Algorithms for Data Science
Frequent Itemsets and Association Rules

Slides provided by: **Silviu Maniu**, Presented by: **Pierre-Henri Paris**

September 12th, 2024

M2 Data Science

Table of contents

Market-Basket Model

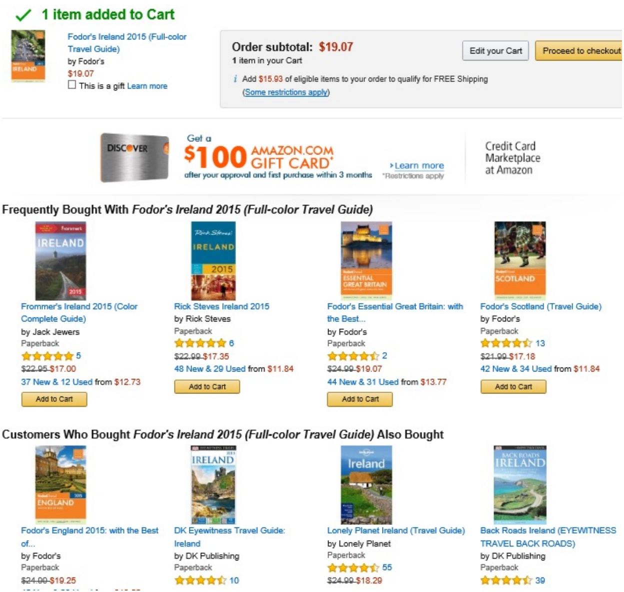
We have a large set of **items** (things sold in shops, markets, supermarkets)

Large set of **baskets** (people buying things all at the same time), each having a *small subset of items*

We have two data mining tasks:

1. we want to find **items that are frequently bought together**
2. we want to find **association rules** (“people who buy X also buy Y”)

Frequent Items in Practice



Association Rules in Practice

Used in supermarket shelf placement

Other Applications

Plagiarism: baskets are sentences, items are documents containing the sentences

- items appearing together too often could be plagiarism

Side-effects in drug combinations: baskets are patients; items are drugs and their side effects

Frequent Itemsets

A set of items that appears in many baskets is said to be **frequent**

Set of items $\mathcal{I}$, itemset $I \in \mathcal{I}$, set of baskets $\mathcal{B}$, basket $B \in \mathcal{B}$

Support of itemset I: number of baskets containing all items in I:

$$\text{supp}(I) = |\{B \mid I \subseteq B\}|$$

Problem: given a **support threshold** s, we call itemset appearing in at least s baskets – or having support s – **frequent itemsets**

Example

Items $\mathcal{I} = \{m, c, p, b, j\}$; baskets $\mathcal{B}$

$$B_1 = \{m, c, b\} \qquad B_5 = \{m, p, b\}$$
$$B_2 = \{m, p, j\} \qquad B_6 = \{m, c, b, j\}$$
$$B_3 = \{m, b\} \qquad B_7 = \{c, b, j\}$$
$$B_4 = \{c, j\} \qquad B_8 = \{b, j\}$$

Support of itemset $I = \{m, b\}$: $\text{supp}(I) = 4$ (appears in B_1, B_3, B_5, B_6)

For a **support threshold** of 3:

- frequent itemsets: $\{m\}$, $\{c\}$, $\{b\}$, $\{j\}$, $\{m, b\}$, $\{b, c\}$, $\{c, j\}$

Association Rules

Association rules – correlations in the contents of baskets

- written as $\{i_1, i_2, \ldots, i_k\} \to j$ – "if a basket contains $\{i_1, i_2, \ldots, i_k\}$ then *it is likely to contain* j also

There can be many rules, we only care about **interesting** ones:

- **confidence** of an association rule:

$$\mathrm{conf}(I \to j) = \frac{\mathrm{supp}(I \cup \{j\})}{\mathrm{supp}(I)}$$

Association Rules

Association rules – correlations in the contents of baskets

- written as $\{i_1, i_2, \ldots, i_k\} \to j$ – "if a basket contains $\{i_1, i_2, \ldots, i_k\}$ then *it is likely to contain* j also

There can be many rules, we only care about interesting ones:

