Why do we need association rule mining at all?

Motivation for Association Rules(1)

Association Rule Mining can help to better understand purchase behavior!!

For instance, **{beer} => {chips}**

Market Basket Analysis

- In retailing, *most purchases are bought on impulse*. Market basket analysis gives clues as to what a customer might have bought *if the idea had occurred to them*.

  → decide the location and promotion of goods inside a store.

Observation: Purchasers of Barbie dolls are more likely to buy candy.

{barbie doll} => {candy}

→ place high-margin candy near to the Barbie doll display.

Create Temptation: Customers who would have bought candy with their Barbie dolls *had they thought of it* will now be suitably tempted.

Cont…

- Further possibilities:
  - comparing results between different stores, between customers in different demographic groups, between different days of the week, different seasons of the year, etc.

personalization

  - If we observe that a rule holds in one store, but not in any other then we know that there is something interesting about that store.
    - different clientele
    - different organization of its displays (in a more lucrative way …)

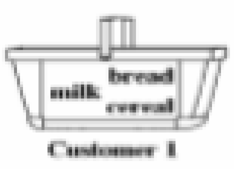
→ investigating such differences may yield useful insights which will improve company sales.

ReCap: Let' s go shopping

- **Objective of Association Rule Mining:**
  - find associations and correlations between different items (products) that customers place in their shopping basket.
  - to better predict, e.g., :

    (i) what my customers buy? (→ spectrum of products)
    (ii) when they buy it? (→ advertizing)
    (ii) which products are bought together? (→ placement)

Introduction into AR



- Formalizing the problem a little bit
  - Transaction Database T: a set of transactions $T = \{t_1, t_2, \ldots, t_n\}$
  - Each transaction contains a set of items (item set)
  - An item set is a collection of items $I = \{i_1, i_2, \ldots, i_m\}$
- General Aim:
  - Find frequent/interesting patterns, associations, correlations, or causal structures among sets of items or elements in databases or other information repositories.
  - Put this relationships in terms of association rules

$$X \Rightarrow Y$$

- where X, Y represent two itemsets

Examples of AR

TID	Items
T1	bread, jelly, peanut-butter
T2	bread, peanut-butter
T3	bread, milk, peanut-butter
T4	beer, bread
T5	beer, milk

Examples: Quality?

- bread ⇒ peanut-butter
- beer ⇒ bread

Reads as:
If you buy bread, then you will peanut-butter as well.

- Frequent Item Sets:
  - Items that appear frequently together
  - I = {bread, peanut-butter}
  - I = {beer, bread}

What is an interesting rule?

TID	Items
T1	bread, jelly, peanut-butter
T2	bread, peanut-butter
T3	bread, milk, peanut-butter
T4	beer, bread
T5	beer, milk

- **Support Count (σ)**
  - Frequency of occurrence of an itemset

    σ ({bread, peanut-butter}) = 3

    σ ({beer, bread}) = 1
- **Support (s)**
  - Fraction of transactions that contain an itemset

    s({bread, peanut-butter}) = 3/5 (0.6)

    s ({beer, bread}) = 1/5 (0.2)
- **Frequent Itemset**
  - = an itemset whose support is greater than or equal to a minimum support threshold (minsup)

What is an interesting rule?

- An association rule is an implication of two itemsets

  $X \Rightarrow Y$

- Most common measures:
  - Support (s)
    - The occurring frequency of the rule, i.e., the number of transactions that contain both X and Y
  - Confidence (c)
    - The strength of the association, i.e., measures the number of how often items in Y appear in transactions that contain X vs. the number of how often items in X occur in general

TID	Items
T1	bread, jelly, peanut-butter
T2	bread, peanut-butter
T3	bread, milk, peanut-butter
T4	beer, bread
T5	beer, milk

$$s = \frac{\sigma(X \cup Y)}{\#\text{ of trans.}}$$

$$c = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

Interestingness of Rules

- Let's have a look at some associations + the corresponding measures

TID	s	c
bread ⇒ peanut-butter	0.60	0.75
peanut-butter ⇒ bread		
beer ⇒ bread		
peanut-butter ⇒ jelly		
jelly ⇒ peanut-butter		
jelly ⇒ milk		

