Association Analysis

Part 1

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 occur together in a (surprisingly) high number of baskets
- Association rules: correlations between subsets of items.

Popular example of association rule: customers who buy diapers and milk are likely to also buy beer

Rigorous formulation of the problem

Dataset $T = \{t_1, t_2, \ldots, t_N\}$ of N transactions (i.e., baskets) over a set I of d items, with $t_i \subseteq I$, for $1 \leq i \leq N$.

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 $\mathrm{Supp}_T(X)$, is $|T_X|/N$, that is, the *fraction* of transactions of T that contain X. (Note that $\mathrm{Supp}_T(\emptyset) = 1$.)

Definition (Association rule and its support and confidence)

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

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

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

Rigorous formulation of the problem (cont'd)

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 (non empty) itemsets X such that $\text{Supp}_T(X) \geq \text{minsup}$. We denote this set by $F_{T,\text{minsup}}$.
2. Compute all (interesting) association rules r such that $\text{Supp}_T(r) \geq \text{minsup}$ and $\text{Conf}_T(r) \geq \text{minconf}$.

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 (itemset or rule). In particular, the thresholds *minsup* and *minconf* define which patterns must be regarded as interesting.
- (*Hypothesis testing setting*) Ideally, we would like that the support and confidence (for rules) 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 (possibly exponential in the input size) which become hard to exploit effectively.
  - False positive/negatives: low thresholds may yield a lot of uninteresting patterns (false positives), while high thresholds may miss some interesting patterns (false negatives).

Applications

Association analysis is one of the most prominent data mining tasks

1. Analysis of true market baskets. Chain stores keep TBs of data about what customers buy. Frequent itemsets and association rules can help the store lay out products so to "tempt" potential buyers, or decide marketing campaigns.
2. Detection of plagiarism. Consider documents (items) and sentences (transactions). For a given transaction t, its constituent items are those documents where sentence t occurs. A frequent pair of items represent two documents that share a lot of sentences ($\Rightarrow$ possible plagiarism)
3. Analysis of biomarkers. Consider transactions associated with patients, where each transaction contains, as items, biomarkers and diseases. Association rules can help associate a particular disease to specific biomarkers.

The mining of frequent itemsets can be generalized to search for: motifs in (biological) sequences or networks; sequential patterns; etc.

Techniques devloped for itemsets can be exploited also in these contexts.

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 even for ground sets I of small size (say $d > 40$)
- As a first approximation, we consider efficient strategies those that require time/space polynomial in both the input and the output sizes. (Polynomiality only w.r.t. input may not be feasible if output is large!)

Potential output explosion (cont'd)

Proof of theorem.

