ID2222: Data Mining

KTH ROYAL INSTITUTE OF TECHNOLOGY

Frequent Itemsets

Vladimir Vlassov

Acknowledgement: Most of the slides are adopted from the slide provided at http://www.mmds.org for the book "Mining of Massive Datasets" by Jure Leskovec, Anand Rajaraman, Jeff Ullman, Stanford University

Outline

- Introduction
- The ”Market-basket” model of the data
- Define:
  - Frequent itemsets: support
  - Association rules: confidence, interestingness
- Algorithms for finding frequent itemsets
  - Finding frequent pairs
  - A-Priori algorithm
  - (Self-study) PCY algorithm + 2 refinements

The Discovery of Frequent Itemsets

Goal: Identify ***items*** that frequently appear together in sets of items (***baskets***) – the ***frequent itemsets***

- E.g., identify products (itemsets) bought together by sufficiently many customers
- Usually, the number of items in a basket is much smaller than the total number of items
- Usually, the number of baskets is very large
  - cannot fit in the main memory

Association Rule Discovery

The problem of frequent itemsets is often viewed as ***the discovery of "association rules."***

- "if-then" rules in the form of implication ***X*** → ***Y*** (if ***X*** then ***Y)***
- where ***X*** and ***Y*** are itemsets (***Y*** can be just an item)

For example: If ***butter*** and ***bread*** are bought together, customers also buy ***milk***

{butter, bread} → ***{milk}***

- The rule needs the support of many (several hundred) transactions to be statistically significant.

We need to count the number of occurrences of a given itemset in all transactions (baskets)

Mining Association Rules

Rakesh Agrawal, Tomasz Imieliński, Arun Swami, ”*Mining association rules between sets of items in large databases,” SIGMOD ‘93*

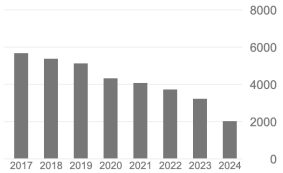
Rakesh Agrawal FOLLOW

Data Insights Laboratories
Verified email at datainsightslabs.com
Data Mining Privacy Web Search Education

TITLE	CITED BY	YEAR
Fast algorithms for mining association rules R Agrawal, R Srikant Proc. 20th int. conf. very large data bases, VLDB 1215, 487-499	31644	1994
Association rules between sets of items in large databases R Agrawal, T Imielinski, AN Swami Proc. of ACM SIGMOD Int. Conf. on Management of Data, Washington, 207-216	26633	1993

Cited by VIEW ALL

	All	Since 2019
Citations	143693	22495
h-index	108	43
i10-index	269	132



source: https://scholar.google.se/citations?user=4XSG7v4AAAAJ&hl=en

Association Rule Discovery

Supermarket shelf management:

Goal: Identify items that are bought together by sufficiently many customers

Approach: Process the sales data collected with barcode scanners to find dependencies among items

A classic rule:

- If a customer buys diapers and milk, then (s)he is likely to buy beer
- Don't be surprised if you find six-packs next to diapers!

The Market-Basket Model

A large set of ***items*** $I = \{i_1, i_2, \ldots, i_m\}$
- e.g., things sold in a supermarket

A large set of ***baskets*** (transactions)
Each basket is a small subset of items $t \subset I$
- e.g., the things a customer bought in one trip to a shop

Transaction/basket database $T = \{t_1, t_2, \ldots, t_n\}$.

Want to discover association rules:
- People who bought $\{x, y, z\}$ tend to buy $\{v, w\}$
- Amazon! (as well as Netflix, Spotify, …)

The Market-Basket Model

Input:

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Output:

Rules Discovered:
{ Milk } → { Coke }
{ Diaper, Milk } → { Beer }

Support of { Diaper, Milk } is 3/5; support of { Diaper, Milk, Beer } is 2/5; the confidence of { Diaper, Milk } → { Beer } is 2/3 – relatively high

Applications – Market Baskets

Items = products; **Baskets** = sets of products someone bought in one trip to the store

Real market baskets: Chain stores keep TBs of data about what customers buy together

- Tells how typical customers navigate stores, lets them position tempting items
- Suggests tie-in "tricks", e.g.
  - run a sale on diapers and raise the price of beer;
  - run a sale on hotdogs and raise the price of mustard
- Need the rule to occur frequently, or no $$'s

Amazon's customers who bought *X* also bought *Y*

Applications – Plagiarism

Baskets = sentences; **Items** = documents containing those sentences

- Items that appear together too often could represent plagiarism
- Notice items do not have to be "*in*" baskets

Applications – Drugs with Side-Effects

Baskets = patients; **Items** = drugs & side-effects

- Has been used to detect combinations of drugs that result in particular side-effects
- **But requires extension:** The absence of an item needs to be observed as well as the presence

Applications – Related Concepts

Baskets = Web pages; **items** = words.

