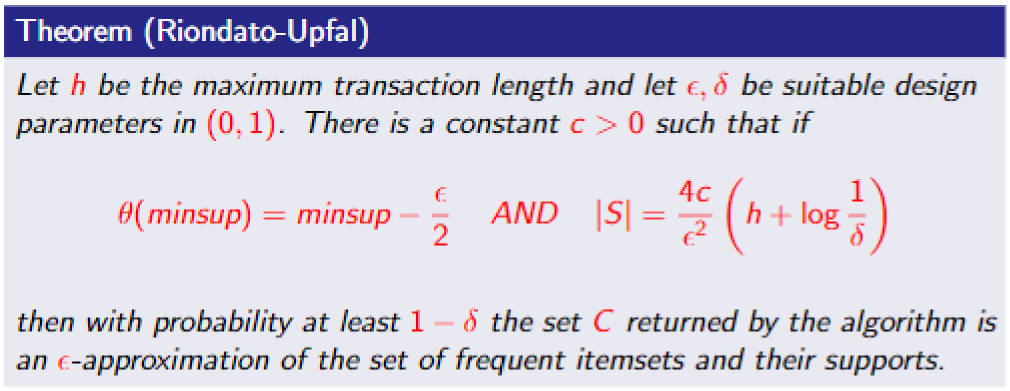
Let *T* be a dataset of *N* transactions over *I*, and minsup in (0,1] a support threshold. Let also θ(minsup) < minsup be a suitably lower support threshold

- Let *S subset T* be a sample drawn at random with replacement with uniform probability and with replacement

- Return the set of pairs

$$C = \left\{ (X, s_x = \text{Supp}_S(X)) \ : \ X \in F_{S,\theta(\text{minsup})} \right\},$$

  where F_{S,θ(minsup)} is the set of frequent itemsets with regard to *S* and θ(minsup)



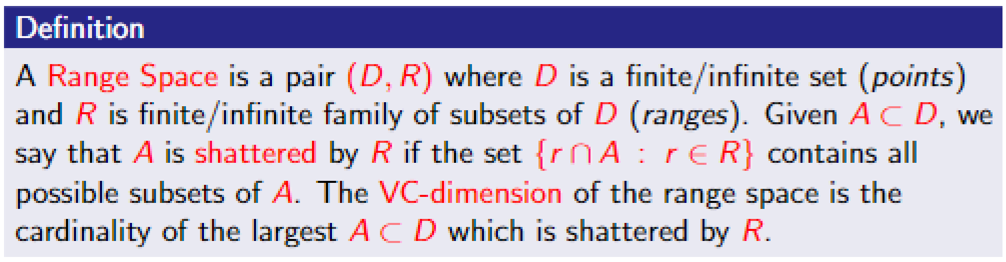
Theorem (Riondato-Upfal)

Let h be the maximum transaction length and let ϵ, δ be suitable design parameters in $(0,1)$. There is a constant $c > 0$ such that if

$$\theta(\textit{minsup}) = \textit{minsup} - \frac{\epsilon}{2} \quad \textit{AND} \quad |S| = \frac{4c}{\epsilon^2}\left(h + \log\frac{1}{\delta}\right)$$

then with probability at least $1 - \delta$ the set C returned by the algorithm is an ϵ-approximation of the set of frequent itemsets and their supports.