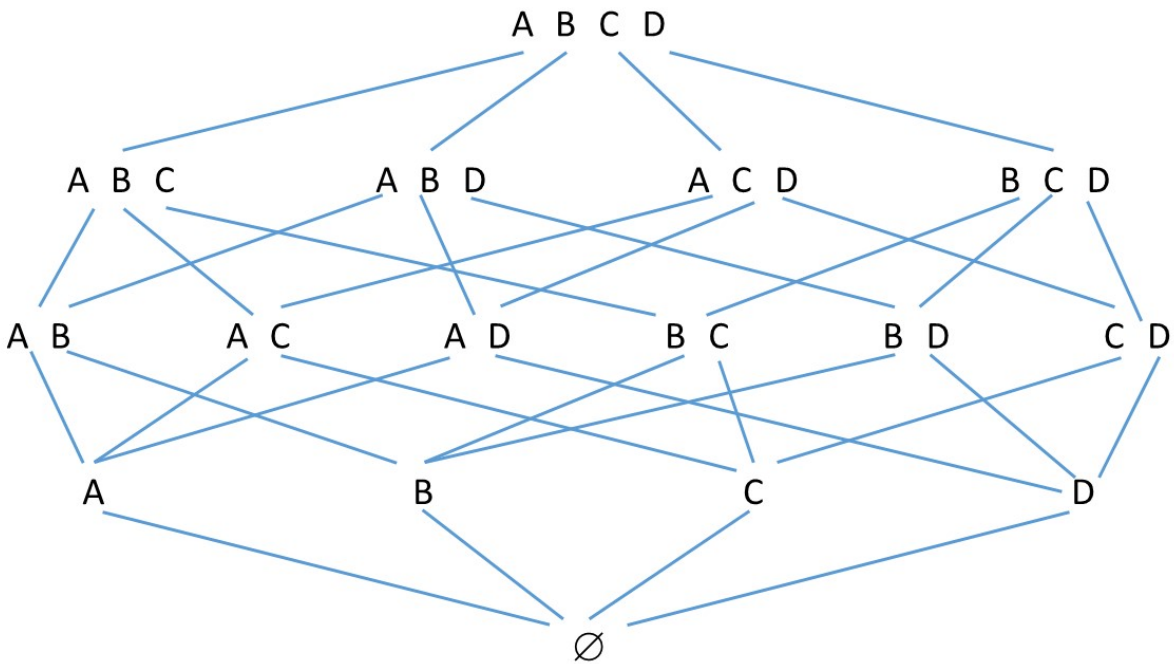
The count of itemsets is trivial. As for the association rules, we count separately those whose LHS has k items, for $1 \leq k < d$. There are $\binom{d}{k}$ possible itemsets of size k, and each of these, say X, can form a rule with $2^{d-k} - 1$ distinct non-empty itemsets, disjoint from X. Thus, the total number of rules is:

$$\begin{aligned}
\sum_{k=1}^{d-1} \binom{d}{k}(2^{d-k} - 1) &= \sum_{k=0}^{d-1} \binom{d}{k}(2^{d-k} - 1) - (2^d - 1) \\
&= \left(\sum_{k=0}^{d} \binom{d}{k} 2^{d-k}\right) - \left(\sum_{k=0}^{d} \binom{d}{k}\right) - (2^d - 1) \\
&= 3^d - 2^d - 2^d + 1 = 3^d - 2^{d+1} + 1
\end{aligned}$$

Use the fact $\sum_{k=0}^{d} \binom{d}{k} 2^{d-k} = \sum_{k=0}^{d} \binom{d}{k} 1^k 2^{d-k}$. □

Lattice of Itemsets

- The family of itemsets under $\subseteq$ forms a lattice, namely a partially ordered set where for each to elements X, Y there exists a unique least upper bound $(X \cup Y)$ and a unique greatest lower bound $(X \cap Y)$.
- The lattice can be represented through the Hasse diagram



Anti-monotonicity of Support

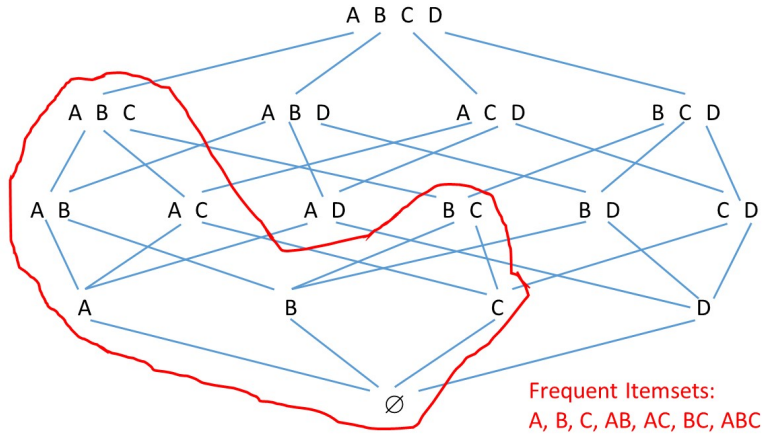
The support function for itemsets exhibits the following property, referred to as anti-monotonicity. For every $X, Y \subseteq I$

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

Immediate consequence. For a given support threshold, we have

1. X is frequent $\Rightarrow \forall W \subseteq X$, W is frequent (*downward closure*)
2. X is not frequent $\Rightarrow \forall W \supseteq X$, 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: (typical of most algorithms)