TID	Items
T1	bread, jelly, peanut-butter
T2	bread, peanut-butter
T3	bread, milk, peanut-butter
T4	beer, bread
T5	beer, milk

$$s = \frac{\sigma(X \cup Y)}{\#\text{ of trans.}}$$

$$c = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

- Support is symmetric / Confidence is asymmetric
- Confidence does not take frequency into account

Mining Association Rules

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Min. support 50%
Min. confidence 50%

Frequent Itemset	Support
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

For rule $A \Rightarrow C$:

support = support($\{A \cap C\}$) = 50%

confidence = support($\{A \cap C\}$)/support($\{A\}$) = 66.6%

The Apriori principle:

Any subset of a frequent itemset must be frequent

Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \dots, a_{100}\}$ contains $\binom{100}{1} + \binom{100}{2} + \dots + \binom{100}{100} = 2^{100} - 1 = 1.27 * 10^{30}$ sub-patterns!
- Solution: *Mine closed patterns and max-patterns instead*
- An itemset X is closed if X is *frequent* and there exists *no super-pattern* Y ɔ X, *with the same support* as X (proposed by Pasquier, et al. @ ICDT'99)
- An itemset X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ɔ X (proposed by Bayardo @ SIGMOD'98)
- Closed pattern is a lossless compression of freq. patterns
    - Reducing the # of patterns and rules

Computational Complexity of Frequent Itemset Mining

- How many itemsets are potentially to be generated in the worst case?
  - The number of frequent itemsets to be generated is senstive to the minsup threshold
  - When minsup is low, there exist potentially an exponential number of frequent itemsets
  - The worst case: M^N where M: # distinct items, and N: max length of transactions
- The worst case complexty vs. the expected probability
  - Ex. Suppose Walmart has 10^4 kinds of products
    - The chance to pick up one product 10^{-4}
    - The chance to pick up a particular set of 10 products: $\sim 10^{-40}$
    - What is the chance this particular set of 10 products to be frequent 10^3 times in 10^9 transactions?

Apriori

- **Apriori algorithm** is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemsets in a dataset for boolean association rule.
- Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties.
- We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets.
- To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called *Apriori property* which helps by reducing the search space.

Cont…

- **Apriori Property –**
  All non-empty subset of frequent itemset must be frequent. The key concept of Apriori algorithm is its anti-monotonicity of support measure. Apriori assumes that
- *All subsets of a frequent itemset must be frequent(Apriori propertry). If an itemset is infrequent, all its supersets will be infrequent.*

Cont…

- The steps followed in the Apriori Algorithm of data mining are:
- Join Step: This step generates (K+1) itemset from K-itemsets by joining each item with itself.
- Prune Step: This step scans the count of each item in the database. If the candidate item does not meet minimum support, then it is regarded as infrequent and thus it is removed. This step is performed to reduce the size of the candidate itemsets.

Steps In Apriori

- Apriori algorithm is a sequence of steps to be followed to find the most frequent itemset in the given database. This data mining technique follows the join and the prune steps iteratively until the most frequent itemset is achieved. A minimum support threshold is given in the problem or it is assumed by the user.
- #1) In the first iteration of the algorithm, each item is taken as a 1-itemsets candidate. The algorithm will count the occurrences of each item.
- #2) Let there be some minimum support, min_sup (eg 2). The set of 1 – itemsets whose occurrence is satisfying the min sup are determined. Only those candidates which count more than or equal to min_sup, are taken ahead for the next iteration and the others are pruned.

