Web Data Mining

The Apriori algorithm

- **The best known algorithm**
- **Two steps**:
  - Find all itemsets that have minimum support (*frequent itemsets*, also called large itemsets).
  - Use frequent itemsets to generate rules.

- E.g., a frequent itemset

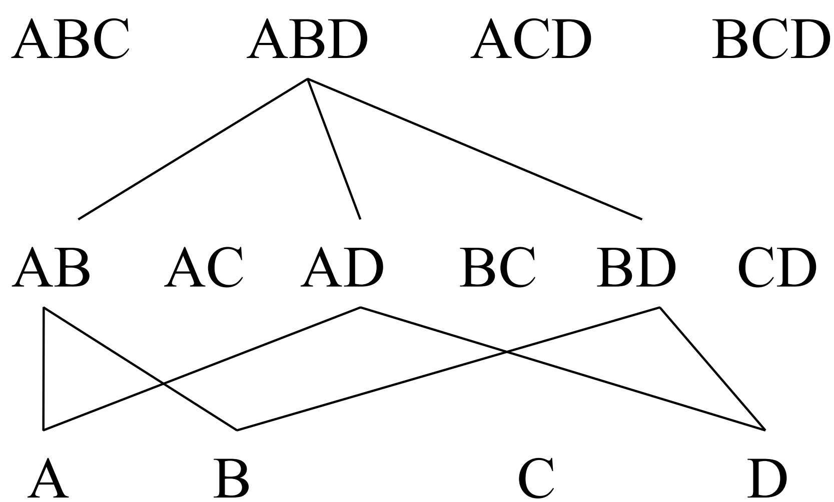
  {Chicken, Clothes, Milk} [sup = 3/7]

  and one rule from the frequent itemset

  Clothes $\rightarrow$ Milk, Chicken [sup = 3/7, conf = 3/3]

Step 1: Mining all frequent itemsets

- A frequent *itemset* is an itemset whose support is ≥ minsup.
- Key idea: The apriori property (downward closure property): any subsets of a frequent itemset are also frequent itemsets



The Algorithm

- Iterative algo. (also called level-wise search): Find all 1-item frequent itemsets; then all 2-item frequent itemsets, and so on.
  - In each iteration *k*, only consider itemsets that contain some *k*-1 frequent itemset.

- Find frequent itemsets of size 1: F_1
- From $k = 2$
  - C_k = candidates of size *k*: those itemsets of size *k* that could be frequent, given F_{k-1}
  - F_k = those itemsets that are actually frequent, $F_k \subseteq C_k$ (need to scan the database once).

Example – Finding frequent itemsets

Dataset T
minsup=0.5

TID	Items
T100	1, 3, 4
T200	2, 3, 5
T300	1, 2, 3, 5
T400	2, 5

itemset:count

1. scan T ➔ C_1: {1}:2, {2}:3, {3}:3, {4}:1, {5}:3
   ➔ F_1: {1}:2, {2}:3, {3}:3, {5}:3
   ➔ C_2: {1,2}, {1,3}, {1,5}, {2,3}, {2,5}, {3,5}
2. scan T ➔ C_2: {1,2}:1, {1,3}:2, {1,5}:1, {2,3}:2, {2,5}:3, {3,5}:2
   ➔ F_2: **{1,3}**:2, **{2,3}**:2, **{2,5}**:3, **{3,5}**:2
   ➔ C_3: {2, 3,5}
3. scan T ➔ C_3: {**2, 3, 5**}:2 ➔ F_3: {**2, 3, 5**}

Details: ordering of items

- The items in *I* are sorted in lexicographic order (which is a total order).
- The order is used throughout the algorithm in each itemset.
- $\{w[1], w[2], \ldots, w[k]\}$ represents a k-itemset w consisting of items $w[1], w[2], \ldots, w[k]$, where $w[1] < w[2] < \ldots < w[k]$ according to the total order.

Details: the algorithm

Algorithm Apriori(T)

$C_1 \leftarrow$ init-pass(T);

$F_1 \leftarrow \{f \mid f \in C_1, f.\text{count}/n \geq minsup\}$;   // n: no. of transactions in T

for (k = 2; $F_{k-1} \neq \emptyset$; 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**

        $c.count$++;

    end

  end

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

end

return $F \leftarrow \bigcup_k F_k$;

Apriori candidate generation

- The candidate-gen function takes $F_{k\text{-}1}$ and returns a superset (called the candidates) of the set of all frequent *k*-itemsets. It has two steps
  - *join* step: Generate all possible candidate itemsets C_k of length k
  - *prune* step: Remove those candidates in C_k that cannot be frequent.