- (Un)usual words appearing together in many documents, e.g., “Ryan” and “Scarlett”, or “Beyoncé” and “Jay-Z”, may indicate a relationship.

Association Rule Discovery: Approach

Given a set of baskets, discover association rules

- People who bought {a, b, c} tend to buy {d, e}

2-step approach

1. Find frequent itemsets
2. Generate/discover association rules

Definitions

An ***itemset X*** is a subset of items ***I: X ⊂ I***

E.g., ***X* = {milk, bread, cereal}** is an itemset.

A ***k-itemset*** is an itemset with ***k*** items.

E.g., **{milk, bread, cereal}** is a 3-itemset

An ***association rule*** is an implication $X \rightarrow Y$,
where $X, Y \subset I$, and $X \cap Y = \emptyset$

Frequent Itemsets. Support

Simplest question: Find sets of items that appear together "frequently" in baskets

Support for itemset ***I*** is the number of baskets containing all items in ***I***

- Often expressed as a fraction of the total number of baskets

Given a **support threshold** ***s***, sets of items that appear in at least ***s*** baskets are called ***frequent itemsets***

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Support of {Beer, Bread} = 2

Example: Frequent Itemsets

Items = {milk, coke, pepsi, beer, juice}

Support threshold = 3 baskets

BID	Baskets
B1	{m, c, b}
B2	{m, p, j}
B3	{m, b}
B4	{c, j}
B5	{m, p, b}
B6	{m, c, b, j}
B7	{c, b, j}
B8	{b, c}

Frequent itemsets (at least in 3 baskets):

singletons	{ m }, { c }, { b }, { j } 5 5 6 4
doubletons	{m, b}, {c, b}, {c, j} 4 4 3
tripletons	{m, c, b}, {c, b, j} 2 – not frequent

Important Observations: Monotonicity of Support

- The support of the itemset $J \subset I$ is at least as big as the support of the itemset I;
- For the itemset I to be frequent, all subsets of I must be frequent

In other words:

- ***The support of an itemset is at least as the support of its superset;***
- ***For an itemset to be frequent, all its subsets must be frequent***

Association Rules. Confidence

If-then rules about the contents of baskets: ***{i_1, i_2, … , i_k} → j***
"if a basket contains all of ***i_1, …,i_k,*** then it is ***likely*** to contain ***j.***"

Confidence of rule ***I → j*** is the ratio of the support for ***I ∪ {j}*** to the support for ***I***.

$$\text{conf}(I \rightarrow j) = \frac{\text{support}(I \cup j)}{\text{support}(I)}$$

The confidence of rule ***I → j*** is the fraction of the baskets with all of ***I*** containing ***j***.

Confidence of this association rule is the probability of ***j*** given $\boldsymbol{I = \{i_1,\ldots,i_k\}}$

Example: Confidence

Let's take a frequent doubleton **{m, b}**

- Support of **{m, b}** is 4

The confidence of **{m, b}** → **c** is

conf({m, b} → c) = 2/4 = 0,5 (50%)

- The itemset **{m, b}** appears in 4 baskets, B1, B3, B5, and B6.
- Of these, **c** appears in B1 and B6, or 2/4 of the baskets.

BID	Baskets
B1	{**m**, **c**, **b**}
B2	{m, p, j}
B3	{**m**, **b**}
B4	{c, j}
B5	{**m**, p, **b**}
B6	{**m**, **c**, **b**, j}
B7	{c, b, j}
B8	{b, c}

Interesting Association Rules

In practice, there are many rules; we want to find significant/interesting ones!

- The rule $X \rightarrow$ ***milk*** may have high confidence for many itemsets ***X*** because milk is just purchased very often (independent of ***X***), and the confidence will be high.