Phase 1: Compute the set F of all frequent itemsets w.r.t. minsup

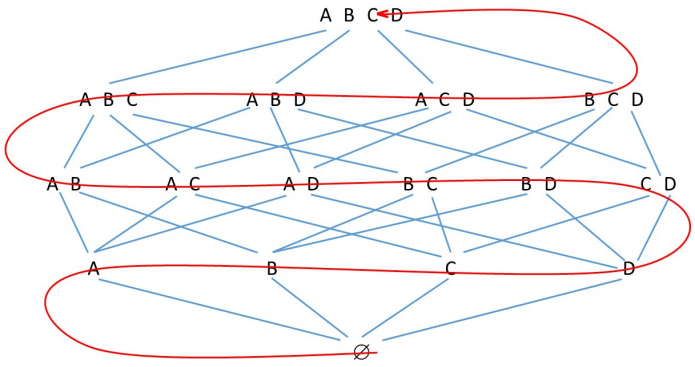
Phase 2: For each itemset $Z \in F$, compute all rules $r : X \to Y$, with $X \cup Y = Z$ and confidence at least minconf.

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

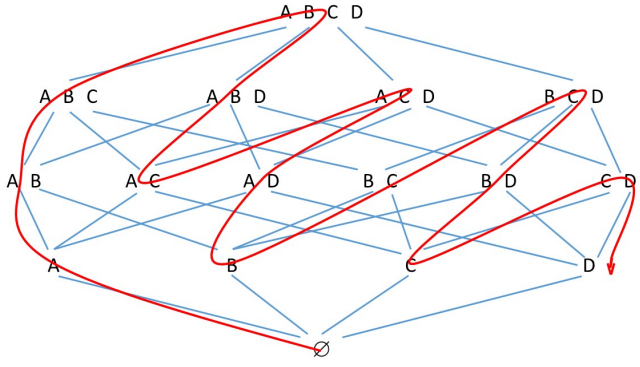
Efficient mining of F.I. and A.R. (cont'd)

Two main approaches

Breadth First



Depth First



F.I. mining: A-Priori algorithm

A-Priori is a popular, paradigmatic data mining algorithm devised by Agrawal and Srikant in 1994 at IBM Almaden (presented at 20th Very Large Data Base Conf., 1994).

Uses the breadth-first approach.

For every itemset $X \subseteq I$, define its absolute support (with respect to a dataset T of N transactions)

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

MAIN ALGORITHM

Input Dataset T of N transactions over I, minsup

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

F.I. mining: A-Priori algorithm (cont'd)

$k \leftarrow 1$

Compute $F_1 = \{i \in I \ ; \ \mathtt{Supp}(\{i\}) \geq \mathtt{minsup}\}$

Compute $O_1 = \{(X, \mathtt{Supp}(X)) \ ; \ X \in F_1\}$

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** $\sigma(c) \leftarrow \sigma(c) + 1$

  $F_k \leftarrow \{c \in C_k \ : \ \sigma(c) \geq N \cdot \mathtt{minsup}\}$;

  $O_k \leftarrow \{(X, \sigma(X)/N) \ ; \ X \in F_k\}$

until $F_k = \emptyset$

return $\bigcup_{k \geq 1} O_k$

F.I. mining: A-Priori algorithm (cont'd)

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 Φ
/* *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 Φ; exit inner loop}
return Φ

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.

Example

DATASET T	
TID	ITEMS
1	ACD
2	BEF
3	ABCEF
4	ABF

Set F_1	
ITEMSET	SUPPORT
A	$3/4$
B	$3/4$
C	$1/2$
E	$1/2$
F	$3/4$

Example (cont'd)

Set C_2			Set F_2
ITEMSET		SUPPORT	
After Generation	After Pruning		
AB	AB	1/2	yes
AC	AC	1/2	yes
AE	AE	1/4	no
AF	AF	1/2	yes
BC	BC	1/4	no
BE	BE	1/2	yes
BF	BF	3/4	yes
CE	CE	1/4	no
CF	CF	1/4	no
EF	EF	1/2	yes

Example (cont'd)

Set C_3			Set F_3
ITEMSET		SUPPORT	
After Generation	After Pruning		
ABC			no
ABF	ABF	1/2	yes
ACF			no
BEF	BEF	1/2	yes

Correctness of A-Priori

We assume the existence of a total ordering of the items, and assume that transactions/itemsets are represented as sorted vectors.