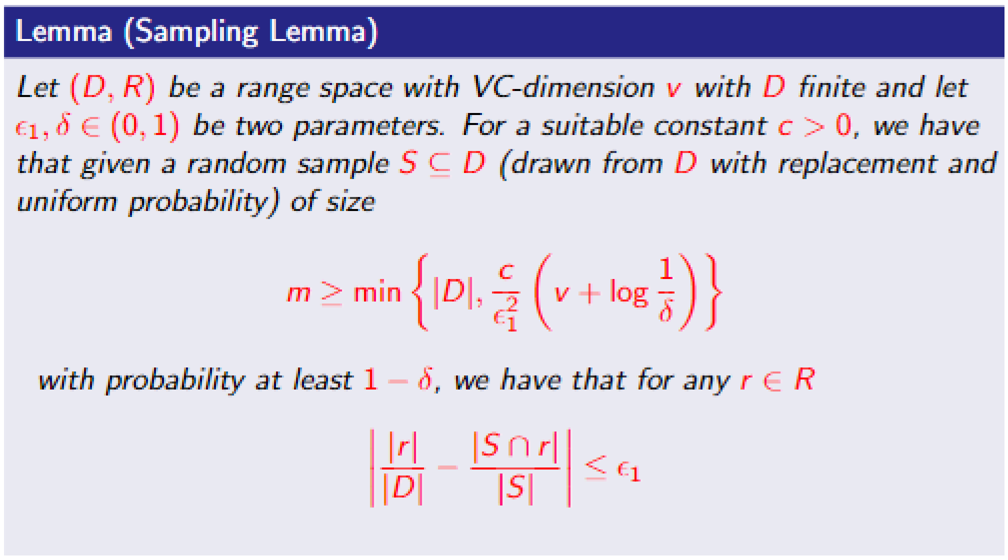
VC-dimension



Definition

A Range Space is a pair (D, R) where D is a finite/infinite set (*points*) and R is finite/infinite family of subsets of D (*ranges*). Given $A \subset D$, we say that A is shattered by R if the set $\{r \cap A \; : \; r \in R\}$ contains all possible subsets of A. The VC-dimension of the range space is the cardinality of the largest $A \subset D$ which is shattered by R.

Analysis of sampling-based approach



Lemma (Sampling Lemma)

Let (D, R) be a range space with VC-dimension v with D finite and let $\epsilon_1, \delta \in (0, 1)$ be two parameters. For a suitable constant $c > 0$, we have that given a random sample $S \subseteq D$ (drawn from D with replacement and uniform probability) of size

$$m \geq \min \left\{ |D|, \frac{c}{\epsilon_1^2} \left(v + \log \frac{1}{\delta} \right) \right\}$$

with probability at least $1 - \delta$, we have that for any $r \in R$

$$\left| \frac{|r|}{|D|} - \frac{|S \cap r|}{|S|} \right| \leq \epsilon_1$$

A dataset *T* of transactions over *I* can be seen as a range space (D,R):

- *D = T*
- *R = {T_X: X subset I and X != emptyset}*, where *T_X* is the set of transactions that contain *X*

It can be shown that the VC-dimension of (D,R) is <= *h*, where *h* is the maximum transaction length

Observations:

- The size of the sample is independent of the support threshold minsup and of the number *N* of transactions. It only depends on the apporximation guarantee embodied in the parameters ε, δ, and on the max transaction length *h*, which is often quite low.
- There are bounds on the VC-dimension tighter than *h*
- The sample-based algorithm yields a 2-round MapReduce algorithm:

  in first round the sample of suitable size is extracted;

  in the second round the approximate set of frequent itemsets is extracted from the sample within one reducer

Limitations of the Support/Confidence framework

1. **Redundancy**: many of the returned patterns may refer to the same piece of information
2. **Difficult control of output size**: it is hard to predict how many patterns will be returned for given support/confidence threshold
3. **Significance**: are the returned patterns significant/interesting?

Closed Itemsets

GOAL: Devise a lossless succint representation of the frequent itemsets.

Consider a dataset *T* of *N* transactions ove the set of itemsets *I*, and a support threshold *minsup*.

Definition (Closed Itemset)

An itemset $X \subseteq I$ is *closed* w.r.t. T if for each superset $Y \supset X$ we have $\text{Supp}(Y) < \text{Supp}(X)$.

Notation

- $\text{CLO}_T = \{X \subseteq I \ : \ X \text{ is closed w.r.t } T\}$
- $\text{CLO-F}_{T,\text{minsup}} = \{X \in \text{CLO}_T \ : \ \text{Supp}(X) \geq \text{minsup}\}$

Observation: the empty itemset, whose support is 1, is closed if and only if it is the only itemset of support 1

Maximal Itemsets

Definition (Maximal Itemset)

An itemset $X \subseteq I$ is *maximal* w.r.t. T and minsup if $\text{Supp}(X) \geq \text{minsup}$ and for each superset $Y \supset X$ we have $\text{Supp}(Y) < \text{minsup}$.

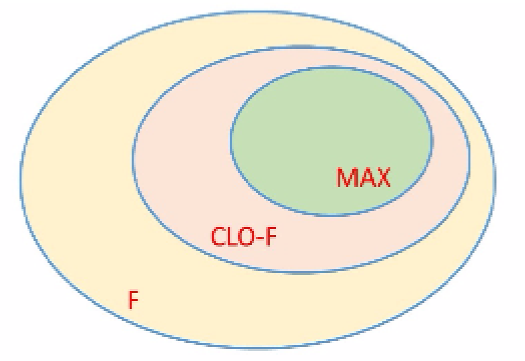
Notation

- $\text{MAX}_{T,\text{minsup}} = \{X \subseteq I \ : \ X \text{ is maximal w.r.t. } T\}$

Closed/maximal itemsets

The following properties can be easily shown (exercise):

- For each itemset $X \subseteq I$ there exists $X' \in \text{CLO}$ such that $X' \supseteq X$ and $\text{Supp}(X') = \text{Supp}(X)$
- For each frequent itemset $X \in \text{F}$ there exists $X' \in \text{MAX}$ such that $X' \supseteq X$
- $\text{MAX} \subseteq \text{CLO-F} \subseteq \text{F}$.



