Association Rule Mining

Introduction *46*

Study core *46*

1 Association Rule Mining: Motivation and Main Concepts 46
2 Apriori Algorithm 47
3 FP-Growth Algorithm 47
4 Assignment Bundle: Frequent Itemset Mining 53

Learning Unit 7

Association Rule Mining

INTRODUCTION

The seventh learning unit introduces students to association rule mining. This unit has the following learning objectives:

LEARNING OBJECTIVES

After studying this unit students should understand the

- general concept of association rule mining
- concepts of support, lift and confidence in a rule
- Apriori algorithm for association rule mining
- FP-growth algorithm for association rule mining
- use of RapidMiner in association rule mining.

Study hints
This learning unit takes two weeks; it should take 19 hours to complete. Part of the work is theoretical in nature and involves reading Provost, pages 289–291. A more detailed discussion concerning the Apriori and FP-growth algorithms is then provided in this chapter of the workbook. Finally, students are required to complete the exercises in the assignment bundle.

A presentation given by the lecturer will summarise the most important points and provide examples in RapidMiner concerning the analysis of data using association rule mining algorithms. Students should dedicate about 9 hours to studying in the first week and 10 hours in the second week.

STUDY CORE

1 Association Rule Mining: Motivation and Main Concepts

Motivation

Association rule mining (ARM) is a rather interesting technique since it allows data analysts to find a relationship between attributes of interest in a dataset. Companies very often use ARM for the following tasks:

- Market Basket Analysis
- Word Net Analysis
- Personalisation and customization of a service

association rule

First of all, an *association rule* takes the following form:

$$LHS \rightarrow RHS$$

LHS means 'left hand side' and RHS means 'right hand side'. The LHS is usually an item, set of items or set of attributes, while the RHS is also an item, set of items or set of attributes. Stated otherwise, the LHS implies the RHS, or rather 'if LHS, then RHS'.

A concrete example of an association rule could be:

$$\{milk, bread, tomatoes\} \rightarrow \{eggs\}$$

This can be read as 'if the customer buys milk, bread and tomatoes, then it is highly likely that eggs will be bought too'.

Support, confidence

In order to select interesting rules from the dataset, the algorithms use two important metrics: *support* and *confidence*. Support is a number between 0 and 1 and indicates how frequently that particular rule is true in the dataset. Confidence is also a number between 0 and 1 and represents how many times the rule has been found to be true.

Lift

Another important concept is the concept of *lift*, that is a number ≥ 1. Unfortunately, there is a complex probabilistic explanation behind the lift concept. In our case though it is simply important to know that the lift represents the degree to which the attributes occur independently from each other, and that a value > 1 implies that there is some sort of dependency. A value of 1 implies that these attributes are independent and no rule can be created between the two attributes or items.

2 Apriori Algorithm

Apriori

itemset

This section introduces an informal description of the Apriori algorithm presented in [1]. The *Apriori* algorithm was designed to work on transactions to identify which items occur simultaneously most often. Here, each of the transactions considered is expected to be a set of *items (itemset).* To determine a relationship to be interesting, the algorithm defines a threshold **T** as the number of transactions in a database in which an itemset has to appear in order to be considered interesting.

Apriori uses an approach that begins by checking items of one element against the transactions in the database and then including further elements in the basket. If an itemset is found to be irrelevant with respect to the threshold **T**, the itemset is discarded to pass on to the next itemset in a breadth first search. At the end of the algorithm, the set of itemsets that pass the threshold **T** check are returned to the user.

In order to further prune the returned itemset (since this may contain a large number of items if only **T** is defined) the user can also specify a support ε (support as defined in Section 1 above in this chapter) stating the percentage of transactions in which the itemset has to present itself in order to consider it a relevant association rule.

limitations

Due to its simplicity, despite being historically relevant the Apriori algorithm has a number of *limitations*. The two most relevant limitations are that it generates a large number of subsets and that its breadth first traversing strategy takes a very long time to traverse the entire database. These limitations led researchers to look into more efficient algorithms for association rule mining, one of which is the FP-Growth algorithm introduced in the next section (also available in RapidMiner).

