Frequent Itemsets

Alexander Schönhuth

Faculty of Technology

Bielefeld University
July 2, 2020

LEARNING GOALS TODAY

- The Market-Basket Model
- Frequent Itemsets: Definition and Applications
- Association Rules
- The A-Priori Algorithm
  - Data Representation
  - Runtime and Space Considerations
  - Monotonicity
  - The Algorithm
- The Algorithm of Park, Chen and Yu (PCY)

Frequent Itemsets
Introduction

Frequent Itemsets: Overview

Foundations

- There are *items* available in the market
- There are *baskets*, sets of items having been purchased together
- A *frequent itemset* is a set of items that is found to commonly appear in many baskets
- The *frequent-itemset problem* is to identify frequent itemsets

Market-Basket Model

Market-basket model

- The market-basket model is a *many-many-relationship*
  - One basket holds many items
  - One item appears in several baskets
- Each basket is an itemset, i.e. a set of (one or several) items
- Usually, the number of items in a basket is small compared to number of items overall
- Number of baskets is usually large; too large to fit in main memory
- Data usually is a sequence of baskets

Frequent Itemsets: Definition

Definition [Frequent Itemset]:

- Let $s > 0$ be a *support threshold*
- Let I be a set of items
- $\text{supp}(I)$, the *support* of I, is the number of baskets in which I appears as a subset

An itemset I is referred to as *frequent* if

$$\text{supp}(I) \geq s \tag{1}$$

that is, if the support of I is at least the support threshold

FREQUENT ITEMSETS: EXAMPLE

Baskets

1. {cat, and, dog, bites}
2. {news, claims, a, cat, mated, with, a, dog, and, produced, viable, offspring}
3. {cat, killer, likely, is, a, big, dog}
4. {professional, free, advice, on, dog, training, puppy, training}
5. {cat, and, kitten, training, and, behavior}
6. {dog, cat, provides, training, in, Oregon}
7. {dog, and, cat, is, a, slang, term, used, by, police, officers, for, a, male-female, relationship}
8. {shop, for, your, show, dog, grooming, and, pet, supplies}

- E.g. supp({dog}) = 7, supp({and}) = 5, supp({dog, and}) = 4
- Let the support threshold $s = 3$
- Exist 5 frequent singletons: {dog},{cat},{a},{and},{training}
- Exist 5 frequent doubletons: {dog, a},{dog, and},{dog, cat},{cat, a},{cat, and}
- Exists 1 frequent triple: {dog,cat,a}

Frequent Itemsets: Applications

- *Retailers / Supermarkets / Chain stores*
  - *Items:* Products offered
  - *Baskets:* Sets of products purchased by one customer during one shopping run
  - *Frequent Itemsets:* Products purchased together unusually often
    ☞ Beer and diapers
- *Related concepts*
  - *Items:* Words, excluding stop words
  - *Baskets:* News articles, documents
  - *Frequent Itemsets:* Groups of words representing joint concept
- *Plagiarism*
  - *Items:* Documents
  - *Baskets:* Sentences
  - *Frequent Itemsets:* Documents containing unusually many sentences in common

Association Rules

- Let j be an item and I be an itemset
- An association rule

  $$I \to j$$

  expresses that if I is likely to appear in a basket, so is j
- In other words, if I shows in basket, one is confident to assume that j does, too

Definition [Confidence]:
The *confidence* of a rule $I \to j$ is defined as

$$\frac{\text{supp}(I \cup \{j\})}{\text{supp}(I)} \tag{2}$$

that is the fraction of I containing baskets that also contain j.

Association Rules: Confidence

Definition [Confidence]:
The *confidence* of a rule $I \to j$ is defined as

$$\frac{\text{supp}(I \cup \{j\})}{\text{supp}(I)}$$

that is the fraction of I containing baskets that also contain j.

