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

Lecture 6: Apriori Algorithm

Johan Alenlöv
IDA, Linköping University, Sweden

Outline

- Content
  - Association Rules
  - Frequent Itemsets
  - Apriori Algorithm
  - Exercise
  - Rule Generation Algorithm
  - Exercise
  - Summary
- Litterature
  - Course Book. 2nd ed.: 5.2.1–2, 5.4. 3rd ed.: 6.2.1–2, 6.4
  - Agrawal, R and Srikant, R. **Fast Algorithms for Mining Association Rules**. In Proc. of the 20th Int. Conf. on Very Large Databases, 1994.

Association Rules

- Assume that we have access to some transactions

Transaction	Items
1	A, B, D
2	A, C, D
3	A, D, E
4	B, E, F
5	B, C, D, E, F

- Assume that the items are **sorted**.

Association Rules

- Assume that we have access to some transactions

Transaction	Items
1	A, B, D
2	A, C, D
3	A, D, E
4	B, E, F
5	B, C, D, E, F

- Assume that the items are **sorted**.
- Want to find association rules,

$$\mathrm{Item}_1, \ldots, \mathrm{Item}_m \to \mathrm{Item}_{m+1}, \ldots, \mathrm{Item}_n.$$

Association Rules

- Assume that we have access to some transactions

Transaction	Items
1	A, B, D
2	A, C, D
3	A, D, E
4	B, E, F
5	B, C, D, E, F

- Assume that the items are **sorted**.
- Want to find association rules,

$$\text{Item}_1, \ldots, \text{Item}_m \rightarrow \text{Item}_{m+1}, \ldots, \text{Item}_n.$$

  If the items in the antecedent are purchased, so are the items in the consequent, e.g.

$$\text{Bread}, \text{Butter} \rightarrow \text{Cheese}$$

- Application: Market basket analysis to support business decisions,

Association Rules

- Assume that we have access to some transactions

Transaction	Items
1	A, B, D
2	A, C, D
3	A, D, E
4	B, E, F
5	B, C, D, E, F

- Assume that the items are **sorted**.
- Want to find association rules,

$$\text{Item}_1, \ldots, \text{Item}_m \to \text{Item}_{m+1}, \ldots, \text{Item}_n.$$

  If the items in the antecedent are purchased, so are the items in the consequent, e.g.

$$\text{Bread}, \text{Butter} \to \text{Cheese}$$

- Application: Market basket analysis to support business decisions,
  - Rules with "cheese" in the consequent may help decide how to boost sales of "cheese".

Association Rules

- Assume that we have access to some transactions

Transaction	Items
1	A, B, D
2	A, C, D
3	A, D, E
4	B, E, F
5	B, C, D, E, F

- Assume that the items are **sorted**.
- Want to find association rules,

$$\mathrm{Item}_1, \ldots, \mathrm{Item}_m \to \mathrm{Item}_{m+1}, \ldots, \mathrm{Item}_n.$$

  If the items in the antecedent are purchased, so are the items in the consequent, e.g.

$$\mathrm{Bread}, \mathrm{Butter} \to \mathrm{Cheese}$$

- Application: Market basket analysis to support business decisions,
  - Rules with "cheese" in the consequent may help decide how to boost sales of "cheese".
  - Rules with "bread" in the antecedent may help determine what happens if "bread" is sold out.

Association Rules

- Assume that we have access to some transactions

Transaction	Items
1	A, B, D
2	A, C, D
3	A, D, E
4	B, E, F
5	B, C, D, E, F

- Assume that the items are **sorted**.
- Want to find association rules,

$$\text{Item}_1, \dots, \text{Item}_m \rightarrow \text{Item}_{m+1}, \dots, \text{Item}_n.$$

  If the items in the antecedent are purchased, so are the items in the consequent, e.g.

$$\text{Bread}, \text{Butter} \rightarrow \text{Cheese}$$

- Application: Market basket analysis to support business decisions,
  - Rules with "cheese" in the consequent may help decide how to boost sales of "cheese".
  - Rules with "bread" in the antecedent may help determine what happens if "bread" is sold out.
- Note that rules **do not** convey causality.

Association Rules

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \rightarrow Y_1, \ldots, Y_n \equiv X \rightarrow Y$$

Association Rules

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \rightarrow Y_1, \ldots, Y_n \equiv X \rightarrow Y$$

- All rules **are not** equally interesting.

Association Rules

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \rightarrow Y_1, \ldots, Y_n \equiv X \rightarrow Y$$

- All rules **are not** equally interesting.
- Interested in rules with minimum **support** and **confidence**

Association Rules

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \to Y_1, \ldots, Y_n \equiv X \to Y$$

- All rules **are not** equally interesting.
- Interested in rules with minimum **support** and **confidence**

  Support Fraction of the transactions which contain X and Y ($p(X, Y)$)

  Support "How general the rule is."