- interest of an association rule:

$$\text{interest}(I \to j) = \text{conf}(I \to j) - \Pr[j] = \text{conf}(I \to j) - \frac{\text{supp}(\{j\})}{|\mathcal{B}|}$$

Example

Items $\mathcal{I} = \{m, c, p, b, j\}$; baskets $\mathcal{B}$

$$B_1 = \{m, c, b\} \qquad B_5 = \{m, p, b\}$$
$$B_2 = \{m, p, j\} \qquad B_6 = \{m, c, b, j\}$$
$$B_3 = \{m, b\} \qquad B_7 = \{c, b, j\}$$
$$B_4 = \{c, j\} \qquad B_8 = \{b, j\}$$

Association rule A: $\{m, b\} \to c$

- confidence $\mathrm{conf}(A) = \frac{\mathrm{supp}(\{m,b,c\})}{\mathrm{supp}(\{m,b\})} = 2/4 = 0.5$
- interest $\mathrm{interest}(A) = \mathrm{conf}(A) - \frac{\mathrm{supp}(\{c\})}{|\mathcal{B}|} = \frac{2}{4} - \frac{4}{8} = 0$ – not very interesting (we want either *high positive values* or *low negative values*)

Mining Association Rules

Problem: find all association rules having support at least s and confidence at least c

- the support of an association rule $I \to j$ is equal to $\text{supp}(I)$
- means that finding the frequent itemsets is the main difficulty: if $I \to j$ has high confidence and support then both I and $I \cup j$ are frequent itemsets!

Mining Association Rules

1. Find all frequent itemsets I
2. Rule generation
   - for every subset $A \subset I$ generate rule $A \to I \backslash A$: since I is frequent A is also frequent, only have to compute the confidence

$$\text{conf}(A \to I \backslash A) = \frac{\text{supp}(I)}{\text{supp}(A)}$$

   - optimization: if ABC $\to$ D is below confidence threshold, then so is AB $\to$ CD
3. **Output** all rules above confidence threshold

Example

Items $\mathcal{I} = \{m, c, p, b, j\}$; baskets $\mathcal{B}$

$$B_1 = \{m, c, b\} \qquad B_5 = \{m, p, b\}$$
$$B_2 = \{m, p, j\} \qquad B_6 = \{m, c, b, j\}$$
$$B_3 = \{m, b\} \qquad B_7 = \{c, b, j\}$$
$$B_4 = \{c, j\} \qquad B_8 = \{b, j\}$$

Support $s = 3$; Confidence $c = 0.75$

Frequent Itemsets:

- $\cancel{\{m\}, \{c\}, \{b\}, \{j\}}$, $\{m, b\}$, $\{b, c\}$, $\{c, j\}$

Rule Generation:

- $m \to b$ $(c = 4/5)$; $\cancel{b \to m}$ $(c = 4/6)$; $\ldots$

Table of contents

Computational Model

We assume that the data is kept in a *disk file*, basket by basket

- also most likely that data **does not fit in main memory**
- cost model: **number of accesses on the disk**

Read data in batches and check subsets in main-memory:

- for pairs of items, this is feasible: $\mathcal{O}(n^2)$ via **nested-loop processing** – dominated by the disk access
- for larger sets, **not feasible** $\mathcal{O}(n^k/k!)$
- **in practice**, frequent items are mostly pairs or triples

In the algorithms we discuss next, we analyze only **the number of passes over the data**

Counting Pairs

Pre-processing: transform item strings into ids (less space used)

Triangular Array - store the counts in an array only for pairs which have $i < j$ (lexicographic order)

- for pair (i, j) update count in $a[k]$ where $k = (i-1)(n-i/2)+j-1$
  – saves half the space

Store triples - store the (i, j, c) triple

- hash table on key i, j containing value c
- saves space when counts are sparse

Monotonicity of Itemsets

Monotonicity of itemsets: if an set of items I is frequent, then so is every subset of I

$$B_1 = \{m, c, b\} \qquad B_5 = \{m, p, b\}$$
$$B_2 = \{m, p, j\} \qquad B_6 = \{m, c, b, j\}$$
$$B_3 = \{m, b\} \qquad B_7 = \{c, b, j\}$$
$$B_4 = \{c, j\} \qquad B_8 = \{b, j\}$$