Example from above

- Confidence of $\{cat, dog\} \to and$ is $3/5$
- Confidence of $\{cat\} \to kitten$ is $1/6$

Association Rules: Interest

- Let n be the number of baskets overall
- Confidence for $I \to j$ can be meaningless if fraction of baskets containing j is large
- Confidence may just reflect that fraction
- So presence of I does not increase confidence to see j as well
- *Interest* is supposed to put this into context

Definition [Interest]:
The *interest* of a rule $I \to j$ is defined as

$$\frac{\text{supp}(I \cup \{j\})}{\text{supp}(I)} - \frac{\text{supp}(\{j\})}{n} \tag{3}$$

that is the confidence of $I \to j$ minus the fraction of baskets that contain j

Association Rules: Interest

Definition [Interest]:
The *interest* of a rule $I \to j$ is defined as

$$\frac{\text{supp}(I \cup \{j\})}{\text{supp}(I)} - \frac{\text{supp}(\{j\})}{n}$$

that is the confidence of $I \to j$ minus the fraction of baskets that contain j

Examples

- $\{diapers\} \to beer$ was found to have great interest
- $\{dog\} \to cat$ has interest $5/7 - 3/4 = -0.036$
- $\{cat\} \to kitten$ has interest $1/6 - 1/8 = 0.042$

Frequent Itemsets to Association Rules

Situation

- Consider frequent itemsets of "reasonably high" support s
  - Note that each frequent itemset suggests to be acted upon
    ☞ keep their number reasonably low
  - Reasonably high often means about 1% of baskets
- Confidence for a rule $I \to j$ should be at least (about) 50%
  ☞ Support for $I \cup \{j\}$ also fairly high

Procedure

- Assume all I with $\text{supp}(I) \geq s$ have been mined
- For J of n items with $\text{supp}(J) \geq s$, there are n possible association rules $J \setminus \{j\} \to J$
- $\text{supp}(J) \geq s$ implies $\text{supp}(J \setminus \{j\}) \geq s$
- Confidence of $J \setminus \{j\} \to J$ is easily computed as

$$\frac{\text{supp}(J)}{\text{supp}(J \setminus \{j\})}$$

Mining Frequent Itemsets
The A-Priori Algorithm

Market-Basket Data: Representation

- Market-basket data is stored in a file basket-by-basket
  - If items refer to identifiers, for example $\{3, 36, 99\}\{6, 78, 11\}\ldots$
- *Assumption:* Average size of basket is rather small
- *Usually,* file does not fit in main memory
- Generating all subsets of size k for a basket of size n requires

$$\binom{n}{k} \approx \frac{n^k}{k!}$$

  runtime

- This often is little time because
  - n was assumed to be small
  - k is usually very small
  - When k is large, one can virtually reduce n further by removing infrequent items

Market-Basket Data: Runtime Consideration

Insight

- Runtime is dominated by transferring data from disk to main memory
- *Consequence:* Processing all baskets is proportional to size of file
- *Runtime of algorithm* is proportional to number of passes through file
- For a *fast frequent itemset mining* algorithm:

Limit number of passes through basket file

Use of Main Memory

- *Issue:* One needs to store counts for itemsets of size k
  - There could be many such itemsets
  - How to store these counts?
- *Consequence:* There is a limit on the number of items an algorithm can deal with
- *Example:*
  - Let there be n items
  - For counting pairs, we need to store $\binom{n}{2} \approx n^2/2$ counts
  - Integers of 4 bytes: need $2n^2$ bytes to store counts
  - Consider machine of 2 GB, or $\approx 2^{31}$ bytes of main memory
  - Then $n < 2^{15} \approx 33\,000$ is required
- *Note:* Items can be hashed to integers, if they are not integers

Storing Itemset Counts: The Triangular-Matrix Method

- In the following, consider storing itemsets of size 2
  - Remember that support threshold is quite large in real applications
  - So, many more pairs than triples, quadruples and so on in real applications
- *Insight:* Storing counts $a[i,j]$ in matrix $A = (a[i,j])_{1 \leq i < j \leq n} \in \mathbb{N}^{n \times n}$ wastes half of A
- *Solution:* Store count for pair of items $\{i,j\}, 1 \leq i < j \leq n$ in

$$a[k] \quad \text{where} \quad k = (i-1)(n - \frac{i}{2}) + j - i \tag{4}$$

This stores pairs in lexicographical order

$$\{1,2\}, \{1,3\}, ..., \{1,n\}, \{2,3\}, ..., \{2,n\}, ..., \{n-2,n\}, \{n-1,n\}$$