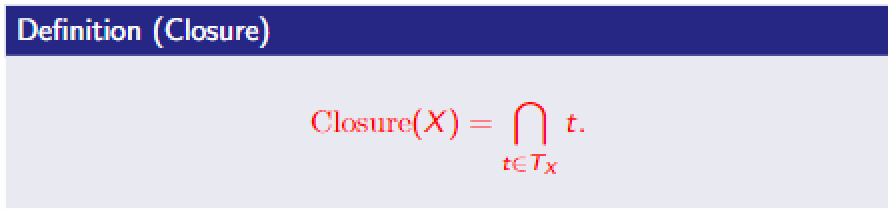
Observations:

- **MAX and CLO-F provide succint representations of F**: from the above properties we immediately conclude that the set of frequent itemsets *coincides* with the set of all subsets of the maximal itemsets, or, equivalently, with the set of all subsets of the frequent closed itemsets.

- **MAX provides a lossy representation of F**: in general, the support of a frequent itemset *cannot be derived* from the maximal itemsets and their supports.
- **CLO-F provides a lossless representation of F**: the support of any frequent itemset can be derived from the frequent closed itemsets and their supports.

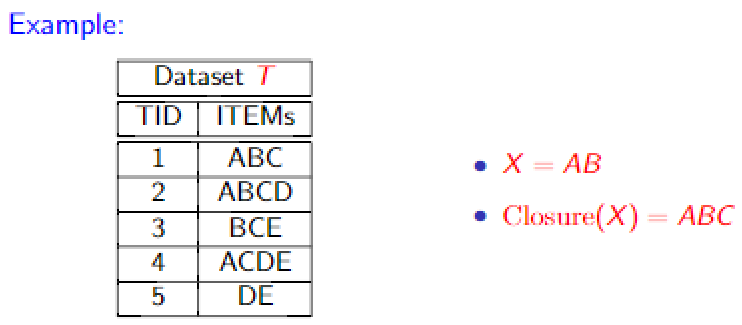
Representativity of closed itemsets

For *X subset of I*, let *T_x* denote the set of transactions where *X* occurs



Definition (Closure)

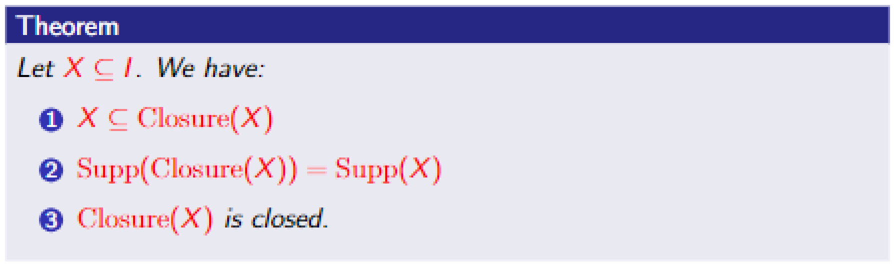
$$\text{Closure}(X) = \bigcap_{t \in T_X} t.$$



Example:

Dataset T	
TID	ITEMs
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

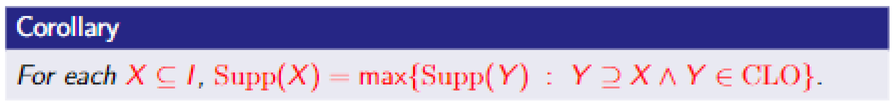
- $X = AB$
- $\text{Closure}(X) = ABC$



Theorem

Let $X \subseteq I$. *We have:*

1. $X \subseteq \text{Closure}(X)$
2. $\text{Supp}(\text{Closure}(X)) = \text{Supp}(X)$
3. $\text{Closure}(X)$ *is closed.*



Corollary

For each $X \subseteq I$, $\text{Supp}(X) = \max\{\text{Supp}(Y) \;:\; Y \supseteq X \wedge Y \in \text{CLO}\}$.

Observations:

- A consequence of the previous corollary is that from the frequent closed itemsets and their supports one can derive all frequent itemsets and their supports. In the sense, frequent closed itemsets and their supports provide a lossless representation of the frequent itemsets and their supports.
- Each (frequent) closed itemset *Y* can be regarded as a representative of all those itemsets *X* such that **Closure(X) = Y**
- There exist efficient algorithms for mining maximal or frequent closed itemsets
- Notions of closure similar to the ones used for itemsets are employed in other mining contexts

Exponentiality of maximal/frequent closed itemsets

Although maximal and frequent closed itemsets provide in practice succint representations of the frequent itemsets, still there are pathological instances where the number of maximal itemsets, hence the number of frequent closed itemsets is exponential in the input size. The following exercise provides an example

Top-*K* frequent (closed) itemsets

How about if we impose explicitly a limit on the output size?

