Lesson 9 Association Rule Mining

Outline

- Frequent Pattern Analysis
- Association Rule Mining - Concepts
- Apriori algorithm
- Mining Groceries dataset using R Apriori algorithm

Frequent Pattern Analysis

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
- Applications
  - Market basket analysis, catalog design, sale campaign analysis, and Web log (click stream) analysis.

Association Rule Mining

- The result of a market basket analysis is a collection of association rules that specify patterns found among the items.
- Input: the transaction data
- Output: Frequent patterns/association rules among items
- Example: According to the transaction data, we find this pattern:

  "Customer who bought a laptop computer and a virus protection software, also bought extended service plan 70 percent of the time."
- How do you use such a pattern/knowledge?
  - Put the items next to each other for ease of finding
  - Promote the items as a package (do not put one on sale if the other(s) are on sale)
  - Place items far apart from each other so that the customer has to walk the aisles to search for it, and by doing so potentially seeing and buying other items

Basic Concepts: Itemsets

- itemset: A set of one or more items
- k-itemset: a set of k items
  - 1-itemsets: {beef}, {milk}, {cheese}
  - 2-itemsets: {beef, chicken}, {chicken, milk}
  - 3-itemsets: {beef, chicken, cheese}

TID	Items bought
100	Beef, Chicken, Milk
200	Beef, Cheese
300	Cheese, Boots
400	Beef, Chicken, Cheese
500	Beef, Chicken, Clothes, Cheese, Milk
600	Chicken, Clothes, Milk
700	Chicken, Milk, Clothes

Basic Concepts: Support

- *Absolute support*, or, support count of X: Frequency or the number of transactions that contain itemset X
- *Relative support:* The fraction of transactions that contain X (i.e., the probability that a transaction contains X)
- When we say "support" it could mean either absolute or relative support. So, interpret it in the context.
- In previous example:
  - Support of {chicken}: 5 (count) or 5/7 = 71.4% (percent)
  - Support of {chicken, milk}: 4 (count) or 5/7 = 57.1% (percent)

Basic Concepts: Frequent Itemsets

- An itemset X is *frequent* if X's support is no less than a predefined minimum support threshold, called *minsup*.
- If minsup = 0.6 or 60%
  - {chicken} is frequent, or is a frequent itemset/pattern
  - {chicken, milk} is not frequent, or is not a frequent itemset/pattern

Basic Concepts: Association Rules

- An *association rule* has the form R1: X -> Y, where X and Y are itemsets. The rule says "whenever X occurs, we expect to find Y as well".
- Support of R1:

  $s(R1) = support(X \cup Y)$

  or probability that a transaction contains X and Y
- Confidence of R1:

  $c(R1) = support(X \cup Y) / support(X)$,

  or conditional probability that a transaction having X also contains Y

The confidence of a rule tells us how reliable the rule is. The closer the value c(R1) comes to 1, the more reliable the rule R1 is.

Basic Concepts: Association Rules

R1: {chicken} -> {milk}

- *s* (R1) = support({chicken, milk}) = 4 (count)
- *c* (R1) = support({chicken, milk}) / support({chicken})
  = 4 /5 = 0.8 or 80%

Among those who purchased chicken, 80% of them also purchased milk.

TID	Items bought
100	Beef, Chicken, Milk
200	Beef, Cheese
300	Cheese, Boots
400	Beef, Chicken, Cheese
500	Beef, Chicken, Clothes, Cheese, Milk
600	Chicken, Clothes, Milk
700	Chicken, Milk, Clothes

Basic Concepts: Strong Rules

- A *strong rule* is a rule X -> Y with the following two properties:
  - support ≥ minimum support (minsup) and
  - confidence ≥ minimum confidence (minconf).

  The values for minsup and minconf are initial values that are set based on past experience and other empirical data.
- Example: Let minsup = 20%, minconf = 50%

  Then,

  {chicken} -> {milk} (support = 57.1%, confidence = 80%), is a strong rule

  {chicken} -> {cheese} (support = 28.6%, confidence = 40%), is not a strong rule

Many mining algorithms

- Mining algorithms are used to discover a set of association rules
- 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 confidence, the mining algorithms return all the strong association rules in T.
- We study only one in this course: the Apriori Algorithm.

