Association Pattern Mining - Frequent Itemset Mining

Problem

- A dataset of transactions
- One transaction is a set of items
- How to find interesting associations between items ?

Image from https://datacrunch.nl

Problem - Original Text

This paper introduces the problem of "mining" a large collection of basket data type transactions for association rules between sets of items with some minimum specified confidence, and presents an efficient algorithm for this purpose. An example of such an association rule is the statement that 90% of transactions that purchase bread and butter also purchase milk. The antecedent of this rule consists of bread and butter and the consequent consists of milk alone. The number 90% is the confidence factor of the rule.

Mining association rules between sets of items in large databases.
Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. *ACM SIGMOD 1993.*
Citations 9799, Downloads 26k
Last 12 Months 1,6k
Nov 8 2021

Application Domains

if *itemset1* then *itemset2* (degree of certainty)

- Market basket analysis
  - {Bread, Cheese, yogurt} => {milk, eggs}
- Medical diagnosis
  - {Coronavirus} => {fever, cough, shortness of breath}
- Power plant failure prediction [1].
- Text mining, Web log mining, Classification, ...

[1] Maik Reder, Nurseda Y. Yürüşen, Julio J. Melero, Data-driven learning framework for associating weather conditions and wind turbine failures, Reliability Engineering & System Safety, Volume 169, 2018.

Mining Association Rules - Solution Pipeline

Mining Association Rules - Solution Pipeline

Definitions

- The input is a database T that contains a set of n transactions, denoted by $T_1 \ldots T_n$
- Each transaction T_i is a subset drawn on the universe of items U
- Transactions can be represented in a binary format.
- Let U= {Bread, Butter, Cheese, Eggs, Milk, Yogurt}

tid	Set of items	Binary representation
1	{Bread, Butter, Milk}	110010
2	{Eggs, Milk, Yogurt}	000111
3	{Bread, Cheese, Eggs, Milk}	101110
4	{Eggs, Milk, Yogurt}	000111
5	{Cheese, Milk, Yogurt}	001011

Table from Charu C. Aggarwal, Data Mining The Textbook, 2015.

Definitions - Support

- Support of an itemset I is the fraction of the transactions in the database T that contain I as a subset

tid	Set of items	Binary representation
1	{Bread, Butter, Milk}	110010
2	{Eggs, Milk, Yogurt}	000111
3	{Bread, Cheese, Eggs, Milk}	101110
4	{Eggs, Milk, Yogurt}	000111
5	{Cheese, Milk, Yogurt}	001011

- Support({Milk, Yogurt})= 3/5= 0.6

Definitions - Support

how frequently the itemset appears in transaction

- Support of an itemset I is the fraction of the transactions in the database T that contain I as a subset

tid	Set of items	Binary representation
1	{Bread, Butter, Milk}	110010
2	{Eggs, Milk, Yogurt}	000111
3	{Bread, Cheese, Eggs, Milk}	101110
4	{Eggs, Milk, Yogurt}	000111
5	{Cheese, Milk, Yogurt}	001011

- Support({Milk, Yogurt})= 3/5= 0.6

Frequent Itemset Mining

- Given a database T, determine all itemsets I that have support of at least *minsup*

tid	Set of items	Binary representation
1	{Bread, Butter, Milk}	110010
2	{Eggs, Milk, Yogurt}	000111
3	{Bread, Cheese, Eggs, Milk}	101110
4	{Eggs, Milk, Yogurt}	000111
5	{Cheese, Milk, Yogurt}	001011

Find all frequent itemsets with minsup= 0.6

{Milk}, {Eggs}, {Yogurt},

{Milk, Yogurt}, {Milk, Eggs}

Apriori Property

Every subset of a frequent itemset is also frequent

Or

Every superset of a infrequent itemset is also infrequent

tid	Set of items	Binary representation
1	{Bread, Butter, Milk}	110010
2	{Eggs, Milk, Yogurt}	000111
3	{Bread, Cheese, Eggs, Milk}	101110
4	{Eggs, Milk, Yogurt}	000111
5	{Cheese, Milk, Yogurt}	001011

{Milk, Yogurt} is frequent => {Milk} is frequent, {Yogurt} is frequent

{Butter} is infrequent => {Butter, x} is infrequent

Apriori Property

Every subset of a frequent itemset is also frequent

Or

Every superset of a infrequent itemset is also infrequent

Itemset lattice:

Size= $2^{|U|}$

Location of maximal frequent itemsets ?