Cont…

- #3) Next, 2-itemset frequent items with min_sup are discovered. For this in the join step, the 2-itemset is generated by forming a group of 2 by combining items with itself.
- #4) The 2-itemset candidates are pruned using min-sup threshold value. Now the table will have 2 –itemsets with min-sup only.
- #5) The next iteration will form 3 –itemsets using join and prune step. This iteration will follow antimonotone property where the subsets of 3-itemsets, that is the 2 –itemset subsets of each group fall in min_sup. If all 2-itemset subsets are frequent then the superset will be frequent otherwise it is pruned.
- #6) Next step will follow making 4-itemset by joining 3-itemset with itself and pruning if its subset does not meet the min_sup criteria. The algorithm is stopped when the most frequent itemset is achieved.

Cont…

TABLE-1

Transaction	List of items
T1	I1,I2,I3
T2	I2,I3,I4
T3	I4,I5
T4	I1,I2,I4
T5	I1,I2,I3,I5
T6	I1,I2,I3,I4

Solution:
Support threshold=50% => 0.5*6= 3 => min_sup=3

1. Count Of Each Item

TABLE-2

Item	Count
I1	4
I2	5
I3	4
I4	4
I5	2

Cont…

- 2. Prune Step: TABLE -2 shows that I5 item does not meet min_sup=3, thus it is deleted, only I1, I2, I3, I4 meet min_sup count.

TABLE-3

Item	Count
I1	4
I2	5
I3	4
I4	4

Cont...

- 3. Join Step: Form 2-itemset. From TABLE-1 find out the occurrences of 2-itemset.

TABLE-4

Item	Count
I1,I2	4
I1,I3	3
I1,I4	2
I2,I3	4
I2,I4	3
I3,I4	2

Cont…

- 4. Prune Step: TABLE -4 shows that item set {I1, I4} and {I3, I4} does not meet min_sup, thus it is deleted.

TABLE-5

Item	Count
I1,I2	4
I1,I3	3
I2,I3	4
I2,I4	3

Cont…

- 5. Join and Prune Step: Form 3-itemset. From the TABLE- 1 find out occurrences of 3-itemset. From TABLE-5, find out the 2-itemset subsets which support min_sup. We can see for itemset {I1, I2, I3} subsets, {I1, I2}, {I1, I3}, {I2, I3} are occurring in TABLE-5 thus {I1, I2, I3} is frequent. We can see for itemset {I1, I2, I4} subsets, {I1, I2}, {I1, I4}, {I2, I4}, {I1, I4} is not frequent, as it is not occurring in TABLE-5 thus {I1, I2, I4} is not frequent, hence it is deleted.

Cont…

TABLE-6

Item
I1,I2,I3
I1,I2,I4
I1,I3,I4
I2,I3,I4

Only {I1, I2, I3} is frequent.

- 6. Generate Association Rules: From the frequent itemset discovered above the association could be:
- {I1, I2} => {I3}
- Confidence = support {I1, I2, I3} / support {I1, I2} = (3/ 4)* 100 = 75%
- {I1, I3} => {I2}
- Confidence = support {I1, I2, I3} / support {I1, I3} = (3/ 3)* 100 = 100%
- {I2, I3} => {I1}
- Confidence = support {I1, I2, I3} / support {I2, I3} = (3/ 4)* 100 = 75%
- {I1} => {I2, I3}
- Confidence = support {I1, I2, I3} / support {I1} = (3/ 4)* 100 = 75%
- {I2} => {I1, I3}
- Confidence = support {I1, I2, I3} / support {I2 = (3/ 5)* 100 = 60%
- {I3} => {I1, I2}
- Confidence = support {I1, I2, I3} / support {I3} = (3/ 4)* 100 = 75%
- This shows that all the above association rules are strong if minimum confidence threshold is 60%.

Associative Classification

- Association rules are mined in a two-step process consisting of frequent itemset mining followed by rule generation.
- The first step searches for patterns of attribute–value pairs that occur repeatedly in a data set, where each attribute–value pair is considered an item. The resulting attribute–value pairs form frequent itemsets (also referred to as frequent patterns).

- The second step analyzes the frequent itemsets to generate association rules. All association rules must satisfy certain criteria regarding their "accuracy" (or confidence) and the proportion of the data set that they actually represent (referred to as support).