Theorem (Correctness)

The A-Priori algorithm for mining frequent itemsets is correct

Proof

By induction on $k \geq 1$, we show that the set F_k computed by the algorithm consists of all frequent itemsets of size k.

- Basis $k = 1$: trivial
- Induction step. Fix $k > 1$ and assume (inductive hypothesis) the property holds up to index $k - 1$. It is sufficient to prove that for an arbitrary frequent itemset X of size k, X is surely included in the set C_k returned by $\text{APRIORI-GEN}(F_{k-1})$.

Correctness of A-Priori (cont'd)

Proof (cont'd).

Let $X = X[1]X[2]\cdots X[k]$ and define

$$\begin{aligned} X(1) &= X[1]X[2]\cdots X[k-2]X[k-1] \\ X(2) &= X[1]X[2]\cdots X[k-2]X[k]. \end{aligned}$$

Clearly, $X = X(1) \cup X(2)$. Also, both $X(1)$ and $X(2)$ have length $k-1$ and are frequent, by anti-monotonicity of support. Thus, $X(1), X(2) \in F_{k-1}$, hence X is added to the pool of candidates in the generation phase and cannot be eliminated in the pruning phase, since, being frequent, all of its subsets are also frequent.

□

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 (typically very large) 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 (see later slides)

Efficiency of A-Priori (cont'd)

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).

Efficiency of A-Priori (cont'd)

Proof of lemma.

Let $M = \sum_{k \geq 1} m_k$, where m_k is the number of frequent itemsets of length k. Consider the invocation $\text{APRIORI-GEN}(F_{k-1})$, for some $k > 1$. By construction, $\text{APRIORI-GEN}(F_{k-1})$ generates at most m_{k-1}^2 candidates.

Moreover, each generated itemset $X = X[1] \cdots X[k-1]X[k]$ can be seen as $X' = X[0 \div k-1] \cup \{X[k]\}$, with $X' \in F_{k-1}$ and $X[k] \in I$. This implies that for each $X' \in F_{k-1}$ less than d candidates $X' \cup \{a\}$, with $a \in I$, are generated. Therefore, the total number of itemsets generated over all invocations of APRIORI-GEN is at most

$$\sum_{k>1} \min\{m_{k-1}^2, dm_{k-1}\} \leq \min\{M^2, dM\}$$

□

Optimizations of A-Priori: frequent pairs

- The support counting for the candidates in C_k, for $k \geq 2$, is typically the most time-consuming step because: (a) requires a pass over the entire dataset; (b) may use much space (number of candidates can be quadratic in the actual number of frequent itemsets)
- In practice, the issue of space (point (b) above) 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.
- Park, Chen and Yu [SIGMOD'95] devised a strategy to filter out some candidates from C_2 based on statistics gathered while computing F_1. This strategy is outlined in the next slide.

Optimizations of A-Priori: frequent pairs (cont'd)

GOAL: Compute F_1 efficiently and, at the same time, gather statistics for filtering out infrequent pairs. Consider an instance with 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 $0 \leq j < K$. Counters are initialized to 0.
- For every transaction t do
  - For each each item $i \in t$, increment γ_i
  - For each pair of items $i_1, i_2 \in t$, increment $\delta_{h(i_1,i_2)}$.
- Key remark: only pairs of items i_1, i_2 such that $\delta_{h(i_1,i_2)} \geq N \cdot \text{minsup}$ have a chance to be frequent.

Optimizations of A-Priori: frequent pairs (cont'd)

- Compute $F_1 = \{i \in I \ : \ \gamma_i \geq N \cdot \text{minsup}\}$
- 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 (based on the available memory), hence many pair counters are used, then filtering out of infrequent pairs become quite effective.