Storing Itemset Counts: The Triples Method

- Store triples $[i, j, c]$ for all pairs $\{i, j\}$ whose count $c > 0$
- For example, do this with hash table, hashing i, j as search key
- *Advantage:* Does not require space for pairs $\{i, j\}$ of count zero
- *Disadavantage:* Requires three times the space if $c > 0$
- *Rationale:* Triangular matrix method better if at least $1/3$ of the $\binom{n}{2}$ pairs appear in basket

Storing Itemset Counts: Example

Example

- Consider
  - 100 000 items
  - 10 000 000 baskets of
  - 10 items each
- Triangular-matrix method: $\binom{10^5}{2} \approx 5 \times 10^9$ integer counts
- Triples method: $10^7 \binom{10}{2} \approx 4.5 \times 10^8$ counts, making for $3 \times 4.5 \times 10^8 = 1.35 \times 10^9$ integers to be stored
- Triples method proves to be more appropriate

MONOTONICITY

THEOREM [MONOTONICITY]:

- Let s be the support threshold.
- Let I, J be sets such that $J \subseteq I$

Then if I is frequent, any subset J of I is, too:

$$\text{supp}(I) \geq s \quad \text{implies} \quad \text{supp}(J) \geq s \tag{5}$$

PROOF.
Each basket that holds I also holds J, as J is contained in I. So, the number of baskets that hold J is at least as large as the number of baskets that hold I. □

Maximal Frequent Itemset

Definition [Maximal Frequent Itemset]:

- Let s be the support threshold.
- Let I be frequent, that is $\text{supp}(I) \geq s$.

I is said to be *maximal* if no superset of I is frequent:

$$\text{for all } J \supsetneq I : \text{supp}(J) < s \tag{6}$$

Example (from above):

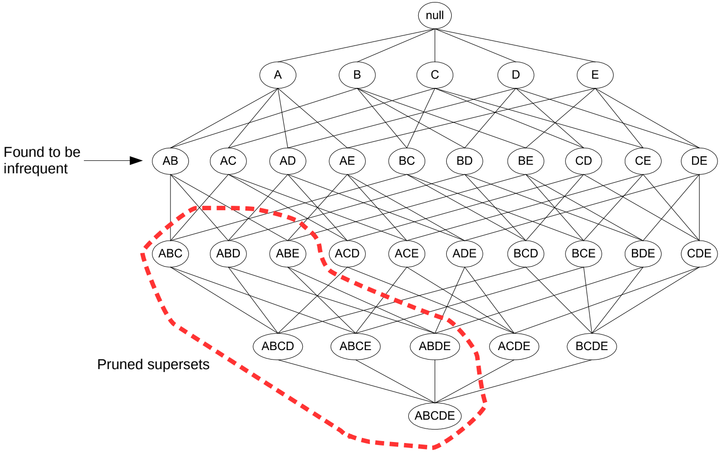
- At support threshold $s = 3$, we found frequent pairs $\{dog, a\}, \{dog, and\}, \{dog, cat\}, \{cat, a\}, \{cat, and\}$
- $\{dog, cat, a\}$ was found the only frequent triple

☞ $\{dog, cat, a\}, \{dog, and\}$ and $\{cat, and\}$ are maximal, while $\{dog, a\}, \{dog, cat\}, \{cat, a\}$ are not

Note on Counting Pairs

- The number of frequent pairs is larger than frequent triples, quadruples, ... why?
- For making sense, number of (maximal) frequent itemsets is supposed to be sufficiently small
  - Human applicants need to work it out on all of them
- So, support threshold is set sufficiently high
- Any maximal frequent itemset holds many more smaller, non-maximal frequent itemsets
- The resulting situation implies that there are many more frequent pairs than triples, many more frequent triples than quadruples, and so on
- *Important:*
  - Still, the possible number of triples, quadruples is (much) greater than pairs
  - Any good frequent itemset *algorithm needs to avoid running through all possible triples, quadruples, and so on*

Monotonicity to the Rescue



Itemsets for items A,B,C,D,E

Neglecting supersets of infrequent pair {A,B}

Adopted from `mmds.org`

A-Priori Algorithm: Motivation

In the following, we focus on determining frequent pairs.

