DATA ANALYTICS

Unit 4: Market Basket Analysis (Apriori Algorithm)

Jyothi R.
Department of Computer Science and
Engineering

DATA ANALYTICS

Market Basket Analysis(Apriori Algorithm)

- Table of Contents:
- Market Basket Analysis.
- The Approach (Apriori Algorithm)
- Handling and Readying the Dataset
- Structural Overview and Prerequisites
- Key terms and Usage
- Interpretations and Analysis
- The Item Frequency Histograms
- Graphical Representation
- Individual Rule Representation
- Interactive Scatterplot
- End Notes and Summary

DATA ANALYTICS

Market Basket Analysis

- The market-basket model of data is used to describe a common form of many- many relationship between two kinds of objects.
- On the one hand, we have items, and on the other we have baskets, sometimes called “transactions.”
- Each basket consists of a set of items (an itemset), and usually we assume that the number of items in a basket is small – much smaller than the total number of items.
- The number of baskets is usually assumed to be very large, bigger
- than what can fit in main memory.
- The data is assumed to be represented in a file consisting of a sequence of baskets.
- In terms of the distributed file system the baskets are the objects of the file, and each basket is of type “set of items.”

DATA ANALYTICS

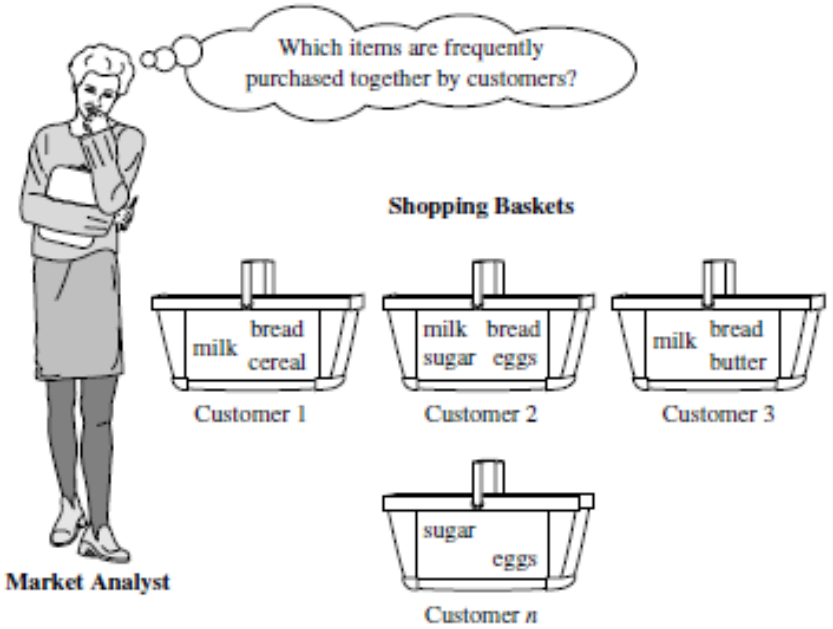
Market Basket Analysis

- Frequent itemset mining leads to the discovery of associations and correlations among items in large transactional or relational data sets.
- With massive amounts of data continuously being collected and stored, many industries are becoming interested in mining such patterns from their databases.
- The discovery of interesting correlation relationships among huge amounts of business transaction records can help in many business decision-making processes such as catalog design, cross-marketing, and customer shopping behavior analysis.

DATA ANALYTICS

Market Basket Analysis

- A typical example of frequent itemset mining is market basket analysis.
- This process analyzes customer buying habits by finding associations between the different items that
- customers place in their “shopping baskets” as shown in Figure : **Market Basket Analysis**



DATA ANALYTICS

Market Basket Analysis

- The discovery of these associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers.
- For instance, if customers are buying milk, how likely are they to also buy bread (and what kind of bread) on the same trip to the supermarket?
- This information can lead to increased sales by helping retailers do selective marketing and plan their shelf space.

DATA ANALYTICS

Market Basket Analysis

- Market basket analysis. Example:
- Suppose, as manager of an AllElectronics branch, you would like to learn more about the buying habits of your customers.
- Specifically, you wonder, “Which groups or sets of items are customers likely to purchase on a given trip to the store?”
- To answer your question, market basket analysis may be performed on the retail data of
- customer transactions at your store.
- You can then use the results to plan marketing or advertising strategies, or in the design of a new catalog.

