Association Rules Mining

[Mining Massive Datasets]

Prof. [Carlos Castillo]

Topic 13

Universitat Pompeu Fabra Barcelona

Sources

- Data Mining, The Textbook (2015) by Charu Aggarwal (Chapters 4, 5) – slides by Lijun Zhang
- Mining of Massive Datasets 2nd edition (2014) by Leskovec et al. (Chapter 6) - slides
- Data Mining Concepts and Techniques, 3rd edition (2011) by Han et al. (Chapter 6)
- Introduction to Data Mining 2nd edition (2019) by Tan et al. (Chapters 5, 6) – slides ch5, slides ch6

Association rule

- Let X, Y be two itemsets; the rule $X \Rightarrow Y$ is an **association rule** of minimum support **minsup** and minimum confidence **minconf** if:

$$\sup(X \Rightarrow Y) \geq \text{minsup}$$

and

$$\text{conf}(X \Rightarrow Y) \geq \text{minconf}$$

Association rule mining framework

- In the first phase, all the frequent itemsets are generated at the minimum support of minsup
  - The most difficult (computationally expensive) step
- In the second phase, the association rules are generated from the frequent itemsets at the minimum confidence level of minconf
  - Relatively straightforward

A straightforward implementation of the second phase

For each frequent itemset I   // sup(I) ≥ minsup

For each possible partition X, Y = I – X

Check if conf(X⇒Y) ≥ minconf

- Use the **confidence monotonicity property** (next slide) to reduce search space

Confidence monotonicity property

Let X_S, X_L, I be itemsets; assume $X_S \subset X_L \subset I$

Then:

$$\mathrm{conf}(X_L \Rightarrow I - X_L) \geq \mathrm{conf}(X_S \Rightarrow I - X_S)$$

Exercise: prove conf. monotonicity

$$X_S \subset X_L \subset I \Rightarrow \underline{\mathrm{conf}(X_L \Rightarrow I - X_L) \geq \mathrm{conf}(X_S \Rightarrow I - X_S)}$$

Tip: start from what you want to prove:

1. Use the definition of confidence on this

$$\mathrm{conf}(X \Rightarrow Y) = \frac{\sup(X \cup Y)}{\sup(X)}$$

2. Try to arrive to

$$\sup(X_L) \leq \sup(X_S)$$

which we know is true because $X_S \subset X_L$

Answer in
Nearpod Collaborate

Brute-force itemset mining algorithms

Naïve approach

- Generate all candidate itemsets ($2^{|U|}$ of them)
  - Not practical, U=1000 ⇒ more than 10^{300} itemsets
- Calculate sup(I) for every itemset
- Key observation
  - If no k-itemsets are frequent
  - No (k+1)-itemsets are frequent

Improved approach

Start with $k=1$

- Generate all k-itemsets
- Determine $sup(I)$
- If no k-itemset has $sup(I) \geq minsup$, stop
- Otherwise, $k \leftarrow k+1$ and repeat