- For example, the following is an association rule mined from a data set, D, shown with its confidence and support: age = youth ∧ credit = OK ⇒ buys computer = yes [support = 20%, confidence = 93%]

- Association rules can have any number of items in the rule antecedent (left side) and any number of items in the rule consequent (right side).
- However, when mining association rules for use in classification, we are only interested in association rules of the form $p1 \wedge p2 \wedge \ldots pl \Rightarrow Aclass = C$, where the rule antecedent is a conjunction of items, p1, p2,..., pl ($l \leq n$), associated with a class label, C.
- For a given rule, R, the percentage of tuples in D satisfying the rule antecedent that also have the class label C is called the confidence of R

- For example, a confidence of 93% for Rule means that 93% of the customers in D who are young and have an OK credit rating belong to the class buys computer = yes.
- The percentage of tuples in D satisfying the rule antecedent and having class label C is called the support of R. A support of 20% for Rule means that 20% of the customers in D are young, have an OK credit rating, and belong to the class buys computer = yes.

- Associative classification consists of the following steps:
- 1. Mine the data for frequent itemsets, that is, find commonly occurring attribute–value pairs in the data.
- 2. Analyze the frequent itemsets to generate association rules per class, which satisfy confidence and support criteria.
- 3. Organize the rules to form a rule-based classifier.

CBA

- CBA uses a heuristic method to construct the classifier, where the rules are ordered according to decreasing precedence based on their confidence and support.
- If a set of rules has the same antecedent, then the rule with the highest confidence is selected to represent the set.
- When classifying a new tuple, the first rule satisfying the tuple is used to classify it. The classifier also contains a default rule, having lowest precedence, which specifies a default class for any new tuple that is not satisfied by any other rule in the classifier.

The Classification Problem

record	*class*
i_1 , i_2 , i_3	c_1
i_2 , i_3	c_1
i_1 , i_4	c_2
i_1 , i_3	c_1

attributes

$I = \{i_1 , i_2 , i_3 , i_4\}$

classes

$C = \{c_1,c_2\}$

The Classification Problem

record	*class*
i_1, i_2, i_3	c_1
i_2, i_3	c_1
i_1, i_4	c_2
i_1, i_3	c_1

attributes

$I = \{i_1, i_2, i_3, i_4\}$

classes

$C = \{c_1, c_2\}$

i_1, i_2, i_4	**?**

The Classification Problem

record	*class*
i_1, i_2, i_3	c_1
i_2, i_3	c_1
i_1, i_4	c_2
i_1, i_3	c_1

attributes

$I = \{i_1, i_2, i_3, i_4\}$

classes

$C = \{c_1, c_2\}$

query

i_1, i_2, i_4	?

Classification based on Associations (CBA)

- [Bing Liu – KDD98]
- First Classifier that used the paradigm of Association Rules

- Steps in CBA:
  - Mine for CARs satisfying support and confidence thresholds
  - Sort all CARs based on confidence
  - Classify using the rule that satisfies the query and has the highest confidence

Classification based on Associations (CBA)

- [Bing Liu – KDD98]
- First Classifier that used the paradigm of Association Rules

- Steps in CBA:
  - Mine for CARs satisfying support and confidence thresholds
  - Sort all CARs based on confidence
  - Classify using the rule that satisfies the query and has the highest confidence

- Disadvantages:
  - Single rule based classification – Not Robust
  - Cannot handle Fully Confident Associations

Classification based on Associations (CBA)

- [Bing Liu – KDD98]
- First Classifier that used the paradigm of Association Rules

- Steps in CBA:
  - Mine for CARs satisfying support and confidence thresholds
  - Sort all CARs based on confidence
  - Classify using the rule that satisfies the query and has the highest confidence

- Disadvantages:
  - Single rule based classification – Not Robust
  - Cannot handle Fully Confident Associations

Disadvantages with CBA: Single Rule based classification

- Let the classifier have 3 rules :
  - $i_1 \rightarrow c_1$ support: 0.3, confidence: 0.8
  - $i_2, i_3 \rightarrow c_2$ support: 0.7, confidence: 0.7
  - $i_2, i_4 \rightarrow c_2$ support: 0.8, confidence: 0.7

- Query $\{i_1, i_2, i_3, i_4\}$ will be classified to the class c_1 by CBA which might be incorrect.