Any frequent itemset mining Algorithms conceptually maps to a (search) traversal of this lattice.

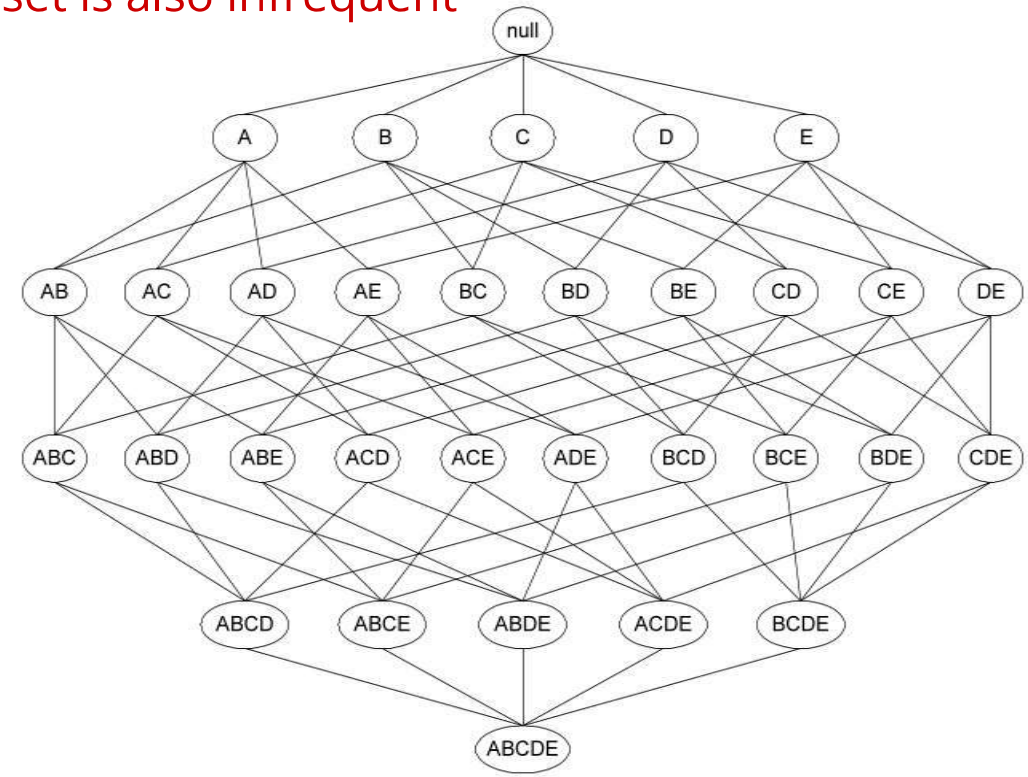


Figure from Han et al., Data Mining: Concepts and Techniques, 2012.

Apriori Property

Every subset of a frequent itemset is also frequent

Or

Every superset of a infrequent itemset is also infrequent

Itemset lattice:

Size= $2^{|U|}$

Location of maximal frequent itemsets ?

Any frequent itemset mining Algorithms conceptually maps to a (search) traversal of this lattice.

