

# Association Rule Mining: Apriori

Yufei Tao

Department of Computer Science and Engineering  
Chinese University of Hong Kong

This lecture we will discuss another fundamental problem in data mining called **association rule mining**.

Let  $U$  be a set of items, referred to as the **universal set**. The details of those items are irrelevant to our problem definition.

We define an **itemset**, denoted as  $I$ , to be a subset of  $U$ . If  $|I| = k$ , then we refer to  $I$  as a  **$k$ -itemset**.

The dataset of the association rule mining problem is a set  $S$  of itemsets. We refer to each of those itemsets as a **transaction**, and denote it by  $T$ .

The **support** of an itemset  $I$  is the number of transactions in  $S$  that contain  $I$ , namely:

$$\text{support}(I) = |\{T \in S \mid I \subseteq T\}|$$

## Example

- $U = \{\text{beer, bread, butter, milk, potato, onion}\}$
- The following table shows a dataset of 5 transactions:

id	items
1	beer, bread
2	beer, butter
3	butter, milk, potato
4	beer, bread, butter, milk, onion
5	beer, bread, butter, milk
6	beer, bread, milk, onion

- If  $I = \{\text{beer, bread}\}$ , then  $\text{support}(I) = 4$ .

An **association rule**  $R$  has the form

$$l_1 \rightarrow l_2$$

where both  $l_1$  and  $l_2$  are non-empty itemsets satisfying  $l_1 \cap l_2 = \emptyset$ .

The **support** of  $R$ , denoted as  $sup(R)$ , equals the support of the itemset  $l_1 \cup l_2$ .

The **confidence** of  $R$  equals

$$conf(R) = \frac{support(l_1 \cup l_2)}{support(l_1)}.$$

## Example

id	items
1	beer, bread
2	beer, butter
3	butter, milk, potato
4	beer, bread, butter, milk, onion
5	beer, bread, butter, milk
6	beer, bread, milk, onion

- The rule “ $\{beer\} \rightarrow \{bread\}$ ” has support 4 and confidence  $4/5$ .
- “ $\{beer\} \rightarrow \{milk\}$ ” has support 3 and confidence  $3/5$ .
- “ $\{butter, potato\} \rightarrow \{milk\}$ ” has support 1 and confidence 1.

## Problem (Association Rule Mining)

Given (i) a set  $S$  of transactions, and (ii) two constants *minsup* and *minconf*, we want to find **all** the association rules  $R$  such that

$$\begin{aligned} \text{sup}(R) &\geq \text{minsup} \\ \text{conf}(R) &\geq \text{minconf}. \end{aligned}$$

Think:

- Why does it make sense to find such association rules?
- Why purposes do *minsup* and *minconf* serve?

Next, we will discuss how to solve the association rule problem. As a naive solution, we could first enumerate all the possible association rules, calculate their support and confidence values, and then output the qualifying ones. However, the method is typically prohibitively slow due to the large number of possible rules.

Next, we describe an algorithm called **Apriori**.



Let  $I$  be an itemset. We say that  $I$  is **frequent** if  $\text{support}(I) \geq \text{minsup}$ .

### Example

id	items
1	beer, bread
2	beer, butter
3	butter, milk, potato
4	beer, bread, butter, milk, onion
5	beer, bread, butter, milk
6	beer, bread, milk, onion

Assume that  $\text{minsup} = 3$ . Then:

- $\{\text{beer}\}$ ,  $\{\text{beer, bread}\}$ , and  $\{\text{beer, bread, milk}\}$  are all frequent itemsets.
- $\{\text{potato}\}$ ,  $\{\text{potato, onion}\}$ , and  $\{\text{beer, milk, onion}\}$  are not frequent itemsets.

If  $I_1 \rightarrow I_2$  is an association rule that should be reported, by definition, it must hold that the itemset  $I_1 \cup I_2$  is frequent.

Motivated by this observation, Apriori runs in two steps:

- 1 (Frequent itemsets computation): Report all the frequent itemsets of  $U$ .
- 2 (Rule generation): Generate association rules from the above frequent itemsets.