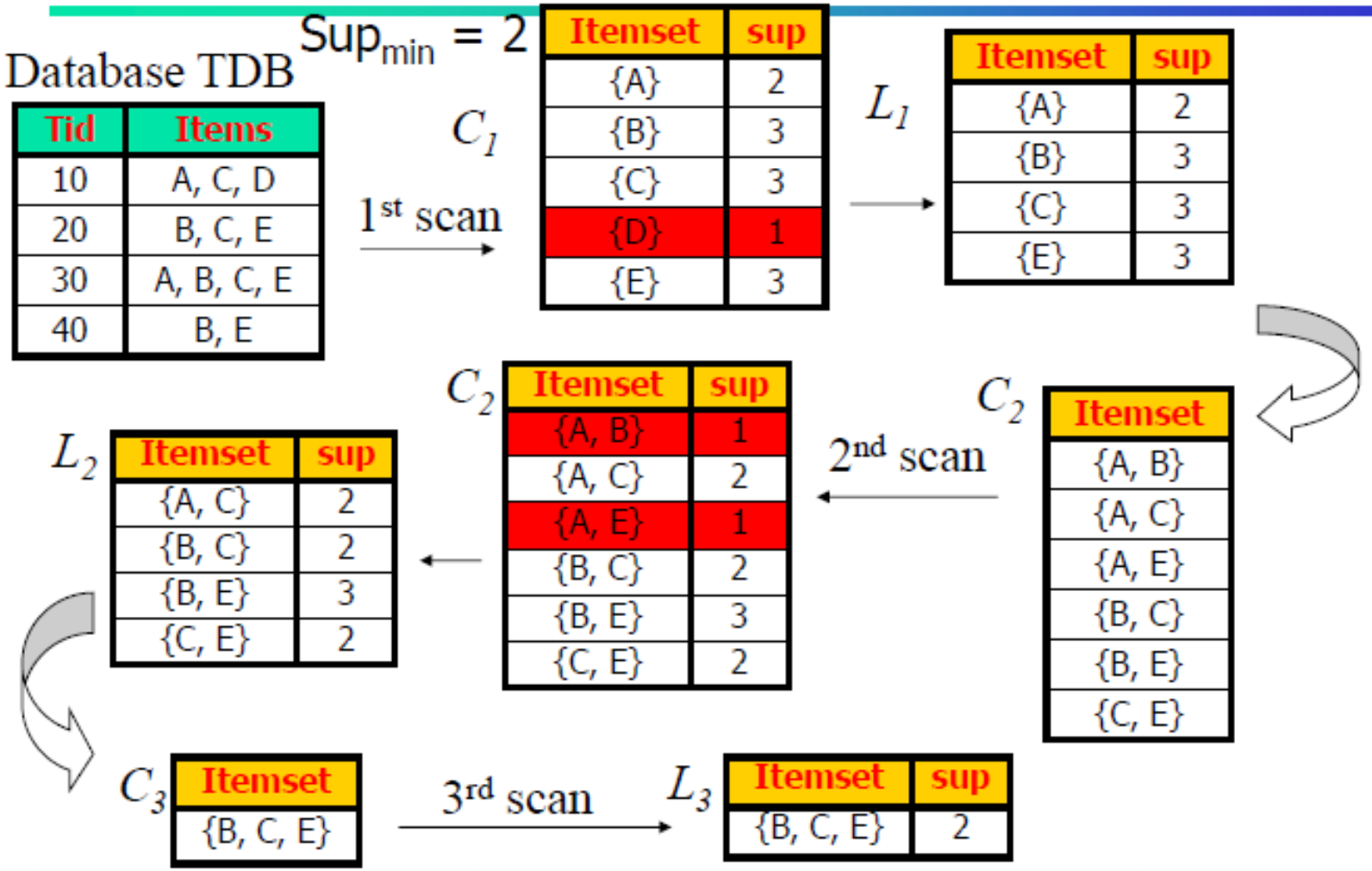
The Apriori Algorithm

- Two step process:
  - First, mine all frequent patterns. A frequent pattern is also called a frequent itemset or a large itemset
  - Second, mine strong rules from frequent itemsets