DATA ANALYTICS

Market Basket Analysis

- Market basket analysis- Example:
- For instance, market basket analysis may help you design different store layouts. In one strategy, items that are frequently purchased together can be placed in proximity to further encourage the combined sale of such items.
- If customers who purchase computers also tend to buy antivirus software at the same time, then placing the hardware display close to the software display may help increase the sales of both items.

Market Basket Analysis

- In an alternative strategy, placing hardware and software at opposite ends of the store
- may entice customers who purchase such items to pick up other items along the way.
- For instance, after deciding on an expensive computer, a customer may observe security systems for sale while heading toward the software display to purchase antivirus software, and may decide to purchase a home security system as well.
- Market basket analysis can also help retailers plan which items to put on sale at reduced prices.
- If customers tend to purchase computers and printers together, then having a sale on printers may encourage the sale of printers as well as computers.

DATA ANALYTICS

Market Basket Analysis

- If we think of the universe as the set of items available at the store, then each item has a Boolean variable representing the presence or absence of that item.
- Each basket can then be represented by a Boolean vector of values assigned to these variables.
- The Boolean vectors can be analyzed for buying patterns that reflect items that are frequently associated or purchased together.
- These patterns can be represented in the form of association rules.
- For example, the information that customers who purchase computers also tend to buy antivirus software at the same time is represented in the following association rule:

$$computer \Rightarrow antivirus\_software\ [support = 2\%, confidence = 60\%].$$

DATA ANALYTICS

Market Basket Analysis

- Rule support and confidence are two measures of rule interestingness.
- They respectively reflect the usefulness and certainty of discovered rules.
- A support of 2% for Rule means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together.
- A confidence of 60% means that 60% of the customers who purchased a computer also bought the software.
- Typically, association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold.
- These thresholds can be a set by users or domain experts.
- Additional analysis can be performed to discover interesting statistical correlations between associated items.

DATA ANALYTICS

Frequent Itemsets, Closed Itemsets, and Association Rules

- A set of items is referred to as an **itemset**.
- An itemset that contains *k* items is a ***k*-itemset**.
- The set f*computer, antivirus softwar*eg is a 2-itemset.
- The **occurrence frequency of an itemset** is the number of transactions that contain the itemset.
- This is also known, simply, as the **frequency**, **support count**, or **count** of the itemset.

Frequent Itemsets, Closed Itemsets, and Association Rules

- Note that the itemset support defined is sometimes referred to as *relative support*,
- whereas the occurrence frequency is called the **absolute support**.
- If the relative support of an itemset *I* satisfies a prespecified **minimum support threshold** (i.e., the absolute support of *I* satisfies the corresponding **minimum support count threshold**), then *I* is a **frequent** itemset.
- The set of frequent *k*-itemsets is commonly denoted by L_k

Frequent Itemsets, Closed Itemsets, and Association Rules

$$support(A \Rightarrow B) = P(A \cup B)$$

$$confidence(A \Rightarrow B) = P(B|A).$$

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

Frequent Itemsets, Closed Itemsets, and Association Rules

- In confidence of rule equation A => B can be easily derived from the support counts of A and A∪B.
- That is, once the support counts of A, B, and A ∪B are found, it is straightforward to derive the corresponding association rules A =>B and B =>A and check whether they are strong.
- Thus, the problem of mining association rules can be reduced to that of mining frequent itemsets.

DATA ANALYTICS

Frequent Itemsets, Closed Itemsets, and Association Rules

- In general, association rule mining can be viewed as a two-step process:

1. Find all frequent itemsets: By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, min sup.
2. Generate strong association rules from the frequent itemsets: By definition, these rules must satisfy minimum support and minimum confidence.

Frequent Itemsets, Closed Itemsets, and Association Rules

- An itemset X is closed in a data set D if there exists no proper super-itemset Y such that Y has the same support count as X in D.
- An itemset X is a closed frequent itemset in set D if X is both closed and frequent in D. An itemset X is a maximal frequent itemset (or max-itemset) in a data set D if X is frequent, and there exists no super-itemset Y such that X ⊂ Y and Y is frequent in D.

Frequent Itemsets, Closed Itemsets, and Association Rules

- Let C be the set of closed frequent itemsets for a data set D satisfying a minimum support threshold, min sup.
- Let M be the set of maximal frequent itemsets for D satisfying min sup.
- Suppose that we have the support count of each itemset in C and M.
- Notice that C and its count information can be used to derive the whole set of frequent itemsets.