Improved approach is a significant improvement

- Let l be the final value of k

$$\sum_{i=1}^{l} \binom{|U|}{i} \ll 2^{|U|}$$

- For $|U| = 1000$, $l=10$, this is $\simeq 10^{23}$

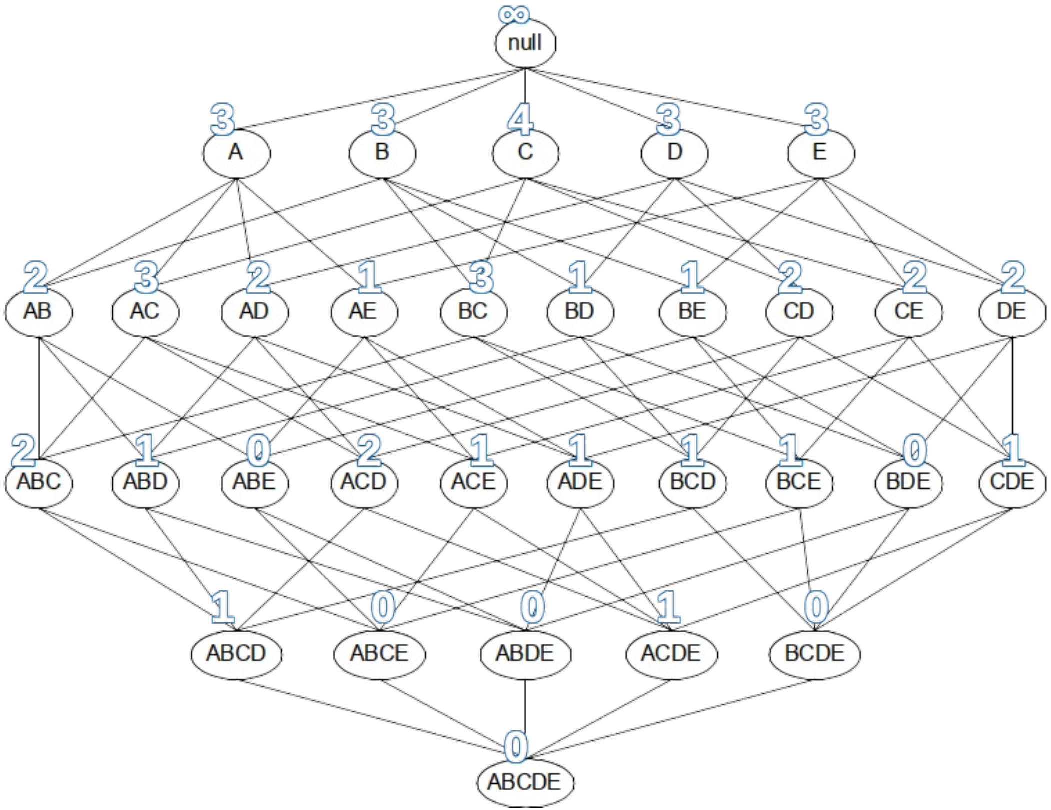
Further improvements to brute-force method

1. Reducing the size of the explored search space (lattice) by pruning candidate itemsets (lattice nodes) using tricks, such as the downward closure property
2. Counting the support of each candidate more efficiently by pruning transactions that are known to be irrelevant for counting a candidate itemset
3. Using compact data structures to represent either candidates or transaction databases that support efficient counting

The Apriori Algorithm

Apriori algorithm principle

- **Downward closure property**: every subset of a frequent itemset is also frequent
- Conversely, if an itemset has a subset that is not frequent, the itemset cannot be frequent
- **What are subsets in the lattice?**



Example

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

Items (1-itemsets)

Item	**Count**	
Bread	4	
Coke	2	X
Milk	4	
Beer	3	
Diaper	4	
Eggs	1	X

Minimum Support = 3

Example

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

Items (1-itemsets)

Item	Count	
Bread	4	
~~Coke~~	~~2~~	X
Milk	4	
Beer	3	
Diaper	4	
~~Eggs~~	~~1~~	X

Minimum Support = 3

Example (cont.)

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

Items (1-itemsets)

Item	Count	
Bread	4	
~~Coke~~	~~2~~	X
Milk	4	
Beer	3	
Diaper	4	
~~Eggs~~	~~1~~	X

Pairs (2-itemsets)

Item	Count	
{Bread, Milk}	3	
{Beer, Bread}	2	X
{Bread, Diaper}	3	
{Beer, Milk}	2	X
{Diaper, Milk}	3	
{Beer, Diaper}	3	

Minimum Support = 3

Example (cont.)

