COMP336 — Big Data

Week 10 Lecture 1: Frequent Itemsets

Diego Mollá

Department of Computer Science
Macquarie University

COMP336 2018H1

Programme

Reading

- Leskovec, Rajaraman, Ullman (2014): Mining of Massive Datasets, Chapter 6. `http://www.mmds.org/`

Programme

Programme

1. The Market-Basket Model
   - Association Rules
   - Frequent Itemsets
   - Finding Association Rules
2. Finding Frequent Itemsets
3. Scaling Up to Big Data

What Are Association Rules?

- Study of "what goes with what":
  - "Customers who bought X also bought Y".
  - What symptoms go with what diagnosis.
- Transaction-based or event-based.
- Also called "market basket analysis" and "affinity analysis".
- Originated with study of customer transactions databases to determine associations among items purchased.

Used in Many Recommender Systems

A Working Example

Purchase of phone faceplates

A store that sells accessories for mobile phones runs a promotion on faceplates. Customers who purchase multiple faceplates from a choice of six different colours get a discount. The store managers, who would like to know what colours of faceplates customers are likely to purchase together, collected a transaction database.

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

Candidate Rules

Example of a rule

IF {red, white} THEN {green}

Terminology

Antecedent {red, white}.

Consequent {green}.

Item set {red, white, green, orange, blue, yellow}.

The antecedent and the consequent are disjoint:

- They have no items in common.

Many Rules are Possible

For example: Transaction 1 supports several rules, such as

1. "If red, then white" ("If a red faceplate is purchased, then so is a white one").
2. "If white, then red".
3. "If red and white, then green".
4. ...

Frequent Item Sets

- Ideally, we want to create all possible combinations of items.
- Problem: computation time grows exponentially as # items increases.
- Solution: consider only "frequent item sets".
- Criterion for frequent: support.

Programme

Definition of Frequent Itemsets

- A set of items that appears in many baskets is said to be "frequent".
- Assume there is a number s, called the support threshold.
- If I is a set of items, the support for I is the number of baskets for which I is a subset.
- We then say that I is frequent if its support is s or more.

In our Working Example

The support for the item set {red, white} is 4

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

Applications of Frequent Itemsets I

Brick-and-Mortar Retailing

Items: Products.

Baskets: Sets of products someone bought in one trip to the store.

- By finding frequent itemsets, a retailer can learn what is commonly bought together and arrange the marketing and sales strategy accordingly.
- A famous example is the itemset "diapers and beer".
  - Strategy: run a sale on diapers and simultaneously raise the price of beer.
  - Strategy: place cans of beer in the aisle for baby products.

Applications of Frequent Itemsets II

Plagiarism Detection

Items: Documents.

Baskets: Sentences.

Clinical Analysis

Items Drugs and side-effects.

Baskets: Patients.

- Has been used to detect combinations of drugs that result in particular side-effects.
- This approach requires an extension: Absence of an item needs to be observed as well as presence.

Programme

1. The Market-Basket Model
   - Association Rules
   - Frequent Itemsets
   - Finding Association Rules
2. Finding Frequent Itemsets
3. Scaling Up to Big Data

Measures of Performance I

Confidence

The % of antecedent transactions that also have the consequent item set.

- Confidence is the conditional probability of taking the consequent given that we have taken the antecedent:

$$\begin{aligned} Confidence &= P(consequent|antecedent) \\ &= \frac{P(antecedent\&consequent)}{P(antecedent)} \end{aligned}$$

- However, if the antecedent or consequent have strong support, this value is not useful.

Measures of Performance II

Lift Ratio

$$Lift\ ratio = \frac{Confidence}{Benchmark\ confidence}$$

- The benchmark confidence is the percentage of transactions in the entire dataset that have the consequent:

$$Benchmark\ confidence = \frac{no.\ transactions\ with\ consequent\ item\ set}{no.\ transactions\ in\ database}$$

- Lift ratio > 1 indicates that the rule is more useful than just selecting transactions randomly.