DATA ANALYTICS

Market Basket Analysis(Apriori Algorithm)

- The basic algorithm for finding frequent item sets.
- We look at how to generate strong association rules from frequent item sets.
- It describes several variations to the Apriori algorithm for improved efficiency and scalability.
- It presents pattern-growth methods for mining frequent item sets that confine the subsequent search space to only the data sets containing the current frequent item sets.

Apriori Algorithm: Finding Frequent Itemsets.

- Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules.
- The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties, as we shall see later.
- Apriori employs an iterative approach known as a level-wise search, where k-itemsets are used to explore .k C1/-itemsets. First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item, and

Apriori Algorithm: Finding Frequent Itemsets.

- Collecting those items that satisfy minimum support.
- The resulting set is denoted by L1.
- Next, L1 is used to find L2, the set of frequent 2-itemsets, which is used to find L3, and so on, until no more frequent k-itemsets can be found.
- The finding of each Lk requires one full scan of the database.

Apriori Algorithm: Finding Frequent Itemsets.

- To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property is used to reduce the search space.

Apriori Algorithm: Finding Frequent Itemsets.

Apriori property: All nonempty subsets of a frequent itemset must also be frequent.

The Apriori property is based on the following observation.

- By definition, if an itemset I does not satisfy the minimum support threshold, min sup, then I is not frequent, that is, P(I) < min sup.
- If an item A is added to the itemset I, then the resulting itemset (i.e., I UA) cannot occur more frequently than I.
- Therefore, I UA is not frequent either, that is, P(I U A) < min sup.
- This property belongs to a special category of properties called **antimonotonicity.**

Apriori Algorithm: Finding Frequent Itemsets.

- In the sense that if a set cannot pass a test, all of its supersets will fail the same test as well.
- It is called antimonotonicity because the property is monotonic in the context of failing a test.
- “How is the Apriori property used in the algorithm?” To understand this, let us look at
- how L_{k-1} is used to find L_k for k _>=2.
- A two-step process is followed, consisting of **join** and **prune** actions.

DATA ANALYTICS

Apriori Algorithm: Finding Frequent Itemsets.

1. **The join step**: To find L_k, a set of **candidate** k-itemsets is generated by joining L_{k-1} with itself.

- This set of candidates is denoted C_k. Let l_1 and l_2 be itemsets in L_{k-1}.
- The notation $l_i[j]$ refers to the jth item in l_i (e.g., $l_1[k-2]$ refers to the second to the last item in l_1).
- For efficient implementation, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order.
- For the (k -1)-itemset, l_i, this means that the items are sorted such that li[1] < li[2] < _ _ _ < li[k -1]. The join, L_{k-1} x L_{k-1}, is performed,
- where members of L_{k-1} are joinable if their first .k -2/ items are in common.

Apriori Algorithm: Finding Frequent Itemsets.

- That is, members l_1 and l_2 of L_{k-1} are joined if ($l_1[1]$ = $l_2[1]$)^(1[2] D l2[2])^_ _ _^($l_1[k-2] = l_2[k-2]$) ^($l_1[k-1] < l_2[k-1]$).
- The condition ($l_1[k-1] < l_2[k-1)$ simply ensures that no duplicates are generated. The resulting itemset formed by joining l_1 and l_2 is {$l_1[1]$, $l_1[2]$, : : : , $l_1[k-2]$, $l_1[k-1]$, $l_2[k-1]$}.

Apriori Algorithm: Finding Frequent Itemsets.

2. The prune step: C_k is a superset of L_k, that is, its members may or may not be frequent,

- But all of the frequent k-itemsets are included in C_k.
- A database scan to determine the count of each candidate in C_k would result in the determination of L_k (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to L_k).
- C_k, however, can be huge, and so this could involve heavy computation.

Apriori Algorithm: Finding Frequent Itemsets.

- To reduce the size of Ck, the Apriori property is used as follows.
- Any (k -1)-itemset that is not frequent cannot be a subset of a frequent k-itemset.
- Hence, if any (k -1)-subset of a candidate k-itemset is not in L_{k-1}, then the candidate cannot be frequent either and so can be removed from C_k.
- This **subset testing** can be done quickly by maintaining a hash tree of all frequent itemsets.

Apriori Algorithm: Finding Frequent Itemsets.

Apriori. Let's look at a concrete example, based on the AllElectronics transaction database, D, of Table 1.

- There are nine transactions in this database, that is, |D|= 9.
- We use Figure 1. to illustrate the Apriori algorithm for finding frequent itemsets in D.