Step 1: Mining all frequent itemsets

- Find the frequent itemsets: the sets of items that have minimum support
  - The apriori property (downward closure property): A subset of a frequent itemset must also be a frequent itemset
    - i.e., if {ABC} is a frequent itemset, {AB} , {BC} and {AC} should be frequent itemsets
    - It can be also stated as: If an itemset contains a subset that is not frequent, then the itemset can never be frequent.
  - Iteratively find frequent itemsets with cardinality from 1 to k (k-itemset)

Worked Example - Step 1



The Apriori Algorithm: Pseudo-code

Remember: C_k: Candidate itemsets of size k

L_k: frequent itemsets of size k

C_1 <- generate Candidate itemsets of size 1

L_1 <- generate frequent itemsets of size 1 from C_1

k <- 1

while L_k != ∅ **do**

    C_{k+1} <- candidate-gen(L_k) //join and prune steps

    for each transaction t in database **do**

        increment the count of all candidates in C_{k+1} that are contained in t

    L_{k+1} <- candidates in C_{k+1} with min_support

return $\cup_k L_k$

Apriori Candidate Generation

- The *candidate-gen function* takes L_k and returns a candidate set C_{k+1} of (k+1)-itemsets. It has two steps
  - join step: Generate all possible candidate itemsets of size k+1
    - When joining two k-itemsets, we join only if the first k-1 items are identical. (Notice that itemsets are in sorted order)
  - prune step: Remove those candidates in C_{k+1} that cannot be frequent. (Using downward closure property)

Apriori Candidate Generation

- L_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, 3, 4, 5}}
- After pruning:
  - C_4 = {{1, 2, 3, 4}}
    because {1, 4, 5} is not in L_3
    ({1, 3, 4, 5} is removed)

Step 2: Generating rules from frequent itemsets

- Frequent itemsets are not the same as association rules!
- One more step is needed to generate association rules
- For each frequent itemset I,

  and for each proper nonempty subset X of I,
  - Let Y = I - X
  - $X \rightarrow Y$ is a strong rule if
    - Confidence($X \rightarrow Y$) $\geq$ minconf,

      (remember confidence($X \rightarrow Y$) = support($X \cup Y$) / support(X))
  - Note: definition of proper nonempty subset X of I: X is a subset of I, X is not empty and X!=I.

Worked Example - Step 2

- Continued with our example from step 1 where we found {B, C, E} is frequent, with minsup=2
  - Proper nonempty subsets: {B, C}, {B, E}, {C, E}, {B}, {C}, {E}
  - These generate the following association rules:
    - R1: {B,C} $\rightarrow$ {E}, confidence=2/2=100%
    - R2: {B,E} $\rightarrow$ {C}, confidence=2/3=67%
    - R3: {C,E} $\rightarrow$ {B}, confidence=2/2=100%
    - R4: {B} $\rightarrow$ {C,E}, confidence=2/3=67%
    - R5: {C} $\rightarrow$ {B,E}, confidence=2/3=67%
    - R6: {E} $\rightarrow$ {B,C}, confidence=2/3=67%
  - Suppose we set minconf = 80%, only R1 and R3 are strong rules.
- Need to perform this step for all frequent itemsets found in L_k (k =2, 3,…)

Generating Rules

- To recap, in order to obtain strong rules $X \rightarrow Y$, we need to compute the confidence of the rules, in other words, computed support($X \cup Y$) and support(X).
- All the required information for confidence computation has already been recorded in frequent itemset generation. No need to see the data any more.
- This step is not as time-consuming as frequent itemsets generation.

