Association Rules

- An important class of regularities in data
- Mining of association rules is a fundamental data mining task.
- Objective is to find all co-occurrence relationships, called associations, among data items.
- Supermarket - how items are purchased by customers
- Cheese $\rightarrow$ Beer [support = 10%, confidence = 80%].
- The rule says that 10% customers buy Cheese and Beer together, and those who buy Cheese also buy Beer 80% of the time.
- Example: 5% of customers buy bed first, then mattress and then pillows
- Sequence or pattern in which the items are purchased is important.
- Support and confidence are two measures of rule strength.

- $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items
- $T = (t_1, t_2, \ldots, t_n)$ be a set of transactions
- Each transaction t_i is a set of items such that $t_i \subseteq I$.
- $X \to Y$, where $X \subset I, Y \subset I$, and $X \cap Y = \varnothing$
- X or Y a set of items (itemset)
- A transaction is simply a set of items purchased by a customer.
- Transaction $t_i \in T$ is said to contain an itemset X; $X \subset t_i$
- Support count of X in T denoted by $X.count$
- The strenth of a rule is measured by its support and confidence.
- The percentage of transactions in T that contains $X \cup Y$, and can be seen as an estimate of the probability, $P(X \cup Y)$.

- The rule support thus determines how frequent the rule is applicable in the transaction set T.
- n be the number of transactions in T.
- The support of the rule $X \rightarrow Y$ is computed as follows:

$$\text{support} = \frac{(X \cup Y).count}{n}$$

- The confidence of a rule, $X \rightarrow Y$, is the percentage of transactions in T that contain X also contain Y.
- It can be seen as an estimate of the conditional probability, $P(Y|X)$.

$$\text{confidence} = \frac{(X \cup Y).count}{X.count}$$

- Confidence determines the predictability of the rule.
- If the confidence of a rule is too low, one cannot reliably infer or predict Y from X.
- A rule with low predictability is of limited use.
- Given a transaction set T, the problem of mining association rules is to discover all association rules in T that have support and confidence greater than or equal to the user-specified minimum support and minimum confidence.
- A large number of association rule mining algorithms - different mining efficiencies.
- The same set of rules although their computational efficiencies and memory requirements may be different.

Apriori Algorithm

- The best known mining algorithm is the Apriori algorithm.
- The Apriori algorithm works in two steps:
  1. Generate all frequent itemsets: A frequent itemset is an itemset that has transaction support above $minsup$.
  2. Generate all confident association rules from the frequent itemsets: A confident association rule is a rule with confidence above $minconf$.
- Frequent Itemset Generation: It relies on the apriori or downward closure property to efficiently generate all frequent itemsets by making multiple passes over the data.
- Downward Closure Property: If an itemset has minimum support, then every non-empty subset of this itemset also has minimum support.

- The idea is simple because if a transaction contains a set of items X, then it must contain any non-empty subset of X.
- This property and the $minsup$ threshold prune a large number of itemsets that cannot be frequent.
- The number of items in an itemset its size, and an itemset of size k a k-itemset.
- To ensure efficient itemset generation, the algorithm assumes that the items in I are sorted in lexicographic order (a total order).
- $\{w[1], w[2], \ldots, w[k]\}$ to represent a k-itemset w consisting of items $w[1], w[2], \ldots, w[k]$, where $w[1] < w[2] < \cdots < w[k]$ according to the total order.

Generating frequent itemsets

$C_1 \leftarrow$ init-pass(T); counts the supports of individual items

$F_1 \leftarrow \{f | f \in C_1, f.count/n \geq minsup\}$; determines whether each of them is frequent

for $(k = 2; F_{k-1} \neq \varnothing; k++)$ do

    $C_k \leftarrow$ candidate-gen(F_{k-1});

    for each transaction $t \in T$ do

        for each candidate $c \in C_k$ do

            if c is contained in t then

            $c.count^{++}$;

        endfor

    endfor

    $F_k \leftarrow \{c \in C_k | c.count/n \geq minsup\}$

endfor

return $F \leftarrow \bigcup_k F_k$;

Function candidate-gen(F_{k-1})

$C_k \leftarrow \varnothing$;
for all $f_1, f_2 \in F_{k-1}$
with $f_1 = \{i_1, \ldots, i_{k-2}, i_{k-1}\}$
and $f_2 = \{i_1, \ldots, i_{k-2}, i'_{k-1}\}$
and $i_{k-1} < i'_{k-1}$ do
    $c \leftarrow \{i_1, \ldots, i_{k-2}, i_{k-1}, i'_{k-1}\}$;
    $C_k \leftarrow C_k \cup \{c\}$;
    for each $(k-1)$-subset s of c do
        if $(s \notin F_{k-1})$ then
        delete c from C_k;
    endfor
endfor
return C_k;

- t1: Beef, Chicken, Milk
- t2: Beef, Cheese
- t3: Cheese, Boots
- t4: Beef, Chicken, Cheese
- t5: Beef, Chicken, Clothes, Cheese, Milk
- t6: Chicken, Clothes, Milk
- t7: Chicken, Milk, Clothes
- $minsup = 30\%$ and $minconf = 80\%$
- Chicken, Clothes $\rightarrow$ Milk [sup = 3/7, conf = 3/3]
- support 42.84% and confidence 100%
- Clothes $\rightarrow$ Milk, Chicken [sup = 3/7, conf = 3/3]

- F1: {{Beef}:4, {Cheese}:4, {Chicken}:5, {Clothes}:3, {Milk}:4}}
- C2: {{Beef, Cheese}, {Beef, Chicken}, {Beef, Clothes}, {Beef, Milk}, {Cheese, Chicken}, {Cheese, Clothes}, {Cheese, Milk}, {Chicken, Clothes}, {Chicken, Milk}, {Clothes, Milk}}
- F2: {{Beef, Chicken}:3, {Beef, Cheese}:3, {Chicken, Clothes}:3, {Chicken, Milk}:4, {Clothes, Milk}:3}
- C3: {{Chicken, Clothes, Milk}}
- F3: {{Chicken, Clothes, Milk}:3}
- {Beef, Cheese, Chicken} is also produced but {Cheese, Chicken} is not in F2, and so the itemset {Beef, Cheese, Chicken} is not included in C3.

- Theoretically, this is an exponential algorithm.
- Interestingness problem: produces a huge number of itemsets (and rules), tens of thousands, or more, very hard to analyze them to find those useful ones.
- Association Rule Generation: use frequent itemsets to generate all association rules
- To generate rules for every frequent itemset f, all nonempty subsets of f are used
- For each such subset α, a rule of the form
- $(f - \alpha) \rightarrow \alpha$, if

$$\text{confidence} \quad = \frac{f.count}{(f-\alpha).count} \geq minconf$$