- CBA, being a single-rule classifier, cannot consider the effects of multiple-parameters.

Classification based on Associations (CBA)

- [Bing Liu – KDD98]
- First Classifier that used the paradigm of Association Rules

- Steps in CBA:
  - Mine for CARs satisfying support and confidence thresholds
  - Sort all CARs based on confidence
  - Classify using the rule that satisfies the query and has the highest confidence

- Disadvantages:
  - Single rule based classification – Not Robust
  - Cannot handle **Fully Confident Associations**

Fully Confident Associations

- An Association $\{ i_1 , i_2 \} \rightarrow \{ i_3 \}$ is **fully confident** if its confidence is 100%.

- If CBA includes the CAR $\{ i_1 , i_2 , i_3 \} \rightarrow c_1$ it will also include $\{ i_1 , i_2 \} \rightarrow c_1$

- CBA does not check for all statistical relationships.

CMAR

- CMAR uses a weighted χ 2 measure to find the "strongest" group of rules, based on the statistical correlation of rules within a group. It then assigns X the class label of the strongest group.
- In this way it considers multiple rules, rather than a single rule with highest confidence, when predicting the class label of a new tuple.
- In experiments, CMAR had slightly higher average accuracy in comparison with CBA. Its runtime, scalability, and use of memory were found to be more efficient.

Classification based on Multiple ARs (CMAR)

- [WenminLi-ICDM01]
- Uses multiple CARs in the classification step

- Steps in CMAR:
  - Mine for CARs satisfying support and confidence thresholds
  - Sort all CARs based on confidence
  - Find all CARs which satisfy the given query
  - Group them based on their class label
  - Classify the query to the class whose group of CARs has the maximum ***weight***

Classification based on Multiple ARs (CMAR)

- [WenminLi-ICDM01]
- Uses multiple CARs in the classification step

- Steps in CMAR:
  - Mine for CARs satisfying support and confidence thresholds
  - Sort all CARs based on confidence
  - Find all CARs which satisfy the given query
  - Group them based on their class label
  - Classify the query to the class whose group of CARs has the maximum *weight*

CMAR CONTD.

R — Rules satisfying query *'q'*

Rules labeled c_1
$i_1 \rightarrow c_1$

Rules labeled c_2
$i_2, i_3 \rightarrow c_2$
$i_2, i_4 \rightarrow c_2$

$$weight = \sum_{r \in R} \left(\min\{\sup[r], \sup[c]\} - \frac{\sup[r] \times \sup[c]}{|T|}\right)^2 \times |T| \times e$$

Output the class with the highest weight

CMAR Disadvantages

- No proper **statistical explanation** given for the mathematical formulae that were employed

- Cannot handle **Fully Confident Associations**

CPAR

- CBA and CMAR adopt methods of frequent itemset mining to generate candidate association rules, which include all conjunctions of attribute–value pairs (items) satisfying minimum support. These rules are then examined, and a subset is chosen to represent the classifier. However, such methods generate quite a large number of rules. CPAR (Classification based on Predictive Association Rules) takes a different approach to rule generation, based on a rule generation algorithm for classification known as FOIL

- During classification, CPAR employs a somewhat different multiple rule strategy than CMAR.
- If more than one rule satisfies a new tuple, X, the rules are divided into groups according to class, similar to CMAR.
- However, CPAR uses the best k rules of each group to predict the class label of X, based on expected accuracy. By considering the best k rules rather than all of a group's rules, it avoids the influence of lower-ranked rules.
- CPAR's accuracy on numerous data sets was shown to be close to that of CMAR.
- However, since CPAR generates far fewer rules than CMAR, it shows much better efficiency with large sets of training data.