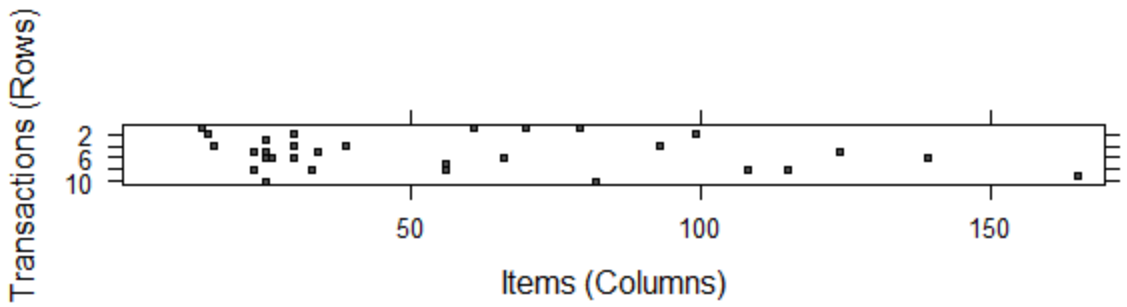
R code – Data Preparation

- How to store the transaction data?
  - We can create matrix - each row in the matrix indicates a transaction; each column indicates an item that appears in the data.
  - m[i][j] = 1 indicates for item j appear in the ith transaction;
  - m[i][j] = 0 indicates otherwise.
- Notice that most the cells contains 0 (non-meaning value)-A matrix like this is called a sparse matrix.
  - A sparse matrix does not store the full matrix in memory; it only stores the cells that are occupied by an item. This is more memory efficient. To create a sparse matrix
    - grocery <- read.transactions(".\\demoData\\grocery.csv", sep = ",")
    - More about a sparse matrix: https://en.wikipedia.org/wiki/Sparse_matrix

R code – Data Exploration

- R has data set called Groceries (in sparse matrix format) which contains 9835 transactions in a month and 169 items. We will use this data set for our analysis.

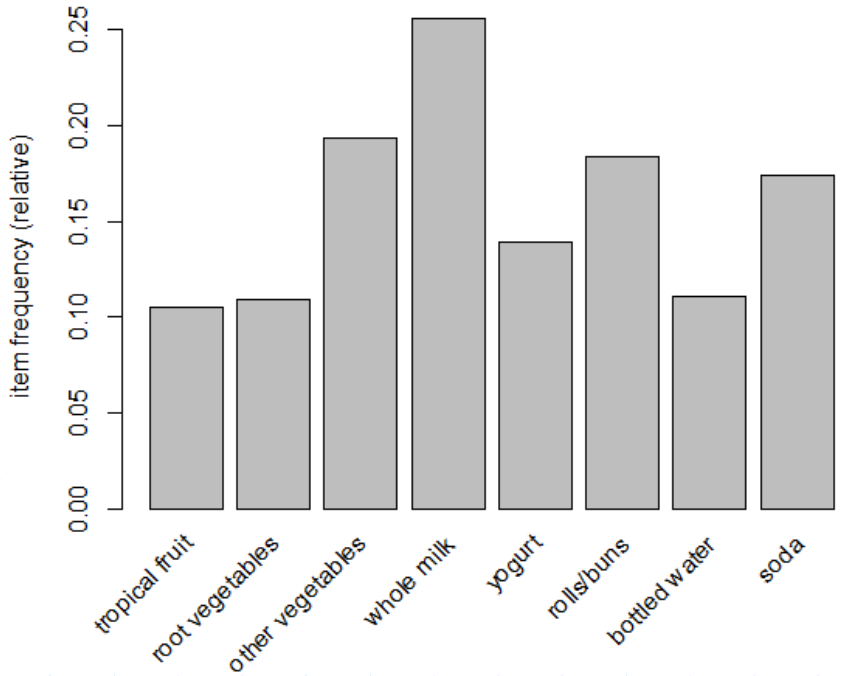
```
summary(Groceries)
#look at first five transactions
inspect(Groceries[1:5])
#visualize the first 10 rows of sparse matrix
image(Groceries[1:10])
```



R code – Data Exploration

- Generate item frequency plots use the itemFrequencyPlot() function. This method produces a bar chart depicting the proportion of transactions containing certain items.
- #plot frequent items with min support = 0.1

  itemFrequencyPlot(Groceries, support = 0.1)



R code – Training a model on the data

- apriori() function is straightforward, but there can sometimes be a fair amount of trial and error needed to find the support and confidence parameters that produce a reasonable number of association rules.

```
rules <- apriori(Groceries,
                 parameter = list(support = 0.006,
                 confidence = 0.25, minlen = 2))
summary(rules)
```

  - set of 463 rules
- Why support = 0.006? The assumption is if an item is purchased twice a day(about 60 times in a month of data), it may be an interesting pattern. 60/9835 is about 0.006.