Interest of an association rule $I \rightarrow j$ is the difference between its confidence and the fraction of baskets that contain ***j***

$$\text{Interest}(I \rightarrow j) = \text{conf}(I \rightarrow j) - \Pr[j]$$

- ***Interesting rules are those with high positive or negative interest values (usually above 0.5)***

If I does not influence j, then we would expect that the fraction of baskets including I that contain j would be the same as the fraction of all baskets that contain j. Such a rule has interest 0. However, it is interesting, in both the informal and technical sense, if a rule has either high interest, meaning that the presence of I in a basket somehow causes the presence of j, or highly negative interest, meaning that the presence of I discourages the presence of j. ***Recall: Confidence*** of this association rule is the probability of ***j*** given $\boldsymbol{I} = \{\boldsymbol{i}_1,\ldots,\boldsymbol{i}_k\}$.

Example: Confidence and Interest

Association rule: {m, b} →c

- **Confidence =** 2/4 = 0.5
- **Interest =** (0.5 – 5/8) = (4/8 – 5/8) = -1/8 ≈ 0
  - *Rule is not very interesting!*
  - Item ***c*** appears in 5/8 of the baskets
- The probability of **c** given **{m, b}** (that is the confidence) is 50%, but **c** appears in baskets rather often, about 60% (with almost the same probability)
  - thus **{m, b}** has no (little) influence on **c**

BID	Baskets
B1	{**m**, **c**, **b**}
B2	{m, p, j}
B3	{**m**, **b**}
B4	{c, j}
B5	{**m**, p, **b**}
B6	{**m**, **c**, **b**, j}
B7	{c, b, j}
B8	{b, c}

if I has no influence on j, then we would expect that the fraction of baskets including I that contain j would be exactly the same as the fraction of all baskets that contain j. Such a rule has interest 0. However, it is interesting, in both the informal and technical sense, if a rule has either high interest, meaning that the presence of I in a basket somehow causes the presence of j, or highly negative interest, meaning that the presence of I discourages the presence of j.

Example: Confidence and Interest

{diapers} → {beer}

interest({diapers} → {beer}) =
(support({diapers,beer})/support({diapers})) –(support({beer})/num_baskets)

The fraction of diaper buyers that buy beer is significantly greater than the fraction of all customers that buy beer. *The rule is of high interest.*

{coke} → Pepsi

interest({coke} → Pepsi) = (support({coke, Pepsi})/support({coke})) –
(support({Pepsi})/num_baskets)

Negative interest – people who buy coke are unlikely also to buy Pepsi

Finding Association Rules

Goal: Find all association rules with support ≥*s* (at least *s*) and confidence ≥*c* (at least *c*)

- **Note:** Support of an association rule is the support of the set of items on the left side

The hard part: Finding the frequent itemsets

- If $I \rightarrow j$ has reasonably high support of I and confidence, then both I and $I \cup \{j\}$ will be "frequent."

$$\text{conf}(I \rightarrow j) = \frac{\text{support}(I \cup j)}{\text{support}(I)}$$

Finding Association Rules (cont)

We are looking for rules $I \rightarrow j$ with reasonably high support and confidence

- both I and $I \cup \{j\}$ will have fairly high support, i.e., should be "frequent."

For brick-and-mortar marketing:

- support of 1% is reasonably high
- confidence of 50% is adequate; otherwise, the rule has a little practical effect

Challenges

- Completeness: find all rules.
- Mining with data on a hard disk (not in memory)

Mining Association Rules

Step 1: Find all frequent itemsets ***I*** with at least as a given support ***s***

Step 2: **Rule generation**

- For every subset ***A*** of ***I***, generate a rule ***A → I \ A***
  - Since ***I*** is frequent, then so is ***A***
  - **Variant 1:** Single pass to compute the rule confidence
    - conf(***A,B→C,D***) = supp(**A,B,C,D**)/supp(**A,B**)
  - **Variant 2:**
    - **Observation:** If **A,B,C→D** is below confidence, so is **A,B→C,D** because of supp(**A,B**) ≥ supp(**A,B,C**) - in denominator of conf.
    - Can generate bigger rules from smaller ones
- **Output the rules above the confidence threshold**

conf(I→j) = supp(I,j)/supp(I)