Association Rules

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \to Y_1, \ldots, Y_n \equiv X \to Y$$

- All rules **are not** equally interesting.
- Interested in rules with minimum **support** and **confidence**

Support Fraction of the transactions which contain X and Y ($p(X, Y)$)

Support "How general the rule is."

Confidence Fraction of the transactions that contain X which also contain Y ($p(Y \mid X)$)

Confidence "How accurate the rules is."

Association Rules

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \to Y_1, \ldots, Y_n \equiv X \to Y$$

- All rules **are not** equally interesting.
- Interested in rules with minimum **support** and **confidence**

Support Fraction of the transactions which contain X and Y ($p(X, Y)$)

Support "How general the rule is."

Confidence Fraction of the transactions that contain X which also contain Y ($p(Y \mid X)$)

Confidence "How accurate the rules is."

- $\text{confidence} = p(Y \mid X) = p(X, Y)/p(X) = \text{support}(X, Y)/\text{support}(X)$.

Association Rules

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \to Y_1, \ldots, Y_n \equiv X \to Y$$

- All rules **are not** equally interesting.
- Interested in rules with minimum **support** and **confidence**

Support Fraction of the transactions which contain X and Y ($p(X, Y)$)

Support "How general the rule is."

Confidence Fraction of the transactions that contain X which also contain Y ($p(Y|X)$)

Confidence "How accurate the rules is."

- $\text{confidence} = p(Y|X) = p(X, Y)/p(X) = \text{support}(X, Y)/\text{support}(X)$.
- Assume that we have access to some transactions

Transaction	Items
1	A, B, D
2	A, C, D
3	A, D, E
4	B, E, F
5	B, C, D, E, F

- $A \to D$

Association Rules

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \rightarrow Y_1, \ldots, Y_n \equiv X \rightarrow Y$$

- All rules **are not** equally interesting.
- Interested in rules with minimum **support** and **confidence**

  Support Fraction of the transactions which contain X and Y ($p(X, Y)$)

  Support "How general the rule is."

  Confidence Fraction of the transactions that contain X which also contain Y ($p(Y|X)$)

  Confidence "How accurate the rules is."

- $\text{confidence} = p(Y|X) = p(X, Y)/p(X) = \text{support}(X, Y)/\text{support}(X)$.
- Assume that we have access to some transactions

Transaction	Items
1	A, B, D
2	A, C, D
3	A, D, E
4	B, E, F
5	B, C, D, E, F

- $A \rightarrow D$ has support 0.6 and confidence 1.

Association Rules

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \to Y_1, \ldots, Y_n \equiv X \to Y$$

- All rules **are not** equally interesting.
- Interested in rules with minimum **support** and **confidence**

Support Fraction of the transactions which contain X and Y ($p(X, Y)$)

Support "How general the rule is."

Confidence Fraction of the transactions that contain X which also contain Y ($p(Y|X)$)

Confidence "How accurate the rules is."

- $\text{confidence} = p(Y|X) = p(X, Y)/p(X) = \text{support}(X, Y)/\text{support}(X)$.
- Assume that we have access to some transactions

Transaction	Items
1	A, B, D
2	A, C, D
3	A, D, E
4	B, E, F
5	B, C, D, E, F

- $A \to D$ has support 0.6 and confidence 1.
- $D \to A$

Association Rules

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \to Y_1, \ldots, Y_n \equiv X \to Y$$

- All rules **are not** equally interesting.
- Interested in rules with minimum **support** and **confidence**

  Support Fraction of the transactions which contain X and Y ($p(X, Y)$)

  Support "How general the rule is."

  Confidence Fraction of the transactions that contain X which also contain Y ($p(Y \mid X)$)

  Confidence "How accurate the rules is."

- $\text{confidence} = p(Y \mid X) = p(X, Y)/p(X) = \text{support}(X, Y)/\text{support}(X)$.
- Assume that we have access to some transactions

Transaction	Items
1	A, B, D
2	A, C, D
3	A, D, E
4	B, E, F
5	B, C, D, E, F

- $A \to D$ has support 0.6 and confidence 1.
- $D \to A$ has support 0.6 and confidence 0.75.

Frequent Itemsets

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \to Y_1, \ldots, Y_n \equiv X \to Y$$

with user-specified minimum **support** and **confidence**

Frequent Itemsets

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \rightarrow Y_1, \ldots, Y_n \equiv X \rightarrow Y$$

  with user-specified minimum **support** and **confidence**
- We define a **frequent** or **large** itemset as a set of items that has minimum support.
  - E.g., $\{A, D\}$ is a frequent itemset in the previous example with mimum support 0.5.