Other optimizations

A large body of literature has investigated several additional strategies to optimize the mining of frequent itemsets. E.g.:

- A data structure trie-like (Hash tree) 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 (smaller than C_k) to be checked for inclusion.
- Several implementations of depth-first mining strategies have been devised and tested (one of the fastest ones to date, *patriciamine*, come from Padova!). Their goal is to avoid several passes over the entire dataset of transactions, which may be huge, and to confine the support counting 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 (set $\bigcup_{k \geq 1} O_k$ returned by A-Priori) all association rules which are based on 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 constraint 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 .

Mining association rules (cont'd)

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

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

- Immediate consequence.

$$\text{Conf}(Z - Y' \rightarrow Y') < \text{minconf} \Rightarrow \text{Conf}(Z - Y \rightarrow Y) < \text{minconf}.$$

Thus, for each frequent 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

ASSOCIATION RULE ALGORITHM

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

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

Algorithm for mining association rules (cont'd)

AP-GENRULES(Z)

$m \leftarrow 1$
$H_{Z,1} \leftarrow \{Y \subset Z \ : \ |Y| = 1 \wedge \texttt{Supp}(Z)/\texttt{Supp}(Z - Y) \geq \texttt{minconf}\}$
$R_{Z,1} \leftarrow \{(r, \texttt{Supp}(r), \texttt{Conf}(r)) \ : \ r : Z - Y \rightarrow Y, \ \texttt{with} \ Y \in H_{Z,1}\}$
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 $(\texttt{Supp}(Z)/\texttt{Supp}(Z - Y) \geq \texttt{minconf})$
      then `add` $(r : Z - Y \rightarrow Y, \texttt{Supp}(r), \texttt{Conf}(r))$ `to` $R_{Z,m}$
      else `remove` Y `from` $H_{Z,m}$
until $H_{Z,m} = \emptyset$
return $\bigcup_{m \geq 1} R_{Z,m}$

Example

Set O	
ITEMSET	SUPPORT
A	1/2
B	3/4
C	3/4
E	3/4
AC	1/2
BC	1/2
BE	3/4
CE	1/2
BCE	1/2

Consider AP-GENRULES(Z), with $Z = BCE$ and minconf $= 3/4$. We have

$$\begin{aligned} H_{Z,1} &= \{E, B\} \\ R_{Z,1} &= \{(BC \to E, 1/2, 1), \\ &= (CE \to B, 1/2, 1)\} \end{aligned}$$

Note that $C \notin H_{Z,1}$ since $\text{Conf}(BE \to C) = 2/3 <$ minconf.

In the first iteration of the repeat, the algorithm computes $H_{Z,2} = \{BE\}$ thorugh APRIORI-GEN, but then it removes BE from $H_{Z,2}$ since $\text{Conf}(C \to BE) = 2/3 <$ minconf. Thus, both $H_{Z,2}$ and $R_{Z,2}$ are empty at the end of the iteration, and the rules $BC \to E$ and $CE \to B$ are returned (with their supports and confidences).

Correctness of the A.R. algorithm

Theorem (Correctness)

The algorithm for mining association rules is correct

Proof

Let Z be a frequent itemset of size $k > 1$. By induction on $m \geq 1$ we show that the set $H_{Z,m}$ computed in each iteration of AP-GENRULES (first initialized by APRIORI-GEN$(H_{Z,m-1})$ and then suitably pruned in the for-each loop) coincides with the set of consequents $Y \subset Z$ of size m such that $\text{Conf}(Z - Y \rightarrow Y) \geq$ minconf.

Correctness of the A.R. algorithm (cont'd)

Proof (cont'd).

- Basis $m = 1$: trivial
- Induction step. Let us fix $m > 1$ and assume (inductive hypothesis) that the property holds up to index $m - 1$. In particular, we assume inductively, that $H_{Z,m-1}$ coincides with the set of consequents $Y \subset Z$ of size $m - 1$ such that the rule $r : Z - Y \to Y$ has confidence at least minconf. It is sufficient to prove that given an arbitrary subset $Y \subset Z$ of size m such that the rule $r : Z - Y \to Y$ has confidence at least `minconf`, Y is included in the output of APRIORI-GEN$(H_{Z,m-1})$. The argument is similar to the one used to prove correctness of the frequent itemsets mining algorithm and is left as an exercise.