Example

For every subset ***A*** of ***I***, generate a rule $A \rightarrow I \setminus A$
conf($A \rightarrow I \setminus A$) = supp($I$) / supp($A$)

BID	Baskets
B1	{m, c, b}
B2	{m, p, j}
B3	{m, c, b, n}
B4	{c, j}
B5	{m, p, b}
B6	{m, c, b, j}
B7	{c, b, j}
B8	{b, c}

Support threshold *s* = *3*, confidence *c* = *0.75*

1) Frequent itemsets:

{b,m} {b,c} {c,m} {c,j} {m,c,b}

2) Generate rules:

{b,m}	{b,c}	...	{m,c,b}
~~**b** → **m**: c = 4/6~~	**b** → **c**: c = 5/6	...	b,m → **c**: c = 3/4
m → **b**: c = 4/5	**c** → **b**: c = 5/6		~~b,c → **m**: c = 3/5~~
			~~b → **c,m**: c = 3/6~~
			...

If K,L,M→N is below confidence, so is K,L→M,N

For every subset ***A*** of ***I***, generate a rule $A \rightarrow I \setminus A$
Since ***I*** is frequent, then so is ***A***

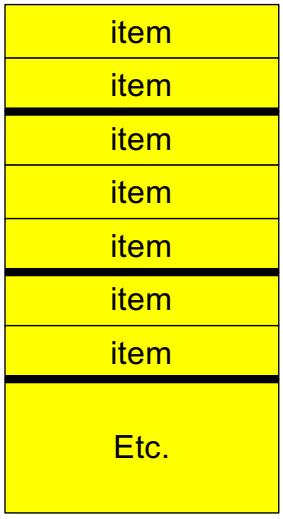
Itemsets: Computation Model

Back to finding frequent itemsets

Typically, data is kept in flat files rather than in a database system:

- Stored on disk
- Stored basket-by-basket
- Baskets are **small,** but we have many baskets and many items
  - Expand baskets into pairs, triples, etc., as you read baskets
  - Use ***k*** nested loops to generate all sets of size ***k***

Note: We want to find frequent itemsets. To find them, we have to count them. To count them, we have to generate them.



Items are positive integers, and boundaries between baskets are –1.

Computation Model

- The cost of mining disk-resident data is the ***number of disk I/Os***
- In practice, association-rule algorithms read data in ***passes***
  - all baskets read in turn
- We measure the cost by the ***number of passes*** over the data.

- **Main-Memory Bottleneck**
  - As we read baskets, we need to count something, e.g., occurrences of pairs of items.
  - The number of different things we can count is limited by the main memory.
  - Swapping counts in/out is a disaster (***why?***)

Chache misses; page faluts

Finding Frequent Pairs

Finding frequent pairs is the hardest problem.

- Frequent pairs are common, frequent triples are rare
- The probability of being frequent drops exponentially with size; the number of sets grows more slowly with size

Let's first concentrate on pairs…

The approach:

- We always need to generate all the itemsets
- *But we would only like to count (keep track) of those itemsets that, in the end, turn out to be frequent*

Naïve Algorithm

Read the file once, counting in main memory the occurrences of each pair:

- From each basket of ***n*** items, generate its ***n(n-1)/2*** pairs by two nested loops (all elements above diagonal in the ***n*** by ***n*** matrix)

Fails if (#items)^2 exceeds main memory

- 100K (Walmart) or 10B (Web pages)
  - Suppose 10^5 items at Walmart; counts are 4B integers
  - Number of pairs of items: $10^5(10^5-1)/2 = 5*10^9$
  - Therefore, $2*10^{10}$B = 20GB of memory needed