Next, we will explain each step in turn.

The next lemma is straightforward:

### Lemma

$$\text{support}(I_1 \cup I_2) \leq \text{support}(I_1).$$

The above is known as the **anti-monotone property**.

### Corollary

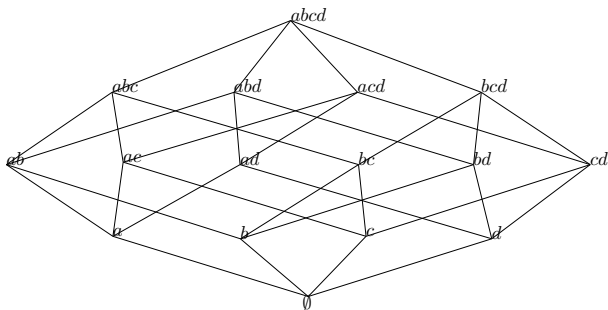
Suppose that  $I_1 \subseteq I_2$ .

- If  $I_2$  is frequent, then  $I_1$  must be frequent.
- If  $I_1$  is not frequent, then  $I_2$  cannot be frequent.

For example, if  $\{beer, bread\}$  is frequent, then so must be  $\{beer\}$  and  $\{bread\}$ . Conversely, if  $\{beer\}$  is not frequent, then neither is  $\{beer, bread\}$ .

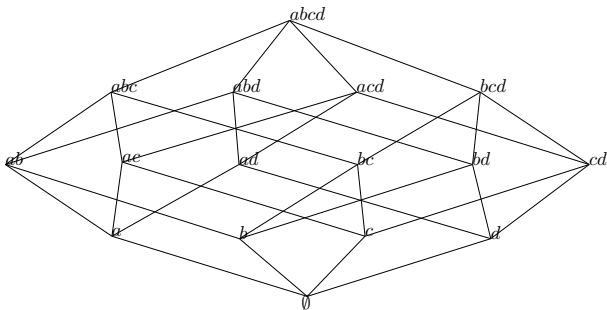
If the universal set  $U$  has  $n$  items, then there are  $2^n - 1$  non-empty itemsets. It is helpful to think of these itemsets in the form of a **lattice** that captures the containment relation among these itemsets.

The figure below shows a lattice for  $n = 4$  (assuming  $U = \{a, b, c, d\}$ ). Note that an itemset  $I_1$  is connected to an itemset  $I_2$  of the upper level if and only if  $I_1 \subset I_2$ .



If we are unlucky, we may have to examine all the itemsets in the lattice. Fortunately, in reality, Corollary 6 implies a pruning rule for us to eliminate itemsets.

For example, if we already know that  $\{a\}$  is infrequent, then we can immediately declare that **all** of  $\{ab\}$ ,  $\{ac\}$ ,  $\{ad\}$ ,  $\{abc\}$ ,  $\{abd\}$ ,  $\{acd\}$ , and  $\{abcd\}$  are infrequent.



Given an integer  $k \in [1, n]$ , let  $F_k$  denote the set of all frequent  $k$ -itemsets (i.e., itemsets of size  $k$ ). Then, the entire set of frequent itemsets equals

$$F_1 \cup F_2 \cup \dots \cup F_n.$$

Our earlier discussion indicates that, if  $F_i = \emptyset$ , then  $F_k$  is also empty for any  $k > i$ .

Therefore, the Apriori algorithm adopts the following approach to find all the frequent itemsets:

- 1  $k = 1$
- 2 Find  $F_k$ . If  $F_k = \emptyset$ , terminate.
- 3  $k \leftarrow k + 1$ ; go to Line 2.

Next, we will clarify the details of Line 2.

## Finding $F_1$ .

This is fairly easy. Suppose that  $U$  has  $n$  items. Then, there are only  $n$  **candidate 1-itemsets**; let  $C_1$  be the set of all these candidate itemsets. For each of them, calculate its support, and report the frequent ones.