Process of Rule Selection I

Generate all rules that meet specified support & confidence.

1. Find frequent item sets (those with sufficient support — see above).
2. From these item sets, generate rules with sufficient confidence (or lift ratio, whatever criteria we set).

Process of Rule Selection II

Example

Rules from {red, white} item set.

- Support for {red,white} = 40%; Support for {red} = 50%; Support for {white} = 80%.

The rules are:

1. IF {red} THEN {white}
   - Confidence = 40% / 50% = 80%.
   - Lift ratio = 80% / 80% = 1.
2. IF {white} THEN {red}
   - Support for {red,white} = 40%.
   - Confidence = 40% / 80% = 50%.
   - Lift ratio = 50% / 50% = 1.

If confidence cutoff is 70%, report only rule 1. If lift ratio $>$ 1, none of these rules are useful.

Interpretation

- Lift ratio shows how effective the rule is in finding consequents (useful if finding particular consequents is important).
- Confidence shows the rate at which consequents will be found (useful in learning costs of promotion).
- Support measures overall impact.

Caution: The Role of Chance

- Random data can generate apparently interesting association rules.
- The more rules you produce, the greater this danger.
- Rules based on large numbers of records are less subject to this danger.

Programme

1. The Market-Basket Model
2. Finding Frequent Itemsets
   - The A-Priori Algorithm
   - An Efficient Implementation of A-Priori
3. Scaling Up to Big Data

Programme

Finding Frequent Itemsets

- A naïve approach to find frequent itemsets of length k would use a nested loop that generates all possible itemsets.
- But in a real scenario there may be many possible different items.
- This approach would be too time-consuming and may exceed memory capacity.
  - Suppose 10^5 items, counts are 4-byte integers.
  - Number of pairs of items: $10^5(10^5 - 1)/2 = 5 * 10^9$.
  - This would require $2 * 10^{10}$ bytes (20Gb of memory).
  - What about counting triples??

Intuition of the A-Priori Algorithm

Monotonicity

- If a set of items I appears at least s times, so does every subset J of I.
- Consequently: if a set of items J has a support below s, so does every set I containing J.
- Applied to a pair of items: If an item i does not appear in s baskets, then no pair including i can appear in s baskets.

Generating Frequent Item Sets

The A-Priori Algorithm

For *k* products . . .

1. User sets a minimum support criterion.
2. First pass: generate list of one-item sets that meet the support criterion.
3. Second pass: use the list of one-item sets to generate list of two-item sets that meet the support criterion.
4. Third pass: use list of two-item sets to generate list of three-item sets.
5. . . .
6. Continue up through k-item sets.

Example with the Phone Faceplates Dataset

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
green	2
orange	3
blue	5
yellow	1

Two-item sets ≥ 3

Set	Support
red, white	4
red, orange	1
red, blue	3
white, orange	3
white, blue	4
orange, blue	0

Three-item sets

Set	Support
red, white, blue	2

Example with the Phone Faceplates Dataset

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
green	2
orange	3
blue	5
yellow	1

Two-item sets ≥ 3

Set	Support
red, white	4
red, orange	1
red, blue	3
white, orange	3
white, blue	4
orange, blue	0

Three-item sets

Set	Support
red, white, blue	2

Example with the Phone Faceplates Dataset

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
green	2
orange	3
blue	5
yellow	1

Two-item sets ≥ 3

Set	Support
red, white	4
red, orange	1
red, blue	3
white, orange	3
white, blue	4
orange, blue	0

Three-item sets

Set	Support
red, white, blue	2

Example with the Phone Faceplates Dataset

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
green	2
orange	3
blue	5
yellow	1

Two-item sets ≥ 3

Set	Support
red, white	4
red, orange	1
red, blue	3
white, orange	3
white, blue	4
orange, blue	0