1. In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemsets, C_1.

   The algorithm simply scans all of the transactions to count the number of occurrences of each item.

Apriori Algorithm: Finding Frequent Itemsets.

2. Suppose that the minimum support count required is 2, that is, *min sup* = 2.

(Here,we are referring to *absolute* support because we are using a support count.

The corresponding relative support is 2/9 = 22%.)

- The set of frequent 1-itemsets, L_1, can then be determined.
- It consists of the candidate 1-itemsets satisfying minimum support.
- In our example, all of the candidates in *C*1 satisfy minimum support.

Apriori Algorithm: Finding Frequent Itemsets.

3. To discover the set of frequent 2-itemsets, L_2, the algorithm uses the join $L1$ |X| $L1$ to generate a candidate set of 2-itemsets, $C2$. $C2$ consists of

$$\binom{|L_1|}{2}$$

2-itemsets. Note that no candidates are removed from $C2$ during the prune step because each subset of the candidates is also frequent.

Apriori Algorithm: Finding Frequent Itemsets.

4. Next, the transactions in D are scanned and the support count of each candidate itemset in C_2 is accumulated, as shown in the middle table of the second row in Figure 1.

5. The set of frequent 2-itemsets, L_2, is then determined, consisting of those candidate 2-itemsets in C_2 having minimum support.

Apriori Algorithm: Finding Frequent Itemsets.

6. The generation of the set of the candidate 3-itemsets, C3, is detailed in Figure 2.

- From the join step, we first get C3 = L2 |x| L2 = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5},
- {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}.
- Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that the four latter candidates cannot possibly be frequent.

Apriori Algorithm: Finding Frequent Itemsets.

- We therefore remove them from C3,
- Thereby saving the effort of unnecessarily obtaining their counts during the subsequent scan of D to determine L3.
- Note that when given a candidate k-itemset,
- we only need to check if its (k -1)-subsets are frequent
- since the Apriori algorithm uses a level-wise search strategy.
- The resulting pruned version of C_3 is shown in the first table of the bottom row of Figure 1.

Apriori Algorithm: Finding Frequent Itemsets.

7. The transactions in D are scanned to determine L_3, consisting of those candidate 3-itemsets in C_3 having minimum support (Figure 1).

8. The algorithm uses L_3 1 L_3 to generate a candidate set of 4-itemsets, C_4. Although the join results in {{I1, I2, I3, I5}}, itemset {I1, I2, I 3, I5} is pruned because its subset {I2, I3, I5} is not frequent. Thus, $C_4 = \mathbf{\Phi}$, and the algorithm terminates, having found all of the frequent itemsets.

Apriori Algorithm: Finding Frequent Itemsets.

Table 1.

Transactional Data for an *AllElectronics* Branch

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Apriori Algorithm: Finding Frequent Itemsets.