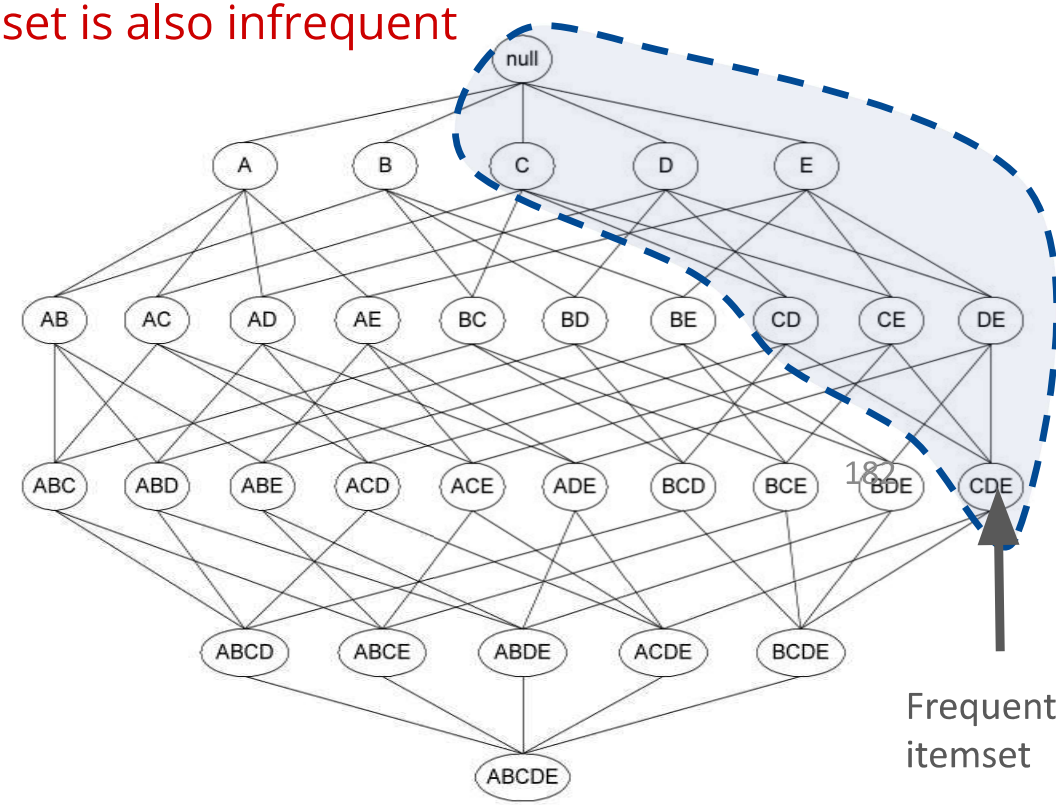


Figure from Han et al., Data Mining: Concepts and Techniques, 2012.

Apriori Property

Every subset of a frequent itemset is also frequent

Or

Every superset of a infrequent itemset is also infrequent

Itemset lattice:

Size= $2^{|U|}$

Location of maximal frequent itemsets ?

Any frequent itemset mining Algorithms conceptually maps to a (search) traversal of this lattice.