□

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 would 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 excessively large number of rules be checked.
- It can be easily shown that the algorithm the algorithm can be implemented in time polynomial in both the input size (sum of all frequent itemsets lengths) and output size (sum of the lengths of all returned association rules).

Frequent itemset 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

The following 4-round MapReduce algorithm is based on the SON's algorithm [VLDB'95].

Let T be a set of N transactions over a set I of items, and let minsup be the support threshold. Assume that transactions are represented as pairs (i, t_i), with $0 \leq i < N$, where i is the Transaction ID (TID) and $t_i \subseteq I$ is the transaction.

Fix a suitable design parameter K, with $1 < K < N$.

Partition-based approach (cont'd)

- Round 1: Partition T arbitrarily into K subsets $T_0, T_1, \ldots T_{K-1}$ of $O(N/K)$ transactions each, and compute the set of frequent itemsets w.r.t. minsup independently in each subset. (Note that the same itemset can be extracted from multiple subsets.)
- Round 2: Gather all frequent itemsets computed in Round 1 and eliminate the duplicates. Call Φ the resulting set of itemsets.
- Round 3: For every $0 \leq j < K$ independently do the following: gather a copy of Φ and T_j and compute, for each $X \in \Phi$, the number of transactions of T_j that contain X (call this number $\sigma(X, j)$.
- Round 4: For each $X \in \Phi$, gather all $\sigma(X, j)$'s, compute the final support $\text{Supp}(X) = (1/N) \sum_{j=0}^{K-1} \sigma(X, j)$ and output X if $\text{Supp}(X) \geq \text{minsup}$.

Partition-based approach: analysis

- Correctness: it follows from the fact that *each itemset frequent in T must be frequent in some T_j*, which you can prove as a simple exercise. 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 (1) the size of Φ, which cannot be easily predicted, and (2) 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

Do we really need to process the entire dataset?

No, if we are happy with some

approximate set of frequent itemsets

(but quality of approximation under control)

What follows is based on [RU14]

Sampling-based approach (cont'd)

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.

Sampling-based approach (cont'd)

Observations

- Condition (1) ensures that the approximate set C comprises all true frequent itemsets
- Condition (2) ensures that: (a) C does not contain itemsets 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.

Sampling-based approach (cont'd)

Simple algorithm

Let T be a dataset of N transactions over I, and $\text{minsup} \in (0, 1]$ a support threshold. Let also $\theta(\text{minsup}) < \text{minsup}$ be a suitably lower support threshold

- Let $S \subseteq T$ 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 $F_{S,\theta(\text{minsup})}$ is the set of frequent itemsets w.r.t. S and $\theta(\text{minsup})$.

How well does C approximate the true frequent itemsets and their supports?

Sampling-based approach (cont'd)

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

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

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.

The proof of the theorem requires the notion of VC-dimension.

VC-dimension [Vapnik,Chervonenkis 1971]

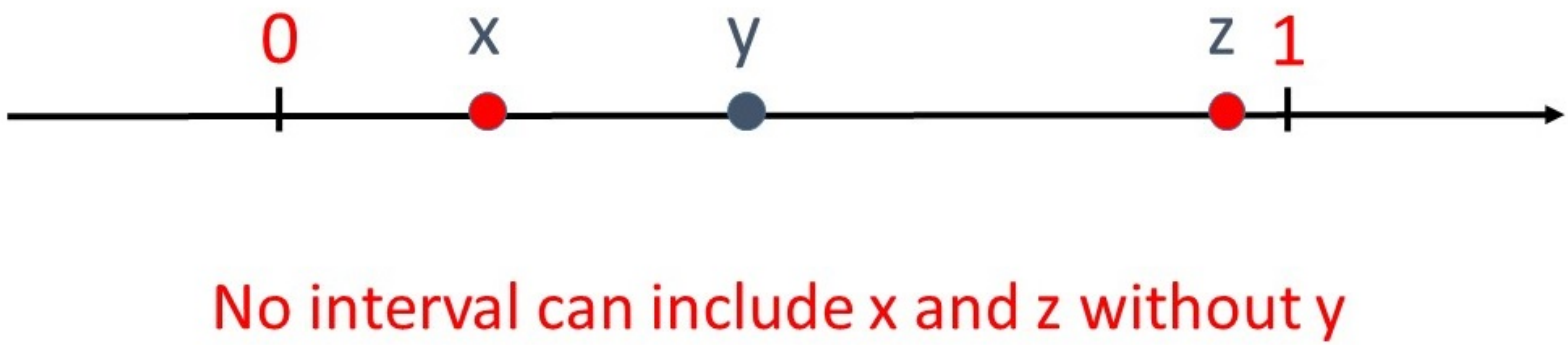
Powerful notion in statistics and learning theory