### Example

$U = \{\text{beer, bread, butter, milk, potato, onion}\}$

$\text{minsup} = 3$

id	items
1	beer, bread
2	beer, butter
3	butter, milk, potato
4	beer, bread, butter, milk, onion
5	beer, bread, butter, milk
6	beer, bread, milk, onion

- $C_1 = \{\{\text{beer}\}, \{\text{bread}\}, \{\text{butter}\}, \{\text{milk}\}, \{\text{potato}\}, \{\text{onion}\}\}$ .
- $F_1 = \{\{\text{beer}\}, \{\text{bread}\}, \{\text{butter}\}, \{\text{milk}\}\}$ .

## Finding $F_k$ ( $k > 1$ ).

The main strategy is to identify a **candidate set**  $C_k$  of  $k$ -itemsets. Then, we can calculate the support of each such  $k$ -itemset, and report the frequent ones.

The key is to limit the size of  $C_k$ . Naively, we may set  $C_k$  to include all the  $k$ -itemsets, the number of which, however, is  $\binom{n}{k}$ . Even when  $k$  is moderately large, this is a huge number such that it would be prohibitively expensive to compute the supports of all of them.

Next, we will discuss another method that generates a  $C_k$  whose size is usually much smaller.



First, impose an arbitrary total order on the items of  $U$  (e.g., the alphabetic order). Let  $I = \{a_1, a_2, \dots, a_k\}$  be a frequent  $k$ -itemset (i.e., an itemset in  $F_k$ ). The lemma below is a straightforward corollary of Corollary 6:

### Lemma

$\{a_1, a_2, \dots, a_{k-2}, a_{k-1}\}$  and  $\{a_1, a_2, \dots, a_{k-2}, a_k\}$  are both frequent  $(k-1)$ -itemsets, namely, both of them need to be in  $F_{k-1}$ .

Next, given a  $(k-1)$ -itemset  $I = \{b_1, b_2, \dots, b_{k-2}, b_{k-1}\}$ , we refer to the sequence  $(b_1, b_2, \dots, b_{k-2})$  as the **prefix** of  $I$ . Note that the prefix includes only the first  $k-2$  items.

Motivated by this, Apriori generates  $C_k$  from  $F_{k-1}$  as follows.

- 1 Sort the itemsets in  $F_{k-1}$  by prefix. We will refer to the set of itemsets with the same prefix as a **group**.
- 2 Process each group as follows. For each pair of different itemsets  $\{a_1, a_2, \dots, a_{k-2}, a_{k-1}\}$  and  $\{a_1, a_2, \dots, a_{k-2}, a_k\}$  in the group, add to  $C_k$  the itemset  $\{a_1, a_2, \dots, a_k\}$ .

## Example

$U = \{\text{beer, bread, butter, milk, potato, onion}\}$   
 $\text{minsup} = 3$

id	items
1	beer, bread
2	beer, butter
3	butter, milk, potato
4	beer, bread, butter, milk, onion
5	beer, bread, butter, milk
6	beer, bread, milk, onion

- We know earlier  $F_1 = \{\{\text{beer}\}, \{\text{bread}\}, \{\text{butter}\}, \{\text{milk}\}\}$ .
- Hence,  $C_2 = \{\{\text{beer, bread}\}, \{\text{beer, butter}\}, \{\text{beer, milk}\}, \{\text{bread, butter}\}, \{\text{bread, milk}\}, \{\text{butter, milk}\}\}$ .
- Hence,  $F_2 = \{\{\text{beer, bread}\}, \{\text{beer, butter}\}, \{\text{beer, milk}\}, \{\text{bread, milk}\}, \{\text{butter, milk}\}\}$ .

## Example

$U = \{\text{beer, bread, butter, milk, potato, onion}\}$   
 $\text{minsup} = 3$

id	items
1	beer, bread
2	beer, butter
3	butter, milk, potato
4	beer, bread, butter, milk, onion
5	beer, bread, butter, milk
6	beer, bread, milk, onion

- We know earlier  $F_2 = \{\{\text{beer, bread}\}, \{\text{beer, butter}\}, \{\text{beer, milk}\}, \{\text{bread, milk}\}, \{\text{butter, milk}\}\}$ .
- Hence,  $C_3 = \{\{\text{beer, bread, butter}\}, \{\text{beer, bread, milk}\}, \{\text{beer, butter, milk}\}\}$ .
- Hence,  $F_3 = \{\{\text{beer, bread, milk}\}\}$ .
- $C_4 = \emptyset$ . Therefore,  $F_4 = \emptyset$ .

