Chapter 2: Association Rules & Sequential Patterns

Instructor: Dr. Mehmet S. Aktas

Acknowledgement: Thanks to Dr. Bing Liu for teaching materials.

Road map

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

Association rule mining

- Proposed by Agrawal et al in 1993.
- It is an important data mining model studied extensively by the database and data mining community.
- Assume all data are categorical.
- No good algorithm for numeric data.
- Initially used for Market Basket Analysis to find how items purchased by customers are related.

Bread $\rightarrow$ Milk [sup = 5%, conf = 100%]

The model: data

- $I = \{i_1, i_2, \ldots, i_m\}$: a set of *items*.
- Transaction t :
  - t a set of items, and $t \subseteq I$.
- Transaction Database T: a set of transactions $T = \{t_1, t_2, \ldots, t_n\}$.

Transaction data: supermarket data

- Market basket transactions:

  t1: {bread, cheese, milk}

  t2: {apple, eggs, salt, yogurt}

  … …

  tn: {biscuit, eggs, milk}

- Concepts:
  - An *item*: an item/article in a basket
  - *I*: the set of all items sold in the store
  - A *transaction*: items purchased in a basket; it may have TID (transaction ID)
  - A *transactional dataset*: A set of transactions

Transaction data: a set of documents

- **A text document data set. Each document is treated as a “bag” of keywords**

doc1:	Student, Teach, School
doc2:	Student, School
doc3:	Teach, School, City, Game
doc4:	Baseball, Basketball
doc5:	Basketball, Player, Spectator
doc6:	Baseball, Coach, Game, Team
doc7:	Basketball, Team, City, Game

The model: rules

- A transaction *t* contains *X*, a set of items (itemset) in *I*, if $X \subseteq t$.
- An association rule is an implication of the form:

$$X \rightarrow Y, \text{ where } X, Y \subset I, \text{ and } X \cap Y = \emptyset$$

- An itemset is a set of items.
  - E.g., X = {milk, bread, cereal} is an itemset.
- A *k*-itemset is an itemset with *k* items.
  - E.g., {milk, bread, cereal} is a 3-itemset

Rule strength measures

- Support: The rule holds with support *sup* in *T* (the transaction data set) if sup% of transactions contain $X \cup Y$.
  - $sup = \Pr(X \cup Y)$.
- Confidence: The rule holds in *T* with confidence *conf* if *conf*% of tranactions that contain *X* also contain *Y*.
  - $conf = \Pr(Y \mid X)$
- An association rule is a pattern that states when *X* occurs, *Y* occurs with certain probability.

Support and Confidence

- Support count: The support count of an itemset *X*, denoted by *X.count*, in a data set *T* is the number of transactions in *T* that contain *X*. Assume *T* has *n* transactions.
- Then,

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

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

Goal and key features

- **Goal:** Find all rules that satisfy the user-specified *minimum support* (minsup) and *minimum confidence* (minconf).
- **Key Features**
  - Completeness: find all rules.
  - No target item(s) on the right-hand-side

An Example:

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

- Itemset X = {x_1, …, x_k}
- Find all the rules *X* → *Y* with minimum support and confidence
  - support, *s*, probability that a transaction contains X ∪ Y
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

- *Let* sup_{min} *= 50%,* $conf_{min}$ *= 50%*
- *Freq. Pat.: {A:3, B:3, D:4, E:3, AD:3}*
- Association rules:
  - *A* → *D* (60%, 100%)
  - *D* → *A* (60%, 75%)

Another example

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

- Transaction data
- Assume:

  minsup = 30%

  minconf = 80%

- An example frequent *itemset*:

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

- Association rules from the itemset:

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

  … …

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

Transaction data representation

- A simplistic view of shopping baskets,
- Some important information not considered. E.g,
  - the quantity of each item purchased and
  - the price paid.