Three-item sets

Set	Support
red, white, blue	2

Example with the Phone Faceplates Dataset

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
green	2
orange	3
blue	5
yellow	1

Two-item sets ≥ 3

Set	Support
red, white	4
red, orange	1
red, blue	3
white, orange	3
white, blue	4
orange, blue	0

Three-item sets

Set	Support
red, white, blue	2

Programme

Considerations for Efficiency

- We assume that there are too many baskets to fit in main memory.
- So the file of baskets must be read from disk.
- We also assume that the number of items in each basket is relatively small.
- It will generally take much longer time to read a basket than to compute pairs or triples of items in the basket.
- There will normally be very few itemsets with size larger than 3 and support over the threshold.

We'd better read the basket file sequentially, and as few times as possible.

A-Priori Algorithm — Pass 1

1. Initialise counters for all items to zero.
2. FOR each basket:
3.   Read basket from disk.
4.   FOR each item in basket:
5.   Increment item's count.

Pass 1 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	1
white	1
green	1
orange	0
blue	0
yellow	0

Pass 1 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	1
white	2
green	1
orange	1
blue	0
yellow	0

Pass 1 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	1
white	3
green	1
orange	1
blue	1
yellow	0

Pass 1 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	2
white	4
green	1
orange	2
blue	1
yellow	0

Pass 1 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	3
white	4
green	1
orange	2
blue	2
yellow	0

Pass 1 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	3
white	5
green	1
orange	2
blue	3
yellow	0

Pass 1 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	3
white	6
green	1
orange	3
blue	3
yellow	0

Pass 1 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	4
white	7
green	2
orange	3
blue	4
yellow	0

Pass 1 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
green	2
orange	3
blue	5
yellow	0

Pass 1 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
green	2
orange	3
blue	5
yellow	1

Pass 1 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
green	2
orange	3
blue	5
yellow	1

A-Priori Algorithm — Storage Considerations for Pass 2

- Storing all possible pairs of items might not fit on memory.
- So, we only store pairs of items that we are counting.
- Pairs of items are stored as triples:

$$(i, j, c)$$

i: Item 1
j: Item 2
c: Count (initialised to zero)

A-Priori Algorithm — Pass 2

```
FOR each basket:
    Read basket from disk.
    FOR each pair (i, j) of items in basket:
        IF both i and j are frequent (according to first pass):
        IF (i, j) has been counted:
            Increment count for (i, j)
        ELSE:
            Add counter for (i, j) initialised to 1.
```

Pass 2 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
orange	3
blue	5

Two-item sets ≥ 3

i	j	c
red	white	1

Only 5 pairs were counted from a total of 15.

Pass 2 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
orange	3
blue	5

Two-item sets ≥ 3

i	j	c
red	white	1
white	orange	1

Only 5 pairs were counted from a total of 15.

Pass 2 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
orange	3
blue	5

Two-item sets ≥ 3

i	j	c
red	white	1
white	orange	1
white	blue	1

Only 5 pairs were counted from a total of 15.

Pass 2 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
orange	3
blue	5

Two-item sets ≥ 3

i	j	c
red	white	2
white	orange	2
white	blue	1
red	orange	1

Only 5 pairs were counted from a total of 15.

Pass 2 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
orange	3
blue	5

Two-item sets ≥ 3

i	j	c
red	white	2
white	orange	2
white	blue	1
red	orange	1
red	blue	1

Only 5 pairs were counted from a total of 15.

Pass 2 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
orange	3
blue	5

Two-item sets ≥ 3

i	j	c
red	white	2
white	orange	2
white	blue	2
red	orange	1
red	blue	1

Only 5 pairs were counted from a total of 15.

Pass 2 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
orange	3
blue	5

Two-item sets ≥ 3

i	j	c
red	white	2
white	orange	3
white	blue	2
red	orange	1
red	blue	1

Only 5 pairs were counted from a total of 15.