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.

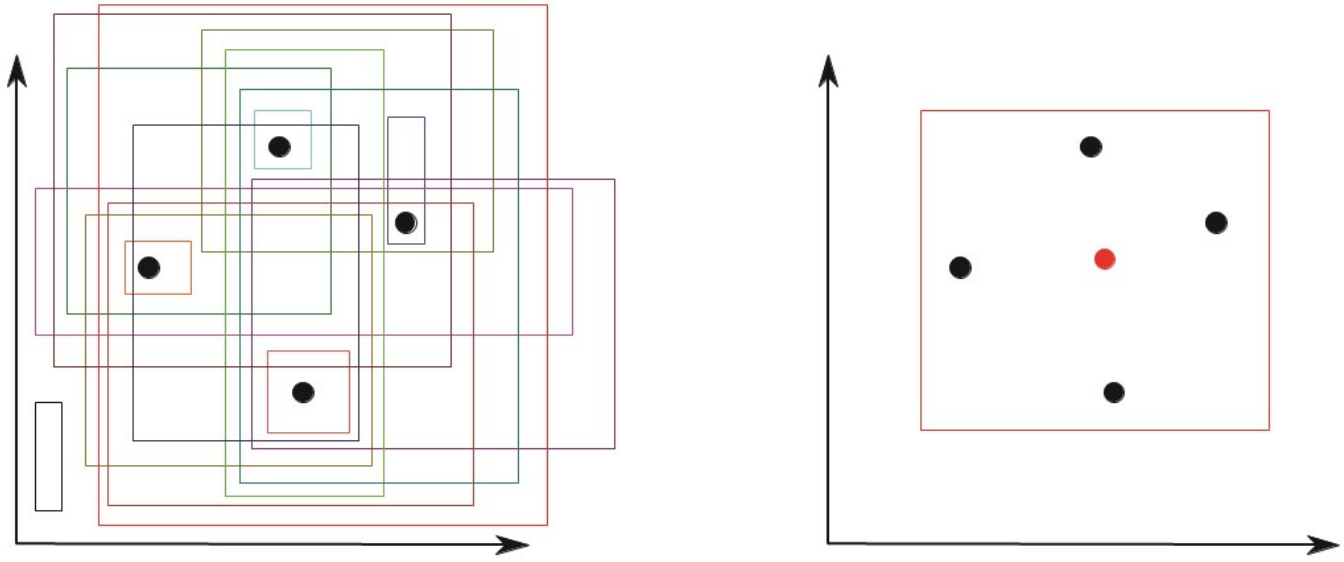
VC-dimension: examples

Let $D = [0, 1]$ and let R be the family of intervals $[a, b] \subseteq [0, 1]$. It is easy to see that the VC-dimension of (D, R) is ≥ 2. Consider any 3 points $0 \leq x < y < z \leq 1$. The following picture show that it is < 3, hence it must be equal to 2.



VC-dimension: examples (cont'd)

Let $D = \Re^2$ and let R be the family of axis-aligned rectangles.



There exist 4 points shattered by R (left), but no set of 5 points can be shattered by R (right)

$\Rightarrow$ the VC-dimension of (D, R) is 4.