2 FP-Growth Algorithm

Frequent pattern growth

This section introduces, again in informal fashion, the FP-growth algorithm [2]. FP-growth stands for '*frequent pattern growth*'. FP-Growth improves upon the Apriori algorithm quite significantly. The major improvement to Apriori is particularly related to the fact that the FP-growth algorithm only needs two passes on a dataset.

FP-tree

- In the first step, the algorithm builds a compact data structure called the *FP-tree.*
- In the second step, the algorithm builds frequent itemsets directly from the FP-tree.

Item frequency parameter φ

In addition, the algorithm has the parameter φ, which is user defined, to define a threshold for which items are considered as frequent.
As an illustrative example, consider the following dataset of transactions:

Transaction ID	Items
1	{apricots, bread}
2	{bread, carrots, dumplings}
3	{apricots, carrots, dumplings, eggs}
4	{ apricots, dumplings, eggs}
5	{ apricots, bread, carrots }
6	{ apricots, bread, carrots, dumplings}
7	{ apricots }
8	{ apricots, bread, carrots}
9	{ apricots, bread, dumplings }
10	{ bread, carrots, eggs}

Considering the first transaction, for example, it means that apricots and bread were bought together.

For this dataset, with regard to step 1, the FP-tree is constructed as follows:

1. Read the first transaction {apricots, bread}.
    a. Create a root node.
    b. Create a path root → apricots → bread and set their counts to 1.
2. Read the second transaction {bread, carrots, dumplings}.
    a. Create a path root → bread → carrot → dumplings.
    b. Set their count to 1.
    c. Transactions 1 and 2 share bread, but their path is distinct because they do not have the same prefix.
    d. Link bread in the first path with bread in the second path.
3. Read the third transaction {apricots, carrots, dumplings, eggs}.
    a. Since this path shares apricots with the first path, set the count of apricots to 2, then add nodes carrots, dumplings and eggs after the apricots node.
    b. Add links between carrots and dumplings.

This is continued in the same fashion until the entire dataset has been mapped. Figure 1 shows a depiction of the algorithm used to create the FP-tree.

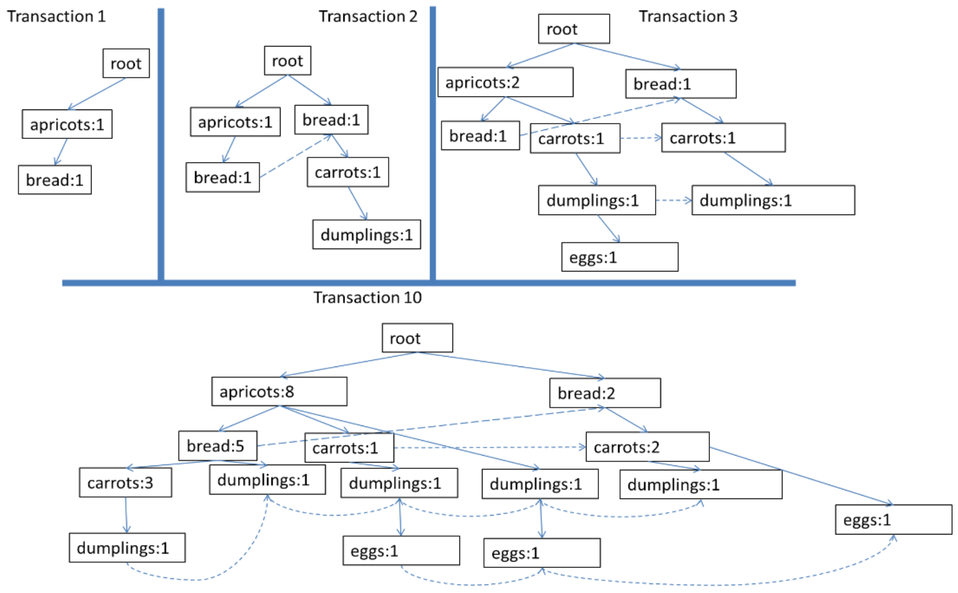


Figure 1: FP-tree building.

Prefix path subtree

After creating the FP-tree, the algorithm proceeds to identify the frequent itemsets by using the FP-tree. In order to do this, the algorithm takes each of the items and checks the entire prefix path subtree preceding the items. For the eggs item, the prefix path subtree is as shown in Figure 2. Informally speaking, the prefix path subtree is nothing more than the part of the tree above the eggs items.