Candidate-gen function

Function candidate-gen(F_{k-1})
$C_k \leftarrow \emptyset$;
forall $f_1, f_2 \in F_{k-1}$
    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**
    $c \leftarrow \{i_1, \ldots, i_{k-1}, i'_{k-1}\}$;   // join f_1 and f_2
    $C_k \leftarrow C_k \cup \{c\}$;
    for each (k-1)-subset s of c **do**
    if ($s \notin F_{k-1}$) **then**
        delete c from C_k;   // prune
    end
end
return C_k;

An example

- F_3 = {{1, 2, 3}, {1, 2, 4}, {1, 3, 4}, {1, 3, 5}, {2, 3, 4}}

- After join
  - C_4 = {{1, 2, 3, 4}, {1, 3, 4, 5}}
- After pruning:
  - C_4 = {{1, 2, 3, 4}}

    because {1, 4, 5} is not in F_3 ({1, 3, 4, 5} is removed)

Step 2: Generating rules from frequent itemsets

- Frequent itemsets $\neq$ association rules
- One more step is needed to generate association rules
- For each frequent itemset *X*,

  For each proper nonempty subset *A* of *X*,
  - Let *B* = X - *A*
  - A $\rightarrow$ B is an association rule if
    - Confidence(A $\rightarrow$ B) $\geq$ minconf,

      support(A $\rightarrow$ B) = support(A$\cup$B) = support(X)

      confidence(A $\rightarrow$ B) = support(A $\cup$ B) / support(A)

Generating rules: an example

- Suppose {2,3,4} is frequent, with sup=50%
  - Proper nonempty subsets: {2,3}, {2,4}, {3,4}, {2}, {3}, {4}, with sup=50%, 50%, 75%, 75%, 75%, 75% respectively
  - These generate these association rules:
    - 2,3 $\rightarrow$ 4, confidence=100%
    - 2,4 $\rightarrow$ 3, confidence=100%
    - 3,4 $\rightarrow$ 2, confidence=67%
    - 2 $\rightarrow$ 3,4, confidence=67%
    - 3 $\rightarrow$ 2,4, confidence=67%
    - 4 $\rightarrow$ 2,3, confidence=67%
    - All rules have support = 50%

Generating rules: summary

- To recap, in order to obtain A $\rightarrow$ B, we need to have support(A $\cup$ B) and support(A)
- All the required information for confidence computation has already been recorded in itemset generation. No need to see the data *T* any more.
- This step is not as time-consuming as frequent itemsets generation.

On Apriori Algorithm

Seems to be very expensive

- Level-wise search
- K = the size of the largest itemset
- It makes at most K passes over data
- In practice, K is bounded (10).
- The algorithm is very fast. Under some conditions, all rules can be found in linear time.
- Scale up to large data sets

More on association rule mining

- Clearly the space of all association rules is exponential, $O(2^m)$, where m is the number of items in *I*.
- The mining exploits sparseness of data, and high minimum support and high minimum confidence values.
- Still, it always produces a huge number of rules, thousands, tens of thousands, millions, ...

Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Sequential pattern mining
- Summary

Different data formats for mining

- The data can be in transaction form or table form

  Transaction form:

  a, b
  a, c, d, e
  a, d, f

  Table form:

Attr1	Attr2	Attr3
a,	b,	d
b,	c,	e

- Table data need to be converted to transaction form for association mining

From a table to a set of transactions

Table form:

Attr1	Attr2	Attr3
a,	b,	d
b,	c,	e

⇒ Transaction form:

(Attr1, a), (Attr2, b), (Attr3, d)

(Attr1, b), (Attr2, c), (Attr3, e)

candidate-gen can be slightly improved. Why?

Example

Consider the following transactions:

(Bread, Eggs, Butter)

(Eggs, Bread, Milk)

(Cheese, Chips)

(Chips, Milk, Egg)

Find all rules that satisfy minimum support=0.5 and minimum confidence=0.5