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 $r \in R$

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

We will not prove the lemma. See [RU14] for pointers to proof.

Analysis of sampling-based approach (cont'd)

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

- $D = T$
- $R = \{T_X : X \subseteq I \wedge X \neq \emptyset\}$, where T_X is the set of transactions that contain X.

It can be shown that the VC-dimension of (D, R) is $\leq h$, where h is the maximum transaction length.

Analysis of sampling-based approach (cont'd)

Proof of Theorem (Riondato-Upfal).

Regard T as a range space (D, R) of VC-dimension h, as explained before. The Sampling Lemma with $\epsilon_1 = \epsilon/2$ shows that with probability $\geq 1 - \delta$ for each itemset X it holds $|\text{Supp}_T(X) - \text{Supp}_S(X)| \leq \epsilon/2$. Assume that this is the case. Therefore:

- For each frequent itemset $X \in F_{T,\text{minsup}}$

  $$\text{Supp}_S(X) \geq \text{Supp}_T(X) - \epsilon/2 \geq \text{minsup} - \epsilon/2 = \theta(\text{minsup}),$$

  hence, the pair $(X, s_X = \text{Supp}_S(X))$ is returned by the algorithm;
- For each pair $(X, s_X = \text{Supp}_S(X))$ returned by the algorithm
  - $\text{Supp}_T(X) \geq s_X - \epsilon/2 \geq \theta(\text{minsup}) - \epsilon/2 \geq \text{minsup} - \epsilon$
  - $|\text{Supp}_T(X) - s_X| \leq \epsilon/2 < \epsilon$.

Hence, the output of the algorithm is an ϵ-approximation of the true frequent itemsets and their supports. □

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 approximation 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 (e.g., through A-Priori) within one reducer.

Exercises

Exercise 1

Argue rigorously that given a family F of itemsets of the same length, represented as sorted arrays of items, function APRIORI-GEN(F) does not generate the same itemset twice.

Exercise 2

Consider two association rules $r_1 : A \to B$, and $r_2 : B \to C$, and suppose that both satisfy support and confidence requirements. Is it true that also $r_3 : A \to C$ satisfies the requirements? If so, prove it, otherwise show a counterexample.

Exercises

Exercise 3

Let

$$\begin{aligned} c_1 &= \text{Conf}(A \to B) \\ c_2 &= \text{Conf}(A \to BC) \\ c_3 &= \text{Conf}(AC \to B) \end{aligned}$$

What relationships do exist among the $c_i's$?

Exercise 4

For a given itemset $X = \{x_1, x_2, \ldots, x_k\}$, define the measure:

$$\zeta(X) = \min\{\text{Conf}(x_i \to X - \{x_i\}) \ : \ 1 \leq i \leq k\}.$$

Say whether ζ is *monotone*, *anti-monotone* or neither one. Justify your answer.

Exercises

Exercise 5

Consider the following alternative implementation of procedure APRIORI-GEN(F_{k-1}) (regard an itemset $X \in F_{k-1}$ as an array of items $X[1], X[2], \ldots, X[k-1]$ in increasing order):

$C_k \leftarrow \emptyset$;
for each $X \in F_{k-1}$ **do**
  for each $(i \in F_1)$ **do**
    if $(i > X[k-1])$ **then** `add` $X \cup \{i\}$ `to` C_k
`remove from` C_k `every itemset containing at least one subset of length` $k-1$ `not in` F_{k-1}
return C_k

Show that the set C_k returned by the above procedure contains all frequent itemsets of length k.

References

LRU14 J.Leskovec, A.Rajaraman and J.Ullman. Mining Massive Datasets. Cambridge University Press, 2014. Chapter 6.

TSK06 P.N.Tan, M.Steinbach, V.Kumar. Introduction to Data Mining. Addison Wesley, 2006. Chapter 6.

RU14 M. Riondato, E. Upfal: Efficient Discovery of Association Rules and Frequent Itemsets through Sampling with Tight Performance Guarantees. ACM Trans. on Knowledge Discovery from Data, 2014.