Frequent Itemsets

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \to Y_1, \ldots, Y_n \equiv X \to Y$$

  with user-specified minimum **support** and **confidence**
- We define a **frequent** or **large** itemset as a set of items that has minimum support.
  - E.g., $\{A, D\}$ is a frequent itemset in the previous example with mimum support 0.5.
- We will find the desired rules in two steps:

Frequent Itemsets

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \to Y_1, \ldots, Y_n \equiv X \to Y$$

  with user-specified minimum **support** and **confidence**
- We define a **frequent** or **large** itemset as a set of items that has minimum support.
  - E.g., $\{A, D\}$ is a frequent itemset in the previous example with mimum support 0.5.
- We will find the desired rules in two steps:
  1. Find all **frequent itemsets** (using apriori or FP grow algorithm).

Frequent Itemsets

- We are interested in finding rules of the form

$$X_1, \dots, X_m \to Y_1, \dots, Y_n \equiv X \to Y$$

  with user-specified minimum **support** and **confidence**
- We define a **frequent** or **large** itemset as a set of items that has minimum support.
  - E.g., $\{A, D\}$ is a frequent itemset in the previous example with mimum support 0.5.
- We will find the desired rules in two steps:
  1. Find all **frequent itemsets** (using apriori or FP grow algorithm).
  2. Generate all rules with minimum confidence from the frequent itemsets.

Frequent Itemsets

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \rightarrow Y_1, \ldots, Y_n \equiv X \rightarrow Y$$

  with user-specified minimum **support** and **confidence**
- We define a **frequent** or **large** itemset as a set of items that has minimum support.
  - E.g., $\{A, D\}$ is a frequent itemset in the previous example with mimum support 0.5.
- We will find the desired rules in two steps:
  1. Find all **frequent itemsets** (using apriori or FP grow algorithm).
  2. Generate all rules with minimum confidence from the frequent itemsets.
- The first step above will use the following **apriori property**:
  - Every subset of a frequent itemset is frequent.

Frequent Itemsets

- We are interested in finding rules of the form

$$X_1, \ldots, X_m \to Y_1, \ldots, Y_n \equiv X \to Y$$

  with user-specified minimum **support** and **confidence**
- We define a **frequent** or **large** itemset as a set of items that has minimum support.
  - E.g., $\{A, D\}$ is a frequent itemset in the previous example with mimum support 0.5.
- We will find the desired rules in two steps:
  1. Find all **frequent itemsets** (using apriori or FP grow algorithm).
  2. Generate all rules with minimum confidence from the frequent itemsets.
- The first step above will use the following **apriori property**:
  - Every subset of a frequent itemset is frequent.
  - Alternatively, every superset of an infrequent itemset is infrequent.

Apriori Algorithm

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
  $C_k = \text{apriori-gen}(L_{k-1})$ // 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

Example: Apriori Algorithm

Run the Apriori algorithm with the following database and minsup 2.

Tid	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

Apriori Algorithm

- Self-join step in MySQL:

  insert into C_k
  select $I.item_1, \ldots, I.item_{k-1}, J.item_{k-1}$
  from L_{k-1} I, L_{k-1} J
  where $I.item_1 = J.item_1, \ldots, I.item_{k-2} = J.item_{k-2}, I.item_{k-1} < J.item_{k-1}$

- Self-joint step in R:

  $$\text{merge}(L_{k-1},\ L_{k-1},\ \text{by=c}(L_{k-1}.item_1, \ldots, L_{k-1}.item_{k-2}))$$

  Note that **duplicates** will be produced because the condition $I.item_{k-1} < J.item_{k-1}$ is **not** enforced.

- To make the prune step fast, store the results in a **hash table**.
- Clever data structures are typically used for counting the support. (line 4-6 in apriori algorithm)

Exercise

- Run the apriori algorithm on the database below with minimum support 2.

Tid	Items
1	A, B, C
2	A, B, C, D, E
3	A, C, D
4	A, C, D, E
5	A, B, C, D

- Show the execution details (i.e. self-join, prune, support counting) not just the large itemsets.

$$L_1 = \{\{A\}, \{B\}, \{C\}, \{D\}, \{E\}\}$$
$$L_2 = \{\{A, B\}.\{A, C\}, \{A, D\}, \{A, E\}, \{B, C\}, \{B, D\}, \{C, D\}, \{C, E\}, \{D, E\}\}$$
$$L_3 = \{\{A, B, C\}, \{A, B, D\}, \{A, C, D\}, \{A, C, E\}, \{A, D, E\}, \{B, C, D\}, \{C, D, E\}\}$$
$$L_4 = \{\{A, B, C, D\}, \{A, C, D, E\}\}$$

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

Rule Generation Algorithm

- Having a large itemset L we wish to generate rules of the form

$$X \to L \setminus X,$$

where $X \subseteq L$.