Many mining algorithms

- There are a large number of them!!
- They use different strategies and data structures.
- Their resulting sets of rules are all the same.
  - Given a transaction data set *T*, and a minimum support and a minimum confident, the set of association rules existing in *T* is uniquely determined.
- Any algorithm should find the same set of rules although their computational efficiencies and memory requirements may be different.
- We study only one: the Apriori Algorithm

Road map

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

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

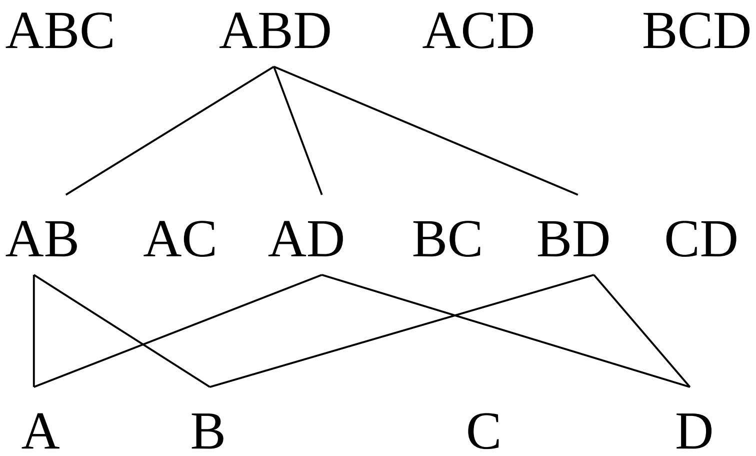
  {Chicken, Clothes, Milk}   [sup = 3/7]

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

    because {2, 3, 5} and {1, 4, 5} are not in F_3 ({1, 2, 3, 5} and {1, 3, 4, 5} are 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.
- Scale up to large data sets.

Road map

- Basic concepts of Association Rules
- Apriori algorithm
- Different data formats for mining
- Mining class association rules
- 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)

Road map

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

Mining class association rules (CAR)

- Normal association rule mining does not have any target.
- It finds all possible rules that exist in data, i.e., any item can appear as a consequent or a condition of a rule.
- However, in some applications, the user is interested in some targets.
  - E.g, the user has a set of text documents from some known topics. He/she wants to find out what words are associated or correlated with each topic.

Problem definition

- Let T be a transaction data set consisting of n transactions.
- Each transaction is also labeled with a class Y.
- Let I be the set of all items in T, Y be the set of all class labels and $I \cap Y = \emptyset$.
- A **class association rule** (**CAR**) is an implication of the form

  $X \rightarrow y$, where $X \subseteq I$, and $y \in Y$.
- The definitions of **support** and **confidence** are the same as those for normal association rules.

An example

- **A text document data set**

doc 1:	Student, Teach, School	: Education
doc 2:	Student, School	: Education
doc 3:	Teach, School, City, Game	: Education
doc 4:	Baseball, Basketball	: Sport
doc 5:	Basketball, Player, Spectator	: Sport
doc 6:	Baseball, Coach, Game, Team	: Sport
doc 7:	Basketball, Team, City, Game	: Sport

- Let *minsup* = 20% and *minconf* = 60%. The following are two examples of class association rules:

  Student, School $\rightarrow$ Education [sup= 2/7, conf = 2/2]

  game $\rightarrow$ Sport [sup= 3/7, conf = 2/3]

Mining algorithm

- Unlike normal association rules, CARs can be mined directly in one step.
- The key operation is to find all **ruleitems** that have support above *minsup*. A **ruleitem** is of the form:

  ($condset$, y)

  where **condset** is a set of items from I (*i.e.*, $condset \subseteq I$), and $y \in Y$ is a class label.
- Each ruleitem basically represents a rule:

  $condset \rightarrow y$,
- The Apriori algorithm can be modified to generate CARs

Road map

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

Sequential pattern mining

- Association rule mining does not consider the order of transactions.
- In many applications such orderings are significant. E.g.,
  - in market basket analysis, it is interesting to know whether people buy some items in sequence,
    - e.g., buying bed first and then bed sheets some time later.
  - In Web usage mining, it is useful to find navigational patterns of users in a Web site from sequences of page visits of users

Basic concepts

- Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items.
- **Itemset/element**: A non-empty set of items $X \subseteq I$.
- **Sequence:** An ordered list of itemsets. We denote a sequence s by $\langle a_1 a_2 \ldots a_r \rangle$, where a_i is an itemset, which is also called an **element** of s.
- An element (or an itemset) of a sequence is denoted by $\{x_1, x_2, \ldots, x_k\}$, where $x_j \in I$ is an item.
- We assume without loss of generality that items in an element of a sequence are in **lexicographic order**.