Pass 2 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
orange	3
blue	5

Two-item sets ≥ 3

i	j	c
red	white	3
white	orange	3
white	blue	3
red	orange	1
red	blue	2

Only 5 pairs were counted from a total of 15.

Pass 2 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
orange	3
blue	5

Two-item sets ≥ 3

i	j	c
red	white	4
white	orange	3
white	blue	4
red	orange	1
red	blue	3

Only 5 pairs were counted from a total of 15.

Pass 2 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
orange	3
blue	5

Two-item sets ≥ 3

i	j	c
red	white	4
white	orange	3
white	blue	4
red	orange	1
red	blue	3

Only 5 pairs were counted from a total of 15.

Pass 2 — Example

The Dataset

Transaction	Faceplate colours purchased			
1	red	white	green	
2	white	orange		
3	white	blue		
4	red	white	orange	
5	red	blue		
6	white	blue		
7	white	orange		
8	red	white	blue	green
9	red	white	blue	
10	yellow			

One-item sets with support ≥ 3

Set	Support
red	5
white	8
orange	3
blue	5

Two-item sets ≥ 3

i	j	c
red	white	4
white	orange	3
white	blue	4
red	orange	1
red	blue	3

Only 5 pairs were counted from a total of 15.

Programme

1. The Market-Basket Model
2. Finding Frequent Itemsets
3. Scaling Up to Big Data
   - PCY Algorithm
   - Multistage Algorithm

Programme

1. The Market-Basket Model
2. Finding Frequent Itemsets
3. Scaling Up to Big Data
    - PCY Algorithm
    - Multistage Algorithm

Park-Chen-Yu Algorithm

Observation of A-Priori algorithm

- In pass 1, most memory is idle.
- Pass 2 will usually be the most memory-consuming.

$\Rightarrow$ Can we use the idle memory to reduce memory required in pass 2?

Solution by PCY

At pass 1, while counting items, maintain a hash table that will count pairs of items.

PCY Algorithm — First Pass

Algorithm

1. FOR each basket:
2. FOR each item in basket:
3. Add 1 to item's count.
4. FOR each pair of items in basket:
5. Hash the pair to a bucket.
6. Add 1 to the count for that bucket.

Notes

1. We want to use as many hash buckets as we can fit in memory.
2. Generalisation of a Bloom filter: We need to count the pairs so that we can check if their support is at least s.

PCY Algorithm — Second Pass

Algorithm

1. FOR each basket:
2. FOR each pair in the basket:
3. IF both items in the pair are frequent:
4. Use hash function of pass 1 to hash the pair to a bucket.
5. IF the count for that bucket is $\geq s$:
6. Count the pair.

Note

- Several pairs might have hashed to the same bucket.

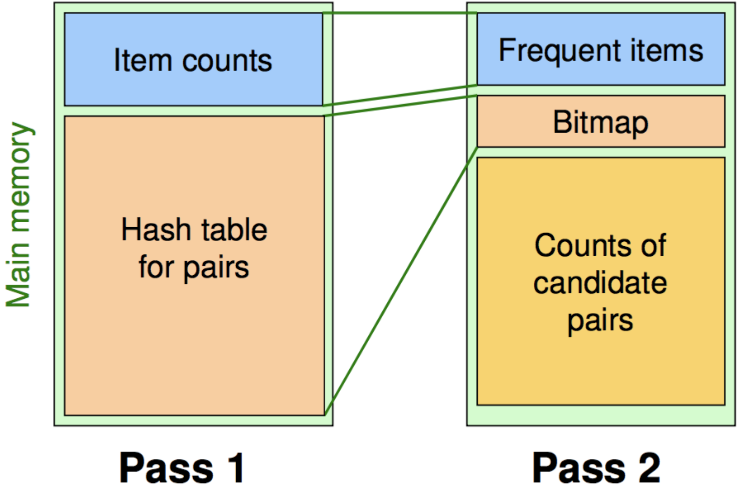
$\Rightarrow$ Even if a pair hashes to a bucket with count $\geq s$, it might not be frequent.