N items in a basket → N by N matrix of items – above diagonal is N(N-1)/2

Two Approaches to Counting Pairs

Assume item numbers and counters are integers of 4B

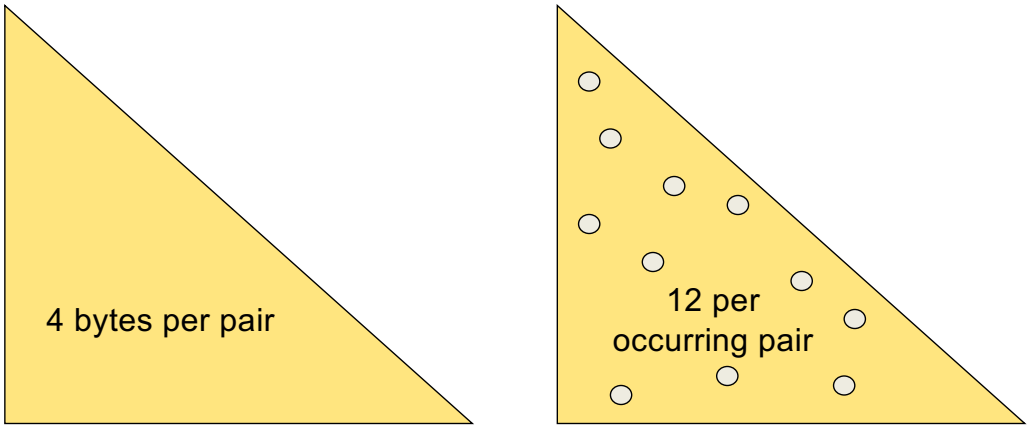
Approach 1: Count all pairs using a matrix

- 4B per pair

Approach 2: Keep a table of triples **[*i*, *j*, *c*]** = "the count of the pair of items **{*i*, *j*}** is ***c***."

- 12B for pairs with a count > 0
- Plus some additional overhead for the hashtable

Comparing the 2 Approaches



Triangular Matrix **Triples**

Comparing the 2 Approaches (cont)

Approach 1: Triangular Matrix

- **n** = total number of items
- Count pair of items **{*i*, *j*}** only if ***i*<*j***
- Keep pair counts in lexicographic order:
  - **{1,2}, {1,3},…, {1,*n*}, {2,3}, {2,4},…,{2,*n*}, {3,4},…**
- Pair {*i*, *j*} is at position **(*i* –1)(*n*– *i*/2) + *j* –1**
- Total number of pairs ***n*(*n* –1)/2**; memory demand = **2*n*^{2} B**
- **Triangular Matrix** requires **4B** per pair.

	1,2	1,3	1,4	1,5
		2,3	2,4	2,5
			3,4	3,5
				4,5

Comparing the 2 Approaches (cont)

Approach 2: Triples

- [i, j, count] – 3 integers – **12B** per occurring pair
  (but only for pairs with a count > 0)
- It beats the triangular matrix if at most 1/3 of possible pairs occur because it uses 3 times more memory per pair than the matrix
- May require extra space for retrieval structure, e.g., a hash table h(i,j) → count

Can we do better?

The problem is if we have too many items, the pairs do not fit into memory.

A-Priori Algorithm – (1) ***

A two-pass approach called ***A-Priori*** limits the memory demand.

Key idea: ***monotonicity of support***

- If a set of items appears at least ***s*** times, so does every subset, i.e., ***the support of a subset is at least as big as the support of its superset.***

The **downward closure property** of frequent patterns

- ***Any subset of a frequent itemset must be frequent***

Contrapositive for pairs: if the item ***i*** does not appear in ***s*** baskets, then no pair, including ***i,*** can appear in ***s*** baskets.

A-Priori Algorithm – (2)

Based on the candidate generation-and-test approach

A-priori pruning principle: If there is any itemset that is infrequent, its superset should not be generated/tested because it's also infrequent

[Agrawal & Srikant,@VLDB'94, Mannila, et al. @ KDD'94]

A-Priori Algorithm – (2)

Pass 1: Read baskets and count in main memory the occurrences of each **individual item**