Basic concepts (contd)

- **Size**: The **size** of a sequence is the number of elements (or itemsets) in the sequence.
- **Length**: The **length** of a sequence is the number of items in the sequence.
  - A sequence of length k is called ***k*-sequence**.
- A sequence $s_1 = \langle a_1 a_2 \ldots a_r \rangle$ is a **subsequence** of another sequence $s_2 = \langle b_1 b_2 \ldots b_v \rangle$, or s_2 is a **supersequence** of s_1, if there exist integers $1 \leq j_1 < j_2 < \ldots < j_{r-1} < j_r \leq v$ such that $a_1 \subseteq b_{j1}$, $a_2 \subseteq b_{j2}$, …, $a_r \subseteq b_{jr}$. We also say that s_2 **contains** s_1.

An example

- Let $I = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
- Sequence $\langle\{3\}\{4, 5\}\{8\}\rangle$ is **contained** in (or is a **subsequence** of) $\langle\{6\}\ \{3, 7\}\{9\}\{4, 5, 8\}\{3, 8\}\rangle$
  - because $\{3\} \subseteq \{3, 7\}$, $\{4, 5\} \subseteq \{4, 5, 8\}$, and $\{8\} \subseteq \{3, 8\}$.
  - The size of the sequence $\langle\{3\}\{4, 5\}\{8\}\rangle$ is 3, and the length of the sequence is 4.

Objective

- Given a set *S* of input data sequences (or sequence database), the problem of mining sequential patterns is to find all the sequences that have a user-specified minimum support.
- Each such sequence is called a **frequent sequence**, or a **sequential pattern**.
- The **support** for a sequence is the fraction of total data sequences in *S* that contains this sequence.

Example

Table 1. A set of transactions sorted by customer ID and transaction time

Customer ID	Transaction Time	Transaction (items bought)
1	July 20, 2005	30
1	July 25, 2005	90
2	July 9, 2005	10, 20
2	July 14, 2005	30
2	July 20, 2005	40, 60, 70
3	July 25, 2005	30, 50, 70
4	July 25, 2005	30
4	July 29, 2005	40, 70
4	August 2, 2005	90
5	July 12, 2005	90

Example (cond)

Table 2. Data sequences produced from the transaction database in Table 1.

Customer ID	Data Sequence
1	⟨{30} {90}⟩
2	⟨{10, 20} {30} {40, 60, 70}⟩
3	⟨{30, 50, 70}⟩
4	⟨{30} {40, 70} {90}⟩
5	⟨{90}⟩

Table 3. The final output sequential patterns

	Sequential Patterns with Support ≥ 25%
1-sequences	⟨{30}⟩, ⟨{40}⟩, ⟨{70}⟩, ⟨{90}⟩
2-sequences	⟨{30} {40}⟩, ⟨{30} {70}⟩, ⟨{30} {90}⟩, ⟨{40, 70}⟩
3-sequences	⟨{30} {40, 70}⟩

GSP mining algorithm

- Very similar to the Apriori algorithm

Algorithm GSP(S)

$C_1 \leftarrow$ init-pass(S); // the first pass over S

$F_1 \leftarrow \{\langle\{f\}\rangle \mid f \in C_1, f.\text{count}/n \geq minsup\}$; // n is the number of sequences in S

for ($k = 2$; $F_{k-1} \neq \emptyset$; k++) **do** // subsequent passes over S

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

    for each data sequence $s \in S$ **do** // scan the data once

        for each candidate $c \in C_k$ **do**

            if c is contained in s **then**

                $c.count$++; // increment the support count

        end

    end

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

end

return $\bigcup_k F_k$;

Fig. 12. The GSP Algorithm for generating sequential patterns

Road map

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

Summary

- Association rule mining has been extensively studied in the data mining community.
- So is sequential pattern mining
- There are many efficient algorithms and model variations.
- Other related work includes
  - Multi-level or generalized rule mining
  - Constrained rule mining
  - Incremental rule mining
  - Maximal frequent itemset mining
  - Closed itemset mining
  - Rule interestingness and visualization
  - Parallel algorithms
  - …