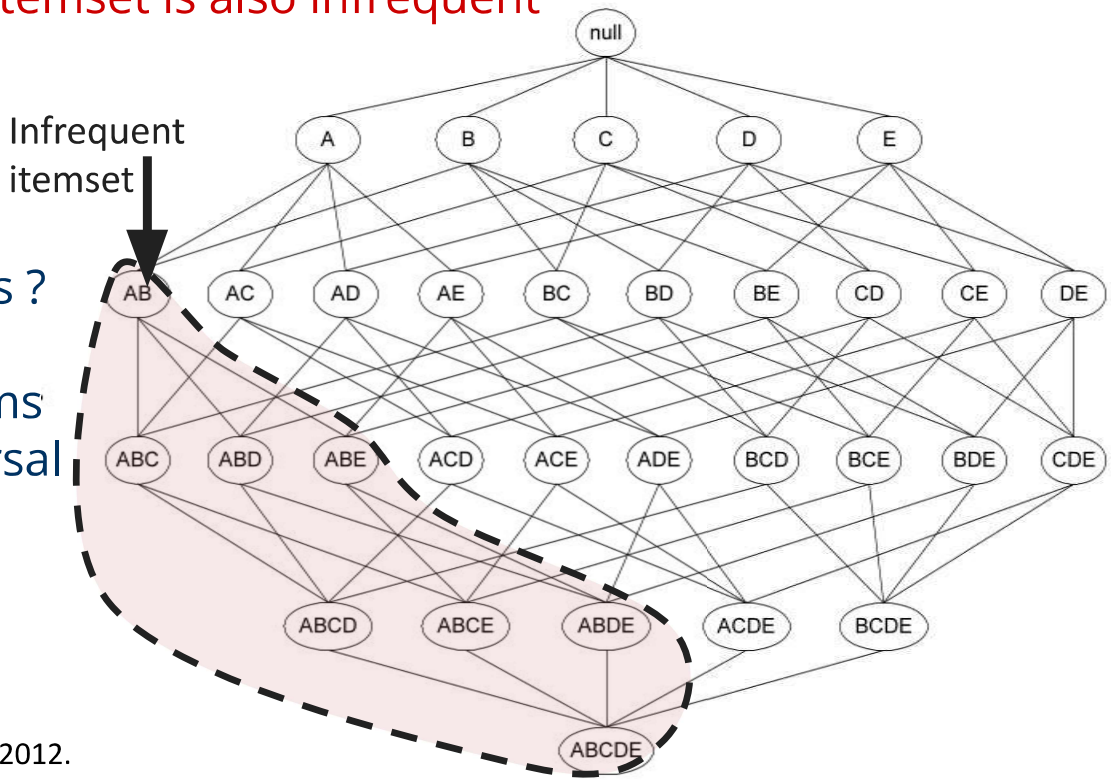


Figure from Han et al., Data Mining: Concepts and Techniques, 2012.

Association Rules

A frequent itemset does not necessarily imply that its items are associated.

tid	Set of items	Binary representation
1	{Bread, Butter, Milk}	110010
2	{Eggs, Milk, Yogurt}	000111
3	{Bread, Cheese, Eggs, Milk}	101110
4	{Eggs, Milk, Yogurt}	000111
5	{Cheese, Milk, Yogurt}	001011

{Milk, Yogurt} is frequent with Sup 0.6, but Milk does not imply Yogurt because 0.4 of the transactions have Milk but no Yogurt.

Association Rules

It indicates the likelihood that
an item Y will be purchased when item X is purchased.

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

tid	Set of items	Binary representation
1	{Bread, Butter, Milk}	110010
2	{Eggs, Milk, Yogurt}	000111
3	{Bread, Cheese, Eggs, Milk}	101110
4	{Eggs, Milk, Yogurt}	000111
5	{Cheese, Milk, Yogurt}	001011

Conf({Eggs, Milk} ⇒ {Yogurt})= Sup({Eggs, Milk,

Yogurt})/ Sup({Eggs, Milk}) = 0.4/0.6= 2/3

Association Rules

(Association Rules) Let X and Y be two sets of items. Then, the rule $X \Rightarrow Y$ is said to be an

association rule at a minsup and a minconf, if:

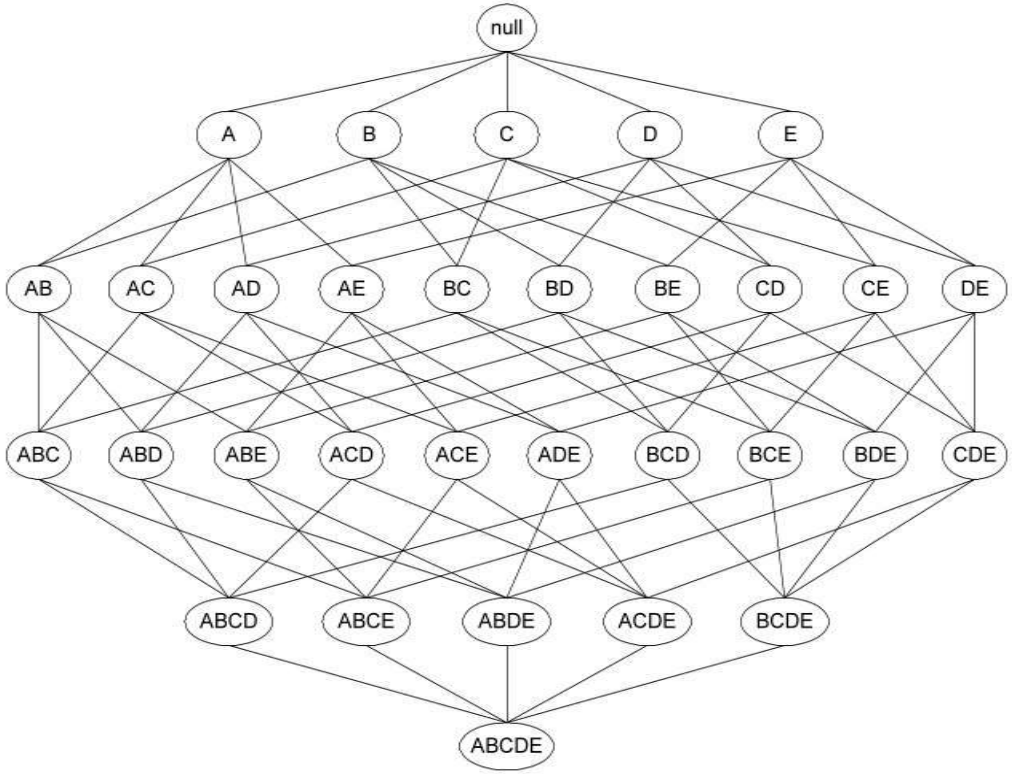
1. $Sup(X \cup Y) >= minsup$, and
2. $Conf(X \Rightarrow Y) >= minconf$.