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

Items (1-itemsets)

Item	Count	
Bread	4	
~~Coke~~	~~2~~	X
Milk	4	
Beer	3	
Diaper	4	
~~Eggs~~	~~1~~	X

Pairs (2-itemsets)

Item	Count	
{Bread, Milk}	3	
~~{Beer, Bread}~~	~~2~~	X
{Bread, Diaper}	3	
~~{Beer, Milk}~~	~~2~~	X
{Diaper, Milk}	3	
{Beer, Diaper}	3	

Minimum Support = 3

Example (cont.)

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

Items (1-itemsets)

Item	Count	
Bread	4	
~~Coke~~	~~2~~	X
Milk	4	
Beer	3	
Diaper	4	
~~Eggs~~	~~1~~	X

Pairs (2-itemsets)

Item	Count	
{Bread, Milk}	3	
~~{Beer, Bread}~~	~~2~~	X
{Bread, Diaper}	3	
~~{Beer, Milk}~~	~~2~~	X
{Diaper, Milk}	3	
{Beer, Diaper}	3	

Triplets (3-itemsets)

Item	Count	
{Bread, Diaper, Milk}	2	X
{Beer, Bread, Diaper}	2	X
{Bread, Diaper, Milk}	2	X
{Beer, Bread, Milk}	1	X

Minimum Support = 3

Example (cont.)

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

Items (1-itemsets)

Item	Count	
Bread	4	
~~Coke~~	~~2~~	X
Milk	4	
Beer	3	
Diaper	4	
~~Eggs~~	~~1~~	X

Pairs (2-itemsets)

Item	Count	
{Bread, Milk}	3	
~~{Beer, Bread}~~	~~2~~	X
{Bread, Diaper}	3	
~~{Beer, Milk}~~	~~2~~	X
{Diaper, Milk}	3	
{Beer, Diaper}	3	

Triplets (3-itemsets)

Item	Count	
~~{Bread, Diaper, Milk}~~	~~2~~	X
~~{Beer, Bread, Diaper}~~	~~2~~	X
~~{Bread, Diaper, Milk}~~	~~2~~	X
~~{Beer, Bread, Milk}~~	~~1~~	X

Minimum Support = 3, **found 8 frequent itemsets**

Pseudocode of Apriori

Algorithm $Apriori$(Transactions: $\mathcal{T}$, Minimum Support: $minsup$)
begin
  $k = 1$;
  $\mathcal{F}_1 = \{$ All Frequent 1-itemsets $\}$;
  while $\mathcal{F}_k$ is not empty **do begin**
    Generate $\mathcal{C}_{k+1}$ by joining itemset-pairs in $\mathcal{F}_k$;
    Prune itemsets from $\mathcal{C}_{k+1}$ that violate downward closure;
    Determine $\mathcal{F}_{k+1}$ by support counting on $(\mathcal{C}_{k+1}, \mathcal{T})$ and retaining
        itemsets from $\mathcal{C}_{k+1}$ with support at least $minsup$;
    $k = k + 1$;
  end;
  return$(\cup_{i=1}^{k} \mathcal{F}_i)$;
end

(1) Generation
(2) Pruning
(3) Support counting

Exercise: Apriori

Answer in
Google Spreadsheet

Use the Apriori algorithm to obtain all rules of the form {a,b}→{c} having
minimum support = 2
and
confidence >= 50%

TID	items
T1	I1, I2 , I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

Note: to generate only rules of the form {a,b}→{c}, use only the itemsets of size 3

Summary

Things to remember

- Support and confidence on a rule
- Downward closure property
  - every subset of a frequent itemset is also frequent
  - hence, if an itemset X has a subset that is not frequent, X cannot be frequent
- Apriori algorithm

Exercises for TT13-TT14

- Data Mining, The Textbook (2015) by Charu Aggarwal
  - Exercises 4.9 → 9-10
    [but note the provided solution to these might have a mistake]
- Mining of Massive Datasets 2nd edition (2014) by Leskovec et al.
  - Exercises 6.2.7 → 6.2.5 and 6.2.6
- Introduction to Data Mining 2nd edition (2019) by Tan et al.
  - Exercises 5.10 → 9-12