R code – Inspecting rules

```
> inspect(rules[1:5])
    lhs                 rhs                  support     confidence lift     count
[1] {pot plants}     => {whole milk}       0.006914082 0.4000000  1.565460 68
[2] {pasta}          => {whole milk}       0.006100661 0.4054054  1.586614 60
[3] {herbs}          => {root vegetables}  0.007015760 0.4312500  3.956477 69
[4] {herbs}          => {other vegetables} 0.007727504 0.4750000  2.454874 76
[5] {herbs}          => {whole milk}       0.007727504 0.4750000  1.858983 76
```

Lift Formula

- The lift of a rule measures how much more likely one itemset is purchased with another itemset compared to its typical rate of purchase.
- We develop the Lift Formula by considering an example.

Lift Example – Problem 1

Suppose among 1000 transactions, milk is purchased 50 times, {milk, bread} are purchased together 10 times. This means that

support({milk}) = 50/1000 = 5% support({milk,bread}) = 10/1000 = 1%

confidence(milk->bread) = 1/5 = 20%

Problem 1: Assuming bread is purchased 300 times among the 1000 transactions, does the rule milk->bread tell us that it is more likely that a bread purchase will happen if milk is purchased than just an ordinary bread purchase?

Solution: Compare rate of bread purchase when milk is purchased to rate of bread purchase without assuming milk:

with milk: 10/50 = 20% = confidence(milk -> bread)

without assuming milk is purchased:

300/1000 = 30% = support({bread})

Conclusion: People actually buy less bread when buying milk; the rule indicates a negative correlation. Notice in this case 20% < 30%, i.e., 20%/30% < 1.

(Negative correlation means X and Y occur together less frequently than would happen by chance.)

Lift Example – Problem 2

Problem 2: Assuming bread is purchased 80 times among the 1000 transactions, does the rule milk->bread tell us that it is more likely that a bread purchase will happen if milk is purchased than just an ordinary bread purchase?

Solution: Compare rate of bread purchase when milk is purchased to rate of bread purchase without assuming milk:

with milk: 10/50 = 20% = confidence(milk -> bread) [same as before]

without assuming milk is purchased: 80/1000 = 8%

Conclusion: People buy bread at a higher rate when buying milk; the rule indicates a positive correlation. Notice in this case 20% > 8%,

i.e., 20%/8% > 1.

(Positive correlation means X and Y occur together more frequently than would happen by chance.)

Lift Formula

To see if a rule X->Y indicates that Y is purchased more frequently when X is purchased than otherwise, compare confidence(X->Y) to support(Y). This gives the Lift Formula:

$$lift(X \rightarrow Y) = \frac{confidence(X \rightarrow Y)}{support(Y)}$$

As in the examples, lift is used in the following way:

- lift > 1: positively correlated
- lift < 1: negatively correlated,
- lift = 1: independent (no correlation)

Are all rules mined useful?

- Does {pot plants} ->{whole milk} seem like a very useful rule?
  - Answer: Probably not, as there does not seem to be a logical reason why someone would be more likely to buy milk with a potted milk. Yet our data suggests otherwise.
  - How can we make sense of this fact?

Are all rules mined useful?

- A common approach is to take the association rules and divide them into the following three categories:
  - Actionable: the goal of a market basket analysis is to find such rules that provide a clear and useful insight.
  - Trivial: include any rules that are so obvious that they are not worth mentioning – clear but not useful. For example {diapers} -> {formula}
  - Inexplicable: Rules are inexplicable if the connection between the items is so unclear that figuring out how to use the information is impossible or nearly impossible. The rule may simply be a random pattern in the data.
- How do we know if a rule belongs to one of the three categories?
  - We may not be the best judge. Turn to subject matter experts.

R code – sorting and taking subsets of rules

```
#get top five highest lift rules
inspect(sort(rules, by="lift")[1:5])

#find subset of the rules with berrries appearing in the rule
sub.rules <- subset (rules, items %in% "berries")
inspect(sub.rules)
```

Summary

- Association rules are frequently used to find useful insights in the massive transaction databases of large retailers. As an unsupervised learning process, association rule learner are capable of exacting knowledge from large databases without any prior knowledge of what patterns to seek.
- It takes some effort to reduce the wealth of information into a smaller and more manageable set of results. The Apriori algorithm, does so by setting minimum thresholds of interestingness, and reporting only the associations meeting these criteria.