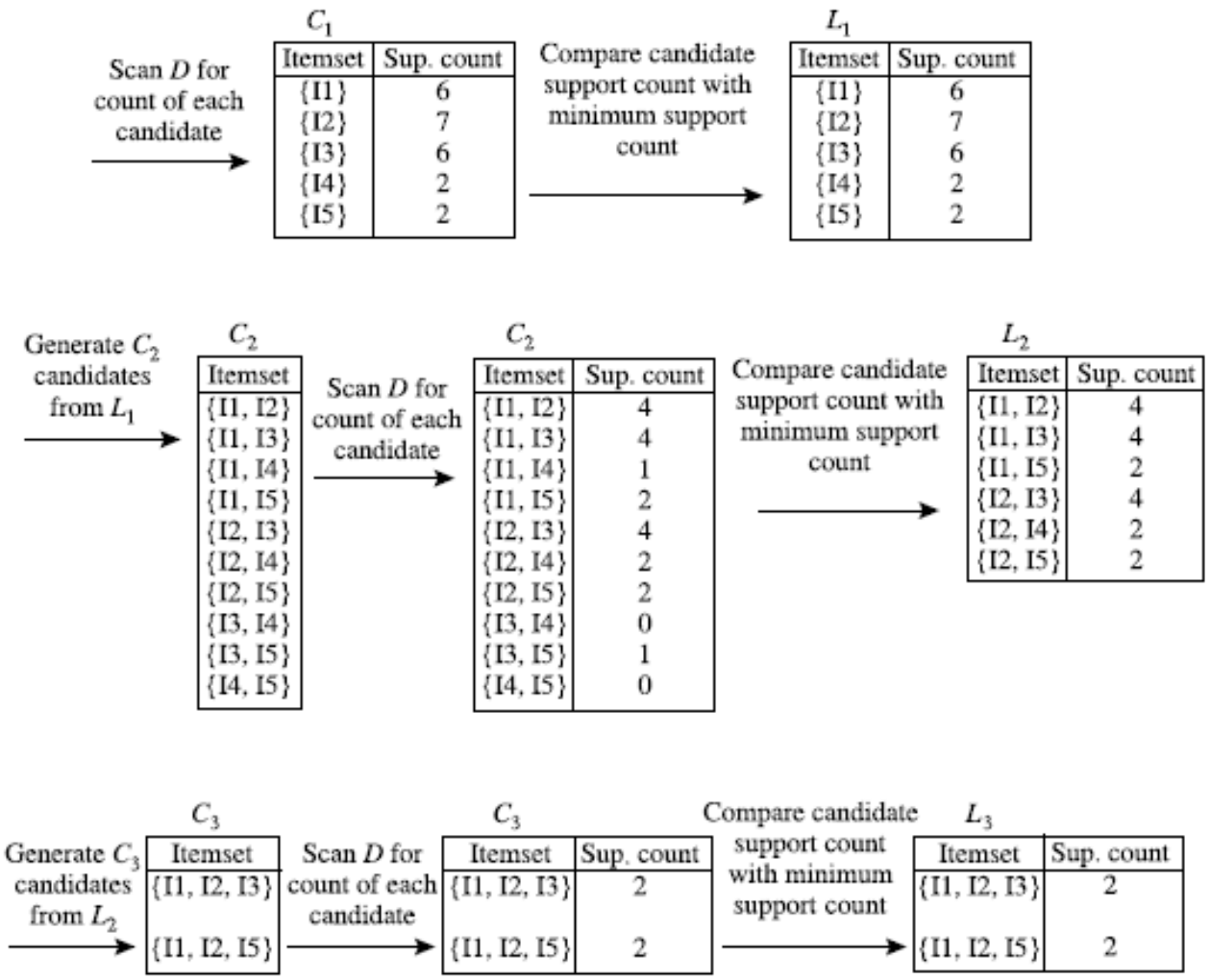


Figure 1.Generation of the candidate itemsets and frequent itemsets, where the minimum support count is 2.

DATA ANALYTICS

Apriori Algorithm: Finding Frequent Itemsets.

(a) Join: $C_3 = L_2 \bowtie L_2 = \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$\bowtie \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$= \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}.$

(b) Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent?

- The 2-item subsets of {I1, I2, I3} are {I1, I2}, {I1, I3}, and {I2, I3}. All 2-item subsets of {I1, I2, I3} are members of L_2. Therefore, keep {I1, I2, I3} in C_3.
- The 2-item subsets of {I1, I2, I5} are {I1, I2}, {I1, I5}, and {I2, I5}. All 2-item subsets of {I1, I2, I5} are members of L_2. Therefore, keep {I1, I2, I5} in C_3.
- The 2-item subsets of {I1, I3, I5} are {I1, I3}, {I1, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I1, I3, I5} from C_3.
- The 2-item subsets of {I2, I3, I4} are {I2, I3}, {I2, I4}, and {I3, I4}. {I3, I4} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I4} from C_3.
- The 2-item subsets of {I2, I3, I5} are {I2, I3}, {I2, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I5} from C_3.
- The 2-item subsets of {I2, I4, I5} are {I2, I4}, {I2, I5}, and {I4, I5}. {I4, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I4, I5} from C_3.

(c) Therefore, $C_3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$ after pruning.

Figure 2. Generation and pruning of candidate 3-itemsets, C_3, from L_2 using the Apriori property.

.

DATA ANALYTICS

Apriori Algorithm: Finding Frequent Itemsets.

Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- min_sup, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
(1)   L1 = find_frequent_1-itemsets(D);
(2)   for (k = 2; Lk−1 ≠ ϕ; k++) {
(3)       Ck = apriori_gen(Lk−1);
(4)       for each transaction t ∈ D { // scan D for counts
(5)           Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)           for each candidate c ∈ Ct
(7)               c.count++;
(8)       }
(9)       Lk = {c ∈ Ck|c.count ≥ min_sup}
(10)  }
(11)  return L = ∪k Lk;

