Association Rules in Data Mining I

- Association Rules show relationships (not inherent in the data) between data items
- Example:
  purchase product A → purchase product B
- Different from functional dependencies
- Association Rules do not represent causality or correlation
- Association Rules detect common usage of items
- Database = set of transactions, each involving items

Association Rules in Data Mining II

- Association rules are frequently used with the **market-basket data** model.
  - A market basket corresponds to the sets of items a consumer purchases during one visit to a supermarket.
- The set of items purchased by customers is known as an **itemset**.
- An **association rule** AR is of the form X=>Y, where $X = \{x_1, x_2, \ldots, x_n\}$, and $Y = \{y_1, y_2, \ldots, y_m\}$ are sets of items, with x_i and y_j being distinct items for all i and all j.
- This AR (also noted LHS => RHS) states that:
  if the customer buys X, they are also **likely** to buy Y.
- The **itemset** of this AR is LHS **U** RHS
  - Interesting association rules are measured by their **support** and **confidence**.

Association Rules Confidence and Support

- **Denote an AR by LHS => RHS**
- **Support**:
  - The **support** of an AR is the percentage of transactions that contain all of the items in the itemset, LHS **U** RHS.
  - It refers to how frequently the specific itemset LHS **U** RHS occurs in the database.
  - If the support is low, this means that there is no overwhelming evidence that items in LHS **U** RHS occur together, i.e LHS => RHS is not a plausible AR
- **Confidence**:
  - The **confidence** of an AR is the conditional probability that the items of RHS will be purchased when the items of the LHS are purchased.
  - It refers to how strong is the implication LHS => RHS
  - Confidence is computed as

    support(LHS **U** RHS) / support(LHS)

    of all transactions containing LHS, how many contain RHS

Example

Transaction_id	Time	Items_bought
101	6:35	milk, bread, cookies, juice
792	7:38	milk, juice
1130	8:05	milk, eggs
1735	8:40	bread, cookies, coffee

- Consider the two ARs: milk => juice & bread => juice
- Their supports are: 50% and 25% resp.
- Their confidences are: 66.7% and 50% resp.

Goal of mining ARs:

generate ARs that exceed some user-specified support and confidence thresholds

Generating Association Rules

- A general 2-step algorithm for generating ARs:

  1) Generate all itemsets that have a support exceeding the given threshold.
     Itemsets with this property are called
     large or **frequent itemsets**.
  2) Generate rules for each large itemset as follows:
     1) For a large itemset X and Y a subset of X, let Z = X – Y
     2) If support(X)/Support(Z) > minimum confidence,
        then the rule Z=>Y (i.e. X-Y=>Y) is a valid rule.

Association Rule Complexity

- Generating rules from large itemsets is "easy"
- Discovering large itemsets is "hard"
  (m items, 2^m itemsets, binomial theorem, exponentiality)
- Two properties are used to reduce the combinatorial search space for AR generation.
  - **Downward Closure**
    - A subset of a large itemset must also be large
      (i.e. subsets of large itemsets exceed minimum support)
  - **Anti-monotonicity**
    - A superset of a small itemset is also small.
      (i.e. the itemset does not have sufficient support to be considered for rule generation, extensions of small itemsets are small)

Generating Association Rules: The Apriori Algorithm

- The **Apriori algorithm** was the first algorithm used to generate association rules.
- The **Apriori algorithm** uses the general 2-step algorithm for creating association rules together with downward closure and anti-monotonicity.

Input: Database of m transactions, D, and a minimum support, *mins*, represented as a fraction of m.

Output: Frequent itemsets, $L_1, L_2, \ldots, L_k$

Begin /* steps or statements are numbered for better readability */

1. Compute support(i_j) = count(i_j)/m for each individual item, $i_1, i_2, \ldots, i_n$ by scanning the database once and counting the number of transactions that item i_j appears in (that is, count(i_j));
2. The candidate frequent 1-itemset, C_1, will be the set of items $i_1, i_2, \ldots, i_n$.
3. The subset of items containing i_j from C_1 where support(i_j) >= mins becomes the frequent

   1-itemset, L_1;
4. $k = 1$;

   termination = false;

   repeat

1. $L_{k+1} =$;
2. Create the candidate frequent (k+1)-itemset, C_{k+1}, by combining members of L_k that have k–1 items in common; (this forms candidate frequent (k+1)-itemsets by selectively extending frequent k-itemsets by one item)
3. In addition, only consider as elements of C_{k+1} those k+1 items such that every subset of size k appears in L_k;
4. Scan the database once and compute the support for each member of C_{k+1}; if the support for a member of C_{k+1} >= mins then add that member to L_{k+1};
5. If L_{k+1} is empty then termination = true

   else $k = k + 1$;

   until termination;

End;

Transaction_id	Time	Items_bought
101	6:35	milk, bread, cookies, juice
792	7:38	milk, juice
1130	8:05	milk, eggs
1735	8:40	bread, cookies, coffee

run the APRIORI algorithm:

mins = 0.5 $m = 4$ $n = 6$

$C_1 = \{$milk, bread, juice, cookies, eggs, coffee$\}$ supports
0.75, 0.5, 0.5, 0.5, 0.25, 0.25

m, b, j, c qualify for L_1 (supports $\geq$ mins)
(c $\mapsto$ cookies)

1st iteration of repeat loop:
extend freq. 1-itemsets to create candidate freq. 2-itemsets

$C_2 = \{(m,b), (m,j), (b,j), (m,c), (b,c), (j,c)\}$ supports
0.25, 0.5, 0.25, 0.25, 0.5, 0.25

(m,j), (b,c) qualify for L_2

NOTE: $(m,e) \notin C_2$ because e is small
ANTIMONOTONICITY

2nd iteration of repeat loop:
extend freq. 2-itemsets to create candidate freq. 3-itemsets

$\nexists$ extension of L_2 itemsets that can be a freq. 3-itemset

NOTE: $(m,j,b) \notin C_3$ because $(m,b) \notin L_2$
DOWNWARD CLOSURE

the APRIORI algorithm terminates with $L_1 = \{m, b, j, c\}$, $L_2 = \{(m,j), (b,c)\}$