723A75 Advanced Data Mining
TDDD41 Data Mining - Clustering and Association Analysis

Lecture 9: Summary and Exercise

Johan Alenlöv
IDA, Linköping University, Sweden

Outline

- Content
  - Summary
  - Proofs
  - Exercise

Association Rules

- **Goal:** Given a **transactional database** find association rules on the form

$$\underbrace{X_1, \dots, X_n}_{(antecedent)} \rightarrow \underbrace{Y_1, \dots, Y_m}_{(consequent)}$$

  with a user-specified minimum **support** and **confidence**.
- **Support**: The fraction of transactions that contains the **full rule** $X \cup Y$. $(p(X \cup Y))$
- **Confidence**: The fraction of transactions that contains X that also contains Y. $(p(Y \mid X))$
- **Why?** Help with decision making.
- Note that association **is not** causality.
- Two step solution:
  1. Generate **all** itemsets with a given minimum support.
  2. Generate **all** rules from these itemsets with minimum confidence.

Apriori Property

- For generating frequent itemsets the following **apriori property** is important.
  - Every subset of a frequent itemset is frequent.
  - Alternatively every superset of an infrequent itemset is infrequent.
- Two algorithms for generating frequent itemsets

Apriori algorithm Using the apriori property to generate candidate sets that are tested.
Use the sets of length k to generate and test candidates of length $k+1$.

FP Grow Construct an FP-tree and find the itemsets by looking at the conditional databases.
Constructs itemsets by building the chains with specific suffixes first.

Rule Generation

- Given a frequent itemset L we wish to find a subset $X \subseteq L$ such that the rule $X \to L \setminus X$ has minimum confidence.
- Using the following property, if $X' \subseteq X$ then

$$\mathrm{Conf}(X \to L \setminus X) \geq \mathrm{Conf}(X' \to L \setminus X'),$$

  we can reduce the number of sets to check.
- The algorithm goes over each subset (starting with maximal size) and then checking all subsets to find rules with minimum support.

Constraints

- Other constraints can be added, such as minimum price, range of prices, sum of prices, etc.
- Constraints can be,

  Monotone If it is true for a set X then it is true for every superset X'. $(X \subseteq X')$

  Antimonotone If it is true for a set X then it is true for every subset X'. $(X \supseteq X')$

  Convertible Monotone If the items are sorted (in some way) then it is **monotone**.

  Convertible Antimonotone If the items are sorted (in some way) then it is **antimonotone**.

  Strongly convertible If it is both **convertible monotone** and **convertible antimonotone**.

  Inconvertible Can't be converted.

- Depending on the type of constraint different modifications to the algorithms are made.

Apriori Algorithm Proof

Algorithm: apriori(D, *minsup*)
Input: A transactional database D and the minimum support *minsup*.
Output: All the large itemsets in D.

$L_1 = \{$ large 1-itemsets $\}$
for $(k = 2; L_{k-1} \neq \emptyset; k++)$ do
  $C_k =$ apriori-gen(L_{k-1}) // Generate candidate large k-itemsets
  for all $t \in D$ do
    for all $c \in C_k$ such that $c \in t$ do
      $c.count++$
  $L_k = \{c \in C_k | c.count \geq minsup\}$
return $\bigcup_k L_k$

Algorithm: apriori-gen(L_{k-1})
Input: Large $(k-1)$-itemsets.
Output: A superset of L_k.