- Requires O(n) memory, where n is #items

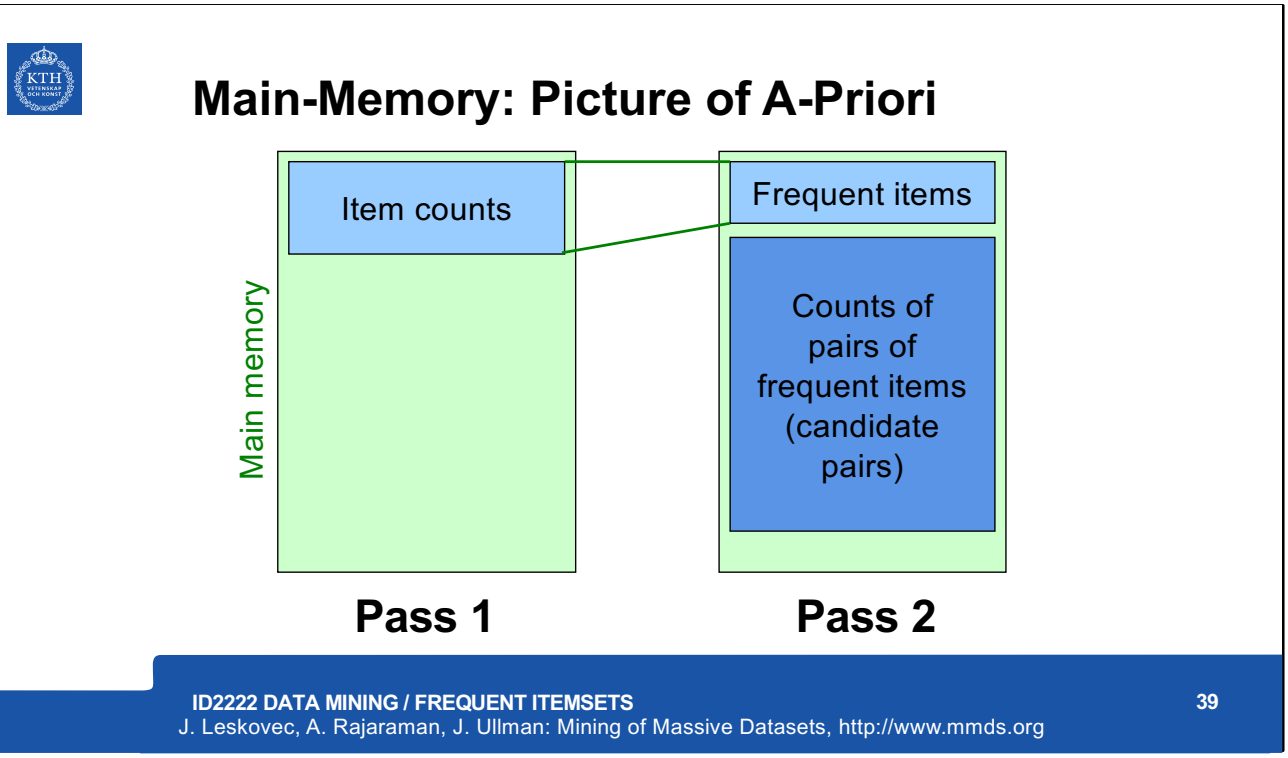
Items that appear ≥s times are the *frequent items*

Typical s=1% as many singletons will be infrequent

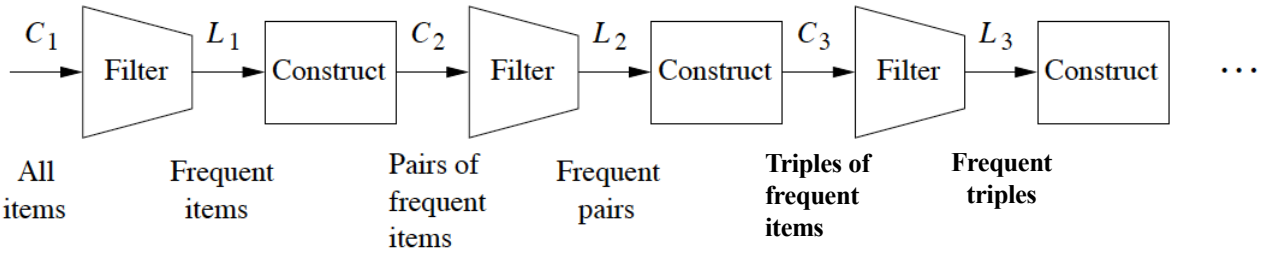
(***s*** is the support threshold)

Pass 2: Read baskets again and count only those pairs where both elements are frequent (discovered in Pass 1)

- Requires memory proportional to square of **frequent** items only (for counts) – $2m^2$ bytes instead $2n^2$
- Plus, a list of the frequent items (so you know what must be counted)


Main-Memory: Picture of A-Priori
Item counts
Frequent items
Counts of pairs of frequent items (candidate pairs)
Main memory
Pass 1
Pass 2

The Pipeline of the A-Priory Algorithm



For each *k*, we construct two sets of *k-tuples* (sets of size *k*):