*N*aive Approach

Consider the algorithm

- For each basket, use double loop to generate all pairs contained in it
- For each pair generated, add 1 to its count
- Store counts using triangular or triples method
- At the end, run through all pairs and determine those whose counts exceed support threshold s
- *Benefit:* Only one pass through all baskets
- *Issue:* Number of pairs considered usually does not fit in main memory

A-Priori Algorithm: Motivation

In the following, we focus on determining frequent pairs.

*N*aive Approach

- *Possible Benefit:* Only pass through all baskets
- *Issue:* Number of pairs considered usually does not fit in main memory

*S*olution: A-Priori-Algorithm

- Have *two passes through baskets* instead of one
- In first run, determine candidate pairs, for which counts are stored
- In second run, determine counts for candidate pairs
- Finally filter for frequent pairs

A-Priori Algorithm: First Pass

Create and Maintain Two Tables

- *First table A:* Let x be an item name, then $A[x]$ reflects that x is the $A[x]$-th item in the order of their appearance in the basket file
- *Second table B:* Let k be an item number, then $B[k]$ is the number of baskets in which item number k appears

Read Baskets: Fill Table B

- For each basket, for each item x in the basket, set

$$B[A[x]] = B[A[x]] + 1 \tag{7}$$

- That is, iteratively increase item counts while running through all items in all baskets

A-Priori Algorithm: Second Pass I

- Let n be the number of items
- Let m be the number of items found to be *frequent*
- By user constraints, usually $m << n$

Create Third Table

- *Third table C:* Let $1 \leq k \leq n$ be an item number. Then

$$C[k] = \begin{cases} 0 & \text{if item number } k \text{ is not frequent} \\ l & \text{if item number } k \text{ was found the } l\text{-th frequent item} \end{cases} \quad (8)$$

So, $C \in \{0, 1, ..., m\}^n$, where

- $C[k] = 0$ $n - m$ times
- $C[k] = i, 1 \leq i \leq m$ exactly one time
- $0 < C[k_1] < C[k_2]$ implies $k_1 < k_2$, expressing that C preserves the order of appearance of items

A-Priori Algorithm: Second Pass II

Count Pairs Data Structure

- Use either triangular or triples method data structure to hold counts
  - For using triangular method, renumbering necessary
- By monotonicity, a pair can only be frequent, if both items are frequent
- So, space required for is $O(m^2)$ rather than $O(n^2)$
  ☞ $m << n$ implies $m^2 << n^2$, so fits in main memory!

Examine Baskets

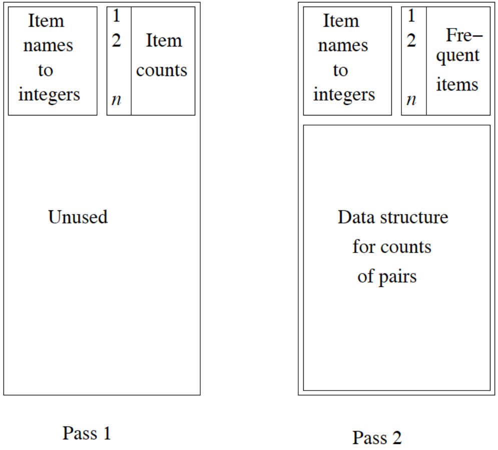
1. For each basket, for each item x, see whether

$$C[A[x]] > 0 \quad \text{that is, whether } x \text{ is frequent} \tag{9}$$

2. Using double loop, generate all pairs of frequent items in the basket
3. For each such pair, increase count by one in pair count data structure

Eventually: examine which pairs are frequent in pair count data structure

A-PRIORI ALGORITHM: MAIN MEMORY USAGE



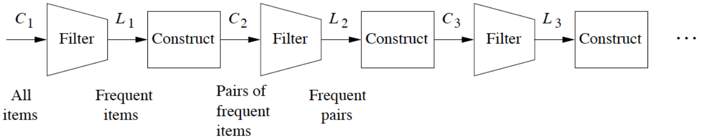
Use of main memory during A-Priori passes