How to mine for association rules given minsup, minconf ?

1. $F \leftarrow$ all frequent itemsets with sup >= minsup.
2. Using F, generate rules with conf >= minconf.

Mining Frequent Itemsets

- The bruteforce way is to try the whole lattice
- Complexity ? How to optimize ?
  - Pruning candidate itemsets (lattice nodes) using the downward closure property.
  - Counting the support of each candidate more efficiently.
  - Using compact data structures that support efficient counting.



Apriori Algorithm

minimum support count= 2

C - Candidate Itemset
F - Frequent Itemset

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

C1

Items et	#
A	6
B	7
C	6
D	2
E	2

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Items et	#
A	6
B	7
C	6
D	2
E	2

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Items et	#
A	6
B	7
C	6
D	2
E	2

C2

Items et	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

C3

Itemset	#
A, B, C	
A, B, D	
A, B, E	
A, C, D	
A, C, E	
B, C, D	
B, C, E	
B, D, E	

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

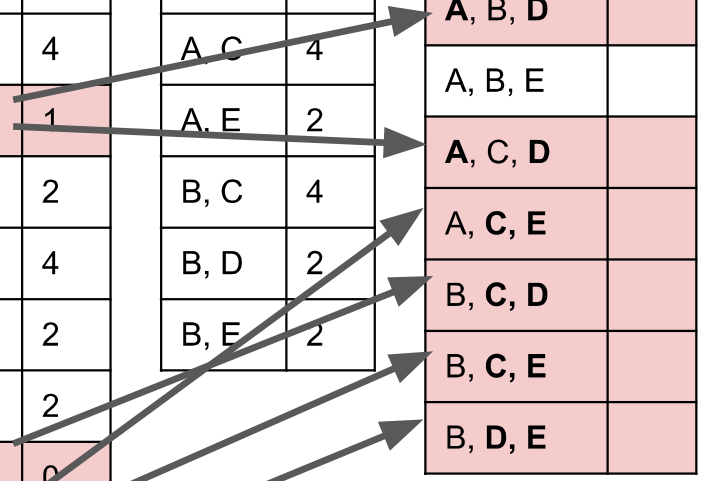
Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

C3

Itemset	#
A, B, C	
A, B, **D**	
A, B, E	
A, C, **D**	
A, **C, E**	
B, **C, D**	
B, **C, E**	
B, **D, E**	



Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

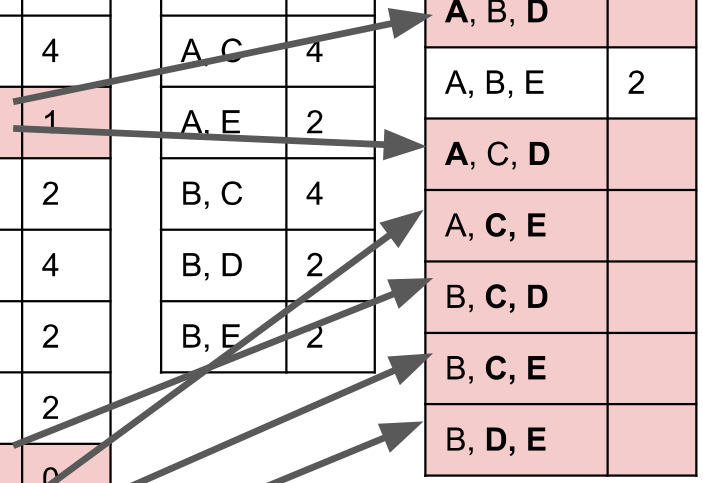
Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