$\Rightarrow$ If a pair does not hash to a bucket with count $\geq s$, we are certain that it is not frequent.

What is the Advantage of Hash Tables?

- Like in the Bloom filter, the space to store the buckets in memory is smaller than the space of storing pairs of items and their counts.
- This is so because the number of buckets is less than the number of pairs of items.
- We only need to know what buckets are over a threshold, so after the first pass we can compact the hash table into a bit vector.
    - Bit $= 1$ if bucket is $\geq s$.
- We can actually chain hash tables to reduce the number of pairs to count... see below.

Main Memory: Picture of PCY



http://www.mmds.org

Programme

The Multistage Algorithm

- The multistage algorithm attempts to reduce the number of pairs to be counted.
- It does this by adding a new pass through the file of baskets.

Pass 1

Count individual items and maintain a hash table of counts of pairs as in PCY algorithm.

Pass 2

Create a new hash table, using an independent hash function, that counts pairs whose items were hashed as frequent in pass 1 and both items in the pair are frequent as determined by pass 1.

Pass 3

Count pairs that hashed to frequent buckets in both hashes.

Multistage Algorithm — Pass 1

Algorithm

1. FOR each basket:
2. FOR each item in basket:
3. Add 1 to item's count.
4. FOR each pair of items in basket:
5. Use hash function 1 to hash the pair to a bucket.
6. Add 1 to the count for that bucket.

Note

This is exactly the same as pass 1 of PCY.

Multistage Algorithm — Pass 2

Algorithm

1. FOR each basket:
2. FOR each pair in the basket:
3. IF both items in the pair are frequent:
4. Use hash function 1 to hash the pair to a bucket.
5. IF the count for that bucket is $\geq s$:
6. Use hash function 2 to hash the pair to a bucket.
7. Add 1 to the count for that bucket..

Note

We keep two separate hash tables, one per hash function.

Multistage Algorithm — Pass 3

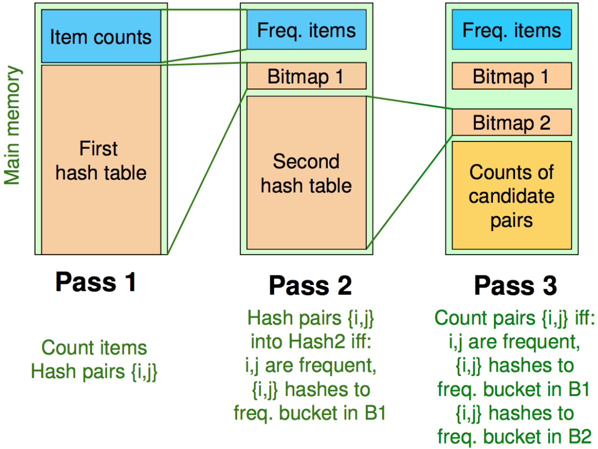
Algorithm

1. FOR each basket:
2. FOR each pair in the basket:
3. IF both items in the pair are frequent:
4. Use hash function 1 to hash the pair to a bucket.
5. Use hash function 2 to hash the pair to a bucket.
6. IF the count for both buckets is $\geq s$:
7. Count the pair.

Do we really need to maintain both hashes in pass 3?

- It would be tempting to avoid checking the first hash in pass 3.
- But there may be pairs that hash to a frequent bucket in hash 2 but do not hash to a frequent bucket in pass 1.
- Remember: due to collisions, an infrequent pair may end up hashed in the same bucket as a frequent pair.
- We therefore need to make sure that the two hash functions are independent.

Main Memory: Multistage



http://www.mmds.org

Take-home Messages

- The Marked-Basket model.
- General approach to find association rules.
- Counting item pairs are the costliest part of finding association rules.
- The A-Priori algorithm.
- Efficient implementation of A-Priori.
- Variants for big data.

What's Next

Week 11

- Large-Scale Machine Learning (I)