Rule Generation Algorithm

- Having a large itemset L we wish to generate rules of the form

$$X \to L \setminus X,$$

  where $X \subseteq L$.
- These rules should have a minimum confidence.

Rule Generation Algorithm

- Having a large itemset L we wish to generate rules of the form

$$X \rightarrow L \setminus X,$$

  where $X \subseteq L$.
- These rules should have a minimum confidence.
- The following apriori property will be used:
  - If X does not result in a rule with minimum confidence for L, then neither does **any subset** $X' \subseteq X$,

Rule Generation Algorithm

- Having a large itemset L we wish to generate rules of the form

$$X \rightarrow L \setminus X,$$

where $X \subseteq L$.
- These rules should have a minimum confidence.
- The following apriori property will be used:
  - If X does not result in a rule with minimum confidence for L, then neither does **any subset** $X' \subseteq X$,

$$\text{confidence}(X \rightarrow L \setminus X) = \frac{\text{support}(L)}{\text{supportX}} \geq \frac{\text{support}(L)}{\text{support}(X')} = \text{confidence}(\mathrm{X}' \rightarrow \mathrm{L} \setminus \mathrm{X}')$$

Rule Generation Algorithm

- Having a large itemset L we wish to generate rules of the form

$$X \to L \setminus X,$$

where $X \subseteq L$.

- These rules should have a minimum confidence.
- The following apriori property will be used:
  - If X does not result in a rule with minimum confidence for L, then neither does **any subset** $X' \subseteq X$,

$$\mathrm{confidence}(X \to L \setminus X) = \frac{\mathrm{support}(L)}{\mathrm{supportX}} \geq \frac{\mathrm{support}(L)}{\mathrm{support}(X')} = \mathrm{confidence}(\mathrm{X'} \to \mathrm{L} \setminus \mathrm{X'})$$

for all large itemsets l_k with $k \geq 2$ do
    call genrules(l_k, l_k, *minconf*)

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

$\mathbb{A} = \{(m-1)$-itemsets $a_{m-1} | a_{m-1} \subseteq a_m\}$
for all $a_{m-1} \in \mathbb{A}$ do
    conf $=$ support(l_k) / support(a_{m-1})   // Confidence of the rule $a_{m-1} \to l_k \setminus a_{m-1}$
    if conf $\geq$ *minconf* then
        output the rule $a_{m-1} \to l_k \setminus a_{m-1}$ with confidence=conf and support=support(l_k)
        if $m-1 > 1$ then call genrules(l_k, a_{m-1}, *minconf*)

Exercise

- Run the genrule algorithm on the database below for the large itemset $\{A, B, C\}$ with minimum confidence 0.8.

Tid	Items
1	A, B, C
2	A, B, C, D, E
3	A, C, D
4	A, C, D, E
5	A, B, C, D

- Show the execution details (i.e. antecedent generation, recursive calls) not just the rules.

$$A, B \rightarrow C$$
$$B, C \rightarrow A$$
$$B \rightarrow A, C$$

Rule Generation Algorithm Proof

for all large itemsets l_k with $k \geq 2$ do
    call genrules(l_k, l_k, *minconf*)

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

$\mathbb{A} = \{(m-1)\text{-itemsets } a_{m-1} | a_{m-1} \subseteq a_m\}$
for all $a_{m-1} \in \mathbb{A}$ do
    conf = support(l_k) / support(a_{m-1})   // Confidence of the rule $a_{m-1} \rightarrow l_k \setminus a_{m-1}$
    if conf $\geq$ *minconf* then
        output the rule $a_{m-1} \rightarrow l_k \setminus a_{m-1}$ with confidence=conf and support=support(l_k)
        if $m-1 > 1$ then call genrules(l_k, a_{m-1}, *minconf*)

Summary

- Processing transactions to find rules of the form,

$$\text{Item}_1, \ldots, \text{Item}_m \rightarrow \text{Iten}_{m+1}, \ldots, \text{Item}_n,$$

  with a user-defined minimum support and confidence.
- We use a two-step solution:
  1. Find all the **large itemsets**.
  2. Generate all the **rules** with minimum confidence.
- We use the **apriori properties**.
- Drawbacks of the apriori algorithm:
  - Candidate generate-and-test.
  - Too many candidates to generate, e.g. if there are 10^4 large 1-itemsets, then more than 10^7 candidate 2-itemsets.
  - Each candidate implies expensive operations, e.g. pattern matching, subset checking, storing.
- Can candidate generation be avoided?