C3

Itemset	#
A, B, C	2
A, B, **D**	
A, B, E	2
A, C, **D**	
A, **C, E**	
B, **C, D**	
B, **C, E**	
B, **D, E**	



Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

F2

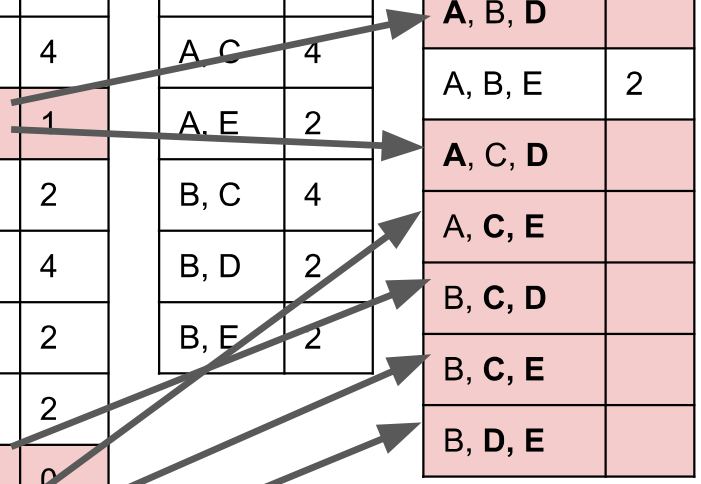
Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

C3

Itemset	#
A, B, C	2
A, B, **D**	
A, B, E	2
A, C, **D**	
A, **C, E**	
B, **C, D**	
B, **C, E**	
B, **D, E**	

F3

Itemset	#
A, B, C	2
A, B, E	2



Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

C3

Itemset	#
A, B, C	2
A, B, **D**	
A, B, E	2
A, C, **D**	
A, **C, E**	
B, **C, D**	
B, **C, E**	
B, **D, E**	

F3

Itemset	#
A, B, C	2
A, B, E	2

C4

Itemset	#
A, B, C, E	

Apriori Algorithm

minimum support count= 2

Database

TID	Items
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

C2

Itemset	#
A, B	4
A, C	4
A, D	1
A, E	2
B, C	4
B, D	2
B, E	2
C, D	0
C, E	1
D, E	0

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

C3

Itemset	#
A, B, C	2
A, B, D	
A, B, E	2
A, C, D	
A, **C, E**	
B, **C, D**	
B, **C, E**	
B, **D, E**	

F3

Itemset	#
A, B, C	2
A, B, E	2

C4

Itemset	#
A, **B, C, E**	

Apriori Algorithm

minimum support count= 2

F1

Itemset	#
A	6
B	7
C	6
D	2
E	2

F2

Itemset	#
A, B	4
A, C	4
A, E	2
B, C	4
B, D	2
B, E	2

F3

Itemset	#
A, B, C	2
A, B, E	2

Apriori Algorithm - Pseudocode

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

Figure from Charu C. Aggarwal, Data Mining The Textbook, 2015.

Limits of Apriori Algorithm

- Uses the apriori property.
  - nice property.
- General-to-specific traversal of the itemset lattice.
  - bi-directional traversal.
- Breadth-first search.
  - depth-first search.
- Generate-and-test strategy.
  - C2 is O(|U| Choose 2)
  - test is count.

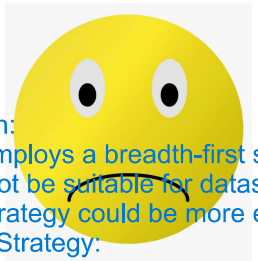
In the worst case, Apriori may generate a large number of candidate itemsets, especially in the early iterations of the algorithm.

In order to calculate support, we need to access database multiple times.

Need for Multiple Passes

Binary Representation:
Apriori often uses binary representation (presence or absence of an item) to represent transactions, which may not capture quantitative information about items.

Apriori Property Dependency:
Limitation: The algorithm relies on the Apriori property, which states that if an itemset is frequent, then all of its subsets must also be frequent. This property can lead to a large number of candidate itemsets being generated and tested, especially in cases where the dataset has a large number of items.
General-to-Specific Traversal:
Limitation: Apriori uses a general-to-specific traversal strategy of the itemset lattice. This means that the algorithm starts with smaller itemsets and gradually explores larger ones. While this approach is efficient in certain cases, it may miss interesting patterns that involve larger itemsets, especially if the support threshold is set too high.