$C_k = \emptyset$ // Self-join
for all $I, J \in L_{k-1}$ do
  if $I_1 = J_1, \ldots, I_{k-2} = J_{k-2}$ and $I_{k-1} < J_{k-1}$ then
    add $\{I_1, \ldots, I_{k-1}, J_{k-1}\}$ to C_k
for all $c \in C_k$ do // Prune
  for all $(k-1)$-subsets s of c do
    if $s \notin L_{k-1}$ then
      remove c from C_k
return C_k

- We prove by induction on k that the algorithm is enough.
- $k = 1$ is trivial.
- Induction hypothesis: Assume that the algorithm is correct up to $k - 1$.
  We now want to prove that the algorithm is correct for k.
  It is enough to show that $L_k \subseteq C_k$.
- We perform a proof by contradiction. Assume that $I \in L_k$ but $I \notin C_k$. Then,
  - $\{I_1, I_2, \ldots, I_{k-2}, I_{k-1}\} \in L_{k-1}$ follows from $I \in L_k$ by the apriori property and the induction hypothesis.
  - $\{I_1, I_2, \ldots, I_{k-2}, I_k\} \in L_{k-1}$ follows from $I \in L_k$ by the apriori property and the induction hypothesis.
  - Then $I \in C_k$ by the self-join step.
  - Since $I \in L_k$, every subset of I is large by the apriori property.
  - Thus I will **not** be removed in the prune-step and $I \in C_k$
  - This is a contradiction and thus the algorithm is correct for k.

Rule Generation Algorithm Proof

```
for all large itemsets l_k with k ≥ 2 do
    call genrules(l_k, l_k, minconf)

Algorithm: genrules(l_k, a_m, minconf)
Input: A large itemset l_k, a set a_m ⊆ l_k, the minimum confidence minconf.
Output: All the rules of the form a → l_k \ a with a ⊆ a_m and confidence equal or above minconf.

𝔸 = {(m − 1)-itemsets a_{m−1} | a_{m−1} ⊆ a_m}
for all a_{m−1} ∈ 𝔸 do
    conf = support(l_k) / support(a_{m−1})          // Confidence of the rule a_{m−1} → l_k \ a_{m−1}
    if conf ≥ minconf then
        output the rule a_{m−1} → l_k \ a_{m−1} with confidence=conf and support=support(l_k)
        if m − 1 > 1 then call genrules(l_k, a_{m−1}, minconf)
```

- We prove by contradiction that the rule generation algorithm is correct.
- Assume that the algorithm missed a rule. Let $a_{m-1} \rightarrow l_k \setminus a_{m-1}$ denote one of the missing rules with the largest antecedent. Then,
  - Note that l_k has minimum support and, thus, it is outputted by the apriori algorithm since this is correct.
  - Then, the rule generation algorithm cannot have missed the rule if $m = k$.
  - Moreover if $m < k$, then

$$\begin{aligned}\text{confidence}(a_m \rightarrow l_k \setminus a_m) = \text{support}(l_k)/\text{support}(a_m) &\geq \text{support}(l_k)/\text{support}(a_{m-1}) \\ &= \text{confidence}(a_{m-1} \rightarrow l_k \setminus a_{m-1}) \geq \textit{minconf}.\end{aligned}$$

  - Note that the algorithm didn't miss the rule $a_m \rightarrow l_k \setminus a_m$.
  - Then the algorithm couldn't have missed the rule $a_{m-1} \rightarrow l_k \setminus a_{m-1}$.
  - This contradicts our assumption and, thus, the algorithm is correct.

Exercise

- Run the Apriori algorithm on the following transactional database with minimum support equal to one transaction. Explain step by step the execution.

Tid	Items
1	A, B, C
2	X, Y, Z
3	A, Y, C
4	X, B, Z

- Repeat with the constraint that the itemsets has to contain A. Make it clear when the constraint is used, don't just run the algorithm and consider the constraint at the end.
- Let the items A, B, C, X, Y, and Z, have the price of respectively $-3, -2, -1, 1, 2$, and 3 units. Repeat the exercise with the constraint: Find the frequent itemsets with range less than 3. Make it clear when the constraint is used, don't just run the algorithm and consider the constraint at the end.
- Repeat the exercises above with the FP grow algorithm
- Apply the rule generation algorithm to the frequent itemset XBZ on the database above in order to find association rules with confidence 0.5