procedure apriori_gen(Lk−1:frequent (k − 1)-itemsets)
(1)   for each itemset l1 ∈ Lk−1
(2)       for each itemset l2 ∈ Lk−1
(3)           if (l1[1] = l2[1]) ∧ (l1[2] = l2[2])
                  ∧... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
(4)               c = l1 ⋈ l2; // join step: generate candidates
(5)               if has_infrequent_subset(c, Lk−1) then
(6)                   delete c; // prune step: remove unfruitful candidate
(7)               else add c to Ck;
(8)           }
(9)   return Ck;

procedure has_infrequent_subset(c: candidate k-itemset;
          Lk−1: frequent (k − 1)-itemsets); // use prior knowledge
(1)   for each (k − 1)-subset s of c
(2)       if s ∉ Lk−1 then
(3)           return TRUE;
(4)   return FALSE;
```

Figure 3.Apriori algorithm for discovering frequent itemsets for mining Boolean association rules.

DATA ANALYTICS

Apriori Algorithm: Finding Frequent Itemsets.

(a) Join: $C_3 = L_2 \bowtie L_2 = \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$\bowtie \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$= \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}.$

(b) Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent?

- The 2-item subsets of {I1, I2, I3} are {I1, I2}, {I1, I3}, and {I2, I3}. All 2-item subsets of {I1, I2, I3} are members of L_2. Therefore, keep {I1, I2, I3} in C_3.
- The 2-item subsets of {I1, I2, I5} are {I1, I2}, {I1, I5}, and {I2, I5}. All 2-item subsets of {I1, I2, I5} are members of L_2. Therefore, keep {I1, I2, I5} in C_3.
- The 2-item subsets of {I1, I3, I5} are {I1, I3}, {I1, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I1, I3, I5} from C_3.
- The 2-item subsets of {I2, I3, I4} are {I2, I3}, {I2, I4}, and {I3, I4}. {I3, I4} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I4} from C_3.
- The 2-item subsets of {I2, I3, I5} are {I2, I3}, {I2, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I5} from C_3.
- The 2-item subsets of {I2, I4, I5} are {I2, I4}, {I2, I5}, and {I4, I5}. {I4, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I4, I5} from C_3.

(c) Therefore, $C_3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$ after pruning.

Figure Generation and pruning of candidate 3-itemsets, C_3, from L_2 using the Apriori property.

.

DATA ANALYTICS

Apriori Algorithm: Finding Frequent Itemsets.

Figure 3. shows pseudocode for the Apriori algorithm and its related procedures.

- Step 1 of Apriori finds the frequent 1-itemsets, $L1$.
- In steps 2 through 10, L_{k-1} is used to generate candidates C_k to find L_k for k _ 2.
- The apriori gen procedure generates the candidates and
- Then uses the Apriori property to eliminate those having a subset that is not frequent (step 3).

Apriori Algorithm: Finding Frequent Itemsets.

- Once all of the candidates have been generated, the database is scanned (step 4).
- For each transaction, a subset function is used to find all subsets of the transaction that are candidates (step 5), and
- The count for each of these candidates is accumulated (steps 6 and 7).
- Finally, all the candidates satisfying the minimum support (step 9) form the set of frequent itemsets, *L* (step 11).

Generating Association Rules from Frequent Itemsets

- Once the frequent itemsets from transactions in a database D have been found,
- it is straightforward to generate strong association rules from them
- where strong association rules satisfy both minimum support and minimum confidence.
- This can be done using Eq. 1. for confidence, which we show again here for completeness:

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

Generating Association Rules from Frequent Itemsets

- The conditional probability is expressed in terms of itemset support count,
- Where support_count (AUB) is the number of transactions containing the itemsets (AUB),
- And support count(A) is the number of transactions containing the itemset A.

Apriori Algorithm: Finding Frequent Itemsets.

- Based on this equation, association rules can be generated as follows:

1. For each frequent itemset l, generate all nonempty subsets of l.
2. For every nonempty subset s of l, output the rule "$s \Rightarrow (l - s)$" if $\frac{support\_count(l)}{support\_count(s)} \geq$ = min conf, where min conf is the minimum confidence threshold.

- Because the rules are generated from frequent itemsets, each one automatically satisfies the minimum support.
- Frequent itemsets can be stored ahead of time in hash tables along with their counts so that they can be accessed quickly.

References

Text Book:

"Business Analytics, The Science of Data-Driven Making", U. Dinesh Kumar, Wiley 2017

"Recommender Systems, The text book, Charu C. Aggarwal, Springer 2016 Section 1.and Section 2.

THANK YOU

Jyothi R.
Assistant Professor,
Department of Computer Science
jyothir@pes.edu