Let *T* be a dataset of *N* transactions over a set *I* of *d* items. Let F_{T,s} and CLO-F_{T,s} denote, respectively, the sets of frequent itemsets and frequent closed itemsets with regard to threshold *s*. For *K* > 0 define

$$\begin{aligned} s(K) &= \max\{s : |\mathrm{F}_{T,s}| \geq K\} \\ sc(K) &= \max\{s : |\mathrm{CLO\text{-}F}_{T,s}| \geq K\} \end{aligned}$$

Then

- Top-K frequent itemsets w.r.t. $T = \mathrm{F}_{T,s(K)}$
- Top-K frequent closed itemsets w.r.t. $T = \mathrm{CLO\text{-}F}_{T,sc(K)}$

Observations:

- *K* is the target number of patterns, but the actual number of Top-*K* frequent (closed) itemsets could be larger than *K*
- How well does the parameter *K* controls the output size?
  - The next theorem shows that for Top-*K* frequent itemsets, *K* provides a somewhat tight control on the output size

Theorem

For $K > 0$, the Top-K frequent closed itemsets are $O(d \cdot K)$, where d is the number of items.

Significance

How do we measure the significance/interest of itemsets/rules?

- **Subjective measures**: user defined criteria based on domain knowledge.
- **Objective measures**: quantitative criteria, often based on statistics, such as support and confidence, for which the user fixes suitable thresholds

Are supoort/confidence adequate to capture significance? In general, the answer is "NO", but with some amendments their effectiveness can be improved

Beyond Confidence

Consider a dataset with 1000 transactions from a supermarket and let the following **contingency table**

	coffee	$\overline{\text{coffee}}$	
tea	150	50	200
$\overline{\text{tea}}$	650	150	800
	800	200	1000

The table should be read as follows:

- 150 transactions contain tea and coffee
- 50 transactions contain tea but not coffee

- 650 transactions contain coffee but no tea
- 150 transactions contain neither tea nor coffee
- Altogether, of the 1000 transactions, 200 contain tea, 800 do not contain tea, 800 contain coffee, and 200 do not contain coffee

Consider rule

r: tea->coffee

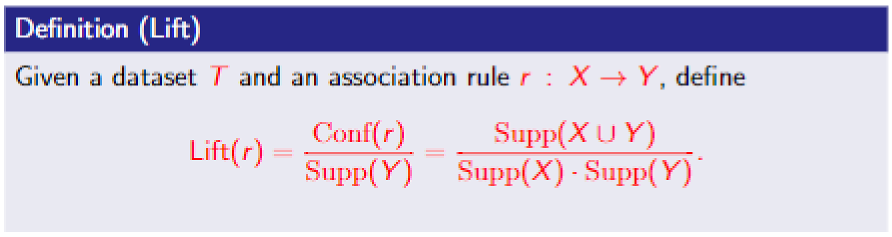
We have

- Supp(r) = 0.15 and Conf(r) = 0.75
- Supp(coffee) = 0.8

Observation: While Conf(r) seems relatively high, in fact a random customer is more likely to buy coffee than a customer who bought tea

Lift

The following measure is often used to assess the significance of high-confidence rules.



Definition (Lift)

Given a dataset T and an association rule $r : X \rightarrow Y$, define

$$\text{Lift}(r) = \frac{\text{Conf}(r)}{\text{Supp}(Y)} = \frac{\text{Supp}(X \cup Y)}{\text{Supp}(X) \cdot \text{Supp}(Y)}.$$

- Lift(r) is sometimes referred to as the *Interest Factor* of the pair of itemsets *X,Y*.
- The denominator represent the expected support of *X union Y* would *X* and *Y* occur independently in the transactions
- Lift(r) ~ 1 => *X* and *Y* are uncorrelated
- Lift(r) >> 1 => *X* and *Y* are positively correlated
- Lift(r) << 1 => *X* and *Y* are negatively correlated

Usually, association rules are mined with the traditional support-confidence framework and then they are sorted by decreasing lift. Those rules with high lift are selected as significant.

In the previous example, Lift(tea->coffee) = 0.9375, hence rule tea->coffee, altough it has high confidence, cannot be considered significant

Lift is symmetric with respect to the two sides of the rule. There exists asymmetric measures to assess the significance association rules.

Blackboard

given *T* and *I*

X, Y subset I closed

Z = X intersect Y

X and Y closed => Z closed

given X and X' subset X

Supp(X) > Supp(X')