Recall that Apriori runs in two steps:

- ① (**Frequent itemsets computation**): Report all the frequent itemsets of  $U$ .
- ② (**Rule generation**): Generate association rules from the above frequent itemsets.

Next, we will explain the second step.

Let  $I$  be a frequent itemset with size  $k \geq 2$ . We first generate **candidate association rules** from  $I$  as follows. Divide  $I$  into **disjoint** non-empty itemsets  $I_1, I_2$ , namely,  $I_1 \cup I_2 = I$  while  $I_1 \cap I_2 = \emptyset$ . Then,  $I_1 \rightarrow I_2$  is taken as a candidate association rule.

As a second step, we compute the confidence values of all such candidate rules, and report those whose confidence values exceed *minconf*.

Note:

- $support(I_1 \rightarrow I_2)$  must be at least *minsup* (why?).
- To calculate the confidence of  $I_1 \rightarrow I_2$ , we need  $support(I)$  and  $support(I_1)$ . Both values are **directly** available from the first step of Apriori (finding frequent itemsets), noticing that  $I_1$  must be a frequent itemset.
- If  $I$  and  $I'$  are two frequent itemsets, no candidate rule generated from  $I$  can be identical to any candidate rule generated from  $I'$  (why?).

A drawback of the above method is that when  $k$  is large, it is quite expensive to compute the confidence values of  $2^k - 2$  association rules. Next, we present a heuristic that can often reduce the number in practice.

As before, fix a frequent  $k$ -itemset  $I$ . Let  $I_1, I_2$  be disjoint non-empty subsets of  $I$  with  $I_1 \cup I_2 = I$ . Similarly, let  $I'_1, I'_2$  also be disjoint non-empty subsets of  $I$  with  $I'_1 \cup I'_2 = I$ . We have:

### Lemma

If  $I_1 \subset I'_1$ , then  $\text{conf}(I_1 \rightarrow I_2) \leq \text{conf}(I'_1 \rightarrow I'_2)$ .

We say that  $I'_1 \rightarrow I'_2$  **left-contains**  $I_1 \rightarrow I_2$ .

### Proof.

$$\text{conf}(I_1 \rightarrow I_2) = \frac{\text{support}(I)}{\text{support}(I_1)} \leq \frac{\text{support}(I)}{\text{support}(I'_1)} = \text{conf}(I'_1 \rightarrow I'_2).$$

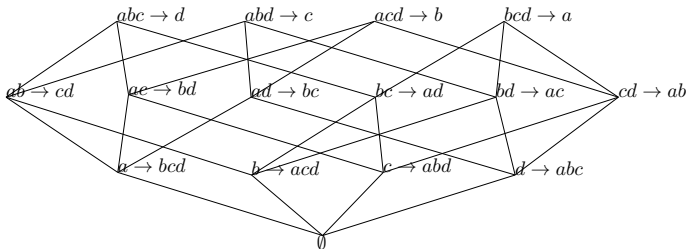


### Example

Suppose that  $I = \{\{beer, bread, milk\}\}$ . It must hold that  $\text{conf}(\{beer, bread\} \rightarrow \{milk\}) \geq \text{conf}(\{beer\} \rightarrow \{milk, bread\})$ .



We can organize all the candidate association rules generated from  $I$  in a lattice. The following figure illustrates the lattice for  $I = \{abcd\}$ . Note that a rule  $R_1$  is connected to another rule  $R_2$  of the upper level if and only if  $R_2$  left-contains  $R_1$ .



Apriori computes the confidence values of the candidate rules by examining them in the **top-down** order from the lattice.

Think: if the confidence value of  $abc \rightarrow d$  is below  $minconf$ , what other candidate rules can be pruned?