Monotonicity:

- $\text{supp}(m, c, b) = 2$
- $\text{supp}(m, c) = 2$; $\text{supp}(m, b) = 3$; $\text{supp}(c, b) = 3$
- $\text{supp}(m) = 5$; $\text{supp}(c) = 4$; $\text{supp}(b) = 6$

A-Priori Principles

We can focus on counting pairs – they are the main bottleneck of the frequent items computations

A-Priori algorithm: designed to reduce the number of pairs we need to count, at the expense of making two passes over the data [Agrawal and Srikant, 1994]

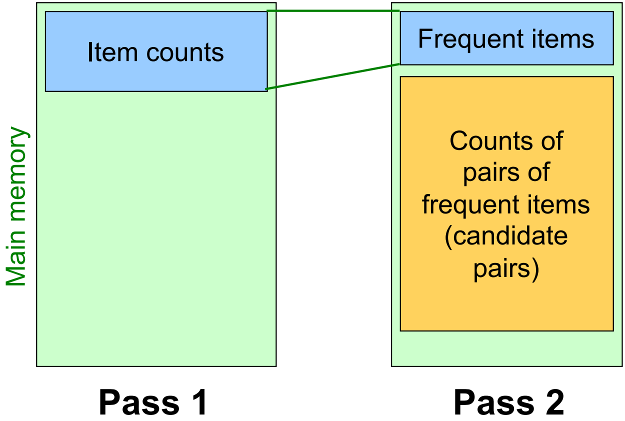
Using monotonicity

- if item i does not have support at least s, then no super-set of i can
- go from singletons, to pairs, to triples, etc.

A-Priori – 2 Passes

1. read baskets and count **support of each item**, keep items having support at least s
2. read baskets again and count *only* the pairs between frequent items
    - memory quadratic only in frequent items, along with a (linear) list of frequent items

A-Priori – 2 Passes



Going Beyond Pairs

For each size of the itemset k, we have two sets of k-tuples:

- C_k candidate tuples which may have support at least s using information from pass $k-1$
- L_k the truly frequent itemsets from C_k

One pass for each k – needs memory space for counts

- in practice, $k = 2$ requires the most memory

Example

Support threshold $s = 2$

$B_1 = \{m, c, b\}$
$B_2 = \{m, p, j\}$
$B_3 = \{m, b\}$
$B_4 = \{m, j\}$

1. $C_1 = \{m\}\ \{c\}\ \{b\}\ \{p\}\ \{j\}$
   - $L_1 = \{m\}\ \{b\}\ \{j\}$
2. $C_2 = \{m, b\}\ \{b, j\}\ \{m, j\}$
   - $L_2 = \{m, b\}\ \{m, j\}$
3. $C_3 = \{m, b, j\}$ (use L_2 and L_1)
   - $L_3 = \emptyset$

Frequent itemsets: $L_1 \cup L_2$

Optimizing A-Priori

Can optimize A-Priori to **use the memory more efficiently** – use hash tables on itemsets to prune sets that can be candidates: **Park-Chen-Yu algorithm** [Park et al., 1995]

Fewer passes over the data:

- **Random sampling**: take only a part of the dataset (enough to fit in memory) and check everything in-memory – have to update the supports
- **SON algorithm**: mine batches of the dataset in-memory; compute the real counts in the second pass – can also be use in MapReduce [Savasere et al., 1995]

Acknowledgments

The contents and some figures taken from Chapter 6 of [Leskovec et al., 2020]. `https://www.mmds.org/`

References i

Agrawal, R. and Srikant, R. (1994).
Fast algorithms for mining association rules in large databases.
In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, page 487–499.

Leskovec, J., Rajaraman, A., and Ullman, J. (2020).
Mining of Massive Datasets.
Cambridge University Press.

Park, J. S., Chen, M.-S., and Yu, P. S. (1995).
An effective hash-based algorithm for mining association rules.
In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, page 175–186.

Savasere, A., Omiecinski, E., and Navathe, S. B. (1995).
An efficient algorithm for mining association rules in large databases.
In *Proceedings of the 21th International Conference on Very Large Data Bases (VLDB)*, page 432–444.