Breadth-First Search:
Limitation: Apriori employs a breadth-first search strategy. While this helps in discovering frequent itemsets efficiently, it might not be suitable for datasets with a skewed distribution of itemset sizes. In such cases, a depth-first search strategy could be more effective.
Generate-and-Test Strategy:
Limitation: The algorithm follows a generate-and-test strategy, where candidate itemsets are generated and then tested for their frequency. This approach can be computationally expensive, especially when dealing with large datasets. Additionally, the size of the candidate itemsets grows rapidly, leading to a potentially large number of combinations to be tested.

Apriori + Tricks

Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- *min_sup*, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
(1)   L1 = find_frequent_1-itemsets(D);
(2)   for (k = 2; Lk−1 ≠ φ; k++) {
(3)       Ck = apriori_gen(Lk−1);
(4)       for each transaction t ∈ D { // scan D for counts
(5)           Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)           for each candidate c ∈ Ct
(7)               c.count++;
(8)       }
(9)       Lk = {c ∈ Ck | c.count ≥ min_sup}
(10)  }
(11)  return L = ∪k Lk;
```

Apriori + Tricks

```
procedure apriori_gen(L_{k-1}:frequent (k-1)-itemsets)
(1)     for each itemset l_1 ∈ L_{k-1}
(2)         for each itemset l_2 ∈ L_{k-1}
(3)             if (l_1[1] = l_2[1]) ∧ (l_1[2] = l_2[2])
                    ∧... ∧ (l_1[k-2] = l_2[k-2]) ∧ (l_1[k-1] < l_2[k-1]) then {
(4)                 c = l_1 ⋈ l_2; // join step: generate candidates
(5)                 if has_infrequent_subset(c, L_{k-1}) then
(6)                     delete c; // prune step: remove unfruitful candidate
(7)                 else add c to C_k;
(8)             }
(9)     return C_k;

procedure has_infrequent_subset(c: candidate k-itemset;
            L_{k-1}: frequent (k-1)-itemsets); // use prior knowledge
(1)     for each (k-1)-subset s of c
(2)         if s ∉ L_{k-1} then
(3)             return TRUE;
(4)     return FALSE;
```

More Apriori tricks

- Transaction reduction: A transaction that does not include any k-frequent itemsets, cannot contain (k+1)-frequent itemsets, and can thus be removed to make the counting faster.
- Partitioning: divide the database D into a set of disjoint partitions. Find all frequent itemsets in every partition (note that minsup remains the same). A local frequent itemset may not be frequent for the whole D, but a frequent itemset in D must be frequent in at least one partition. The improvement of partitioning is that it allows for parallel processing, and that the partition size can be designed to fit in memory.
- Sampling: perform Apriori on a random sample (say 10%). Rule accuracy is not guaranteed, but efficiency is improved. Certain applications can tolerate this.

Mining Association Rules: Step2

tid	Set of items	Binary representation
1	$\{Bread, Butter, Milk\}$	110010
2	$\{Eggs, Milk, Yogurt\}$	000111
3	$\{Bread, Cheese, Eggs, Milk\}$	101110
4	$\{Eggs, Milk, Yogurt\}$	000111
5	$\{Cheese, Milk, Yogurt\}$	001011

- Minsup= 0.4, minconf= 0.8.
- Frequent itemsets: {Bread, Milk}, {Milk, Cheese}, {Eggs, Milk}, {Eggs, Yogurt}, {Eggs, Milk, Yogurt}.
  - Conf(Bread □ Milk)= 0.4/0.4= 1
  - Conf(Milk □Bread)= 0.4/1= 0.4
  - …
  - Conf({Eggs, Milk} □ Yogurt)= 0.4/0.6= 2/3

Mining Association Rules: Step2

- Once the frequent itemsets have been found, foreach of them (let be called *l*):
  - Generate all nonempty subsets of *l*.
  - For every subset *s* of *l*, Output the rule *s -> (l – s)* if its confidence >= minconf.

Recommended Readings

- Aggarwal, Charu C. Data Mining: The Textbook. Springer (2015).
- Han, Jiawei and Kamber, Micheline and Pei, Jian: Data Mining: Concepts and Techniques (3rd. ed.). Morgan Kaufmann Publishers Inc. (2011).