- ***C_k*** = ***candidate k-tuples*** = those that might be frequent sets (support **≥ s**) based on information from the pass for ***k*–1**
- ***L_k*** = the set of truly frequent ***k***-tuples, i.e., filter only those ***k***-tuples from ***C_k*** that have support at least ***s***

Example: Steps of the A-Priori Algorithm

- C_1 = { {b} {c} {j} {m} {n} {p} } /* singletons */
- Count the support of itemsets in C_1
- Prune (filter out) non-frequent: L_1 = { b, c, j, m }
- Generate C_2 = { {b,c} {b,j} {b,m} {c,j} {c,m} {j,m} }
- Count the support of itemsets in C_2
- Prune non-frequent: L_2 = { {b,m} {b,c} {c,m} {c,j} }
- Generate C_3 = { {b,m,c} {b,c,j} {b,m,j} {c,m,j} }
- Count the support of itemsets in C_3
- Prune non-frequent: L_3 = { {b,c,m} }

Note: Candidates in C_k can be generated by combining itemsets from L_{k-1} and singletons from L_1.

Generating Candidates

Candidates in C_k can be generated by combining itemsets from L_{k-1} and singletons from L_1.

One should be careful and selective with candidate generation: for a candidate in C_k to be a frequent itemset, *all its subsets* must be frequent, not only the itemsets from L_{k-1} and L_1 that the candidate is constructed from, i.e., each of its subsets should be in the corresponding L_m, $m = 1,\ldots, k-1$

For example, in C_3, {b,m,j} cannot be frequent since {m,j} is not frequent, therefore {b,m,j} should not be even included in C_3

The Apriori Algorithm – An Example (s = 2)

Database TDB —1st scan→ C_1 → L_1 → C_2 —2nd scan→ C_2 → L_2 → C_3 —3rd scan→ L_3

TID	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

Itemset	Sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

Itemset	Sup
{A}	2
{B}	3
{C}	3
{E}	3

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

Itemset	Sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

Itemset	Sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

Itemset
{B, C, E}

Itemset	Sup
{B, C, E}	2

A-Priori for All Frequent Itemsets

One pass for each ***k*** (itemset size)

Needs room in main memory to count each candidate ***k***–tuple

For typical market-basket data and reasonable support (e.g., 1%), ***k* = 2** requires the most memory

A-Priori – Many Possible Extensions

- Association rules with intervals:
  - For example, Men over 65 have two cars
- Association rules when items are in a taxonomy
  - Bread, Butter → FruitJam
  - BakedGoods, MilkProduct → PreservedGoods
- Lower the support ***s*** as itemset gets bigger

(Self-study) PCY (Park-Chen-Yu) Algorithm

PCY reduces the number of candidates pairs and hence improves memory usage

- In pass 1, there is a lot of memory left, leverage that to help with pass 2
- Maintain a hash table with as many buckets as fit in memory
- Keep count for each bucket into which pairs of items are hashed
- Just the count, not the pairs!

Multistage improves PCY by using several successive hashtables to reduce further the number of candidate pairs

(Self-study) Frequent Itemsets in < 2 Passes

A-Priori, PCY, etc., take *k* passes to find frequent itemsets of size *k*

Can we use fewer passes?

Use 2 or fewer passes for all sizes, but may miss some frequent itemsets

- ***Random sampling***
- ***SON (Savasere, Omiecinski, and Navathe)***
- ***Toivonen (see textbook)***

Confidence of an Association Rule

Association rule $X \rightarrow Y$
"if a basket contains all of X, then it is ***likely*** to contain Y."

- Where X and Y are itemsets; X, $Y \subset I$, and $X \cap Y = \emptyset$

Confidence of $X \rightarrow Y$ is the fraction of the baskets with all of X that also contain Y.
Confidence of $X \rightarrow Y$ is the probability of Y given X.

$$conf(X \rightarrow Y) = P(Y|X) = \frac{P(X \cup Y)}{P(X)} = \frac{support(X \cup X)}{support(X)}$$

- Where *P* is an *empirical probability* (a.k.a., also a relative frequency or experimental probability) calculated on a given set of baskets.
- Remind that the conditional probability of an event *A* given *B*:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

- Note that ∪ ("union") is used above for sets; whereas ∩ ("and") is used for events.

Confidence of this association rule is the probability of j given $I = \{i_1,\ldots,i_k\}$