Adopted from `mmds.org`

A-Priori Algorithm: All Frequent Itemsets

- *One extra pass* for each $k > 2$ to mine frequent itemsets of size k
- The A-Priori algorithm proceeds iteratively
  - Mining frequent itemsets of size $k + 1$ is based on knowing frequent itemsets of size k
- Each iteration consists of two steps for each k:
  - Generate a candidate set C_k
  - Filter candidate set C_k to produce L_k, the truly frequent itemsets of size k
- The algorithm terminates at first k where L_k is empty
  - Monotonicity says we are done mining frequent itemsets

A-PRIORI ALGORITHM: CANDIDATE GENERATION AND FILTERING



A-Priori algorithm: Alternating between candidate generation and filtering

Adopted from `mmds.org`

- *Construct:* Let C_k be all itemsets of size k, every $k-1$ of which belong to L_{k-1}
- *Filter:* Make a *pass through baskets* to count members of C_k; those with count exceeding s will be part of L_k
  - For storing counts for itemsets of size k, extend triples method
  - E.g. storing quadruples for frequent triples, and so on...

UNIVERSITÄT BIELEFELD

A-Priori Algorithm Extensions
The PCY Algorithm

Bottleneck: Size of C_2

- The predominant bottleneck in most applications of A-Priori is the size of C_2, the candidate pairs
- Several algorithms address to trim down that size
- Exemplary algorithms:
  - The algorithm of Park, Chen and Yu *(PCY algorithm)*
  - The Multistage algorithm
  - The Multihash algorithm
- We will briefly treat the PCY algorithm here

The PCY Algorithm

- *Observation:* Much of main memory during first pass of A-Priori remains unused
- Use that space for a hash table H that
  - hashes pairs of items $\{i, j\}$ to
  - buckets holding integers $H[\{i, j\}] \in \mathbb{N}$, where

$$H[\{i, j\}] \quad \text{is number of times any pair hashed to that bucket} \tag{10}$$

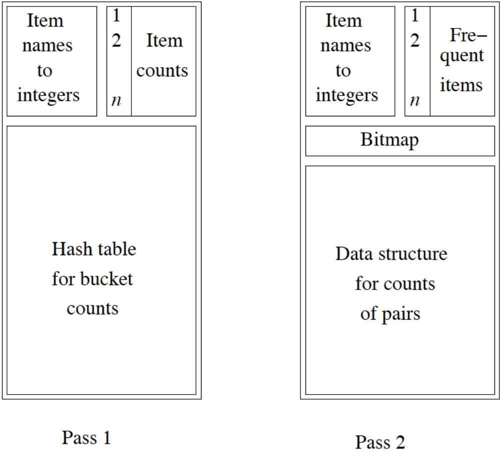
- To construct H, use double loop through baskets:
  - hash each resulting pair to bucket
  - increase the integer in that bucket by one
- A *frequent bucket b* exceeds the support threshold s

The PCY Algorithm

- A *frequent bucket* b exceeds the support threshold s
- So, for any bucket b:
  - If b is infrequent, none of the pairs that hashed to b are frequent
  - If b is frequent, pairs hashing to it could be frequent
- *Definition of* C_2*:* For $\{i, j\} \in C_2$, both
  - i and j must be frequent
  - $\{i, j\}$ must hash to a frequent bucket
- *Use of* C_2 *in second pass:*
  - Transform H into bitmap H'

$$H'[\{i,j\}] = \begin{cases} 1 & \text{if } H[\{i,j\}] \geq s \\ 0 & \text{if } H[\{i,j\}] < s \end{cases} \tag{11}$$

PCY Algorithm: Main Memory Usage



Use of main memory during A-Priori passes

Adopted from `mmds.org`

MATERIALS / OUTLOOK

- See *Mining of Massive Datasets*, sections 6.1, 6.2, 6.3.1, 6.4.1, 6.4.2, 6.4.5
- As usual, see `http://www.mmds.org/` in general for further resources
- Next lecture: "Mining Data Streams II / PageRank I"
    - See *Mining of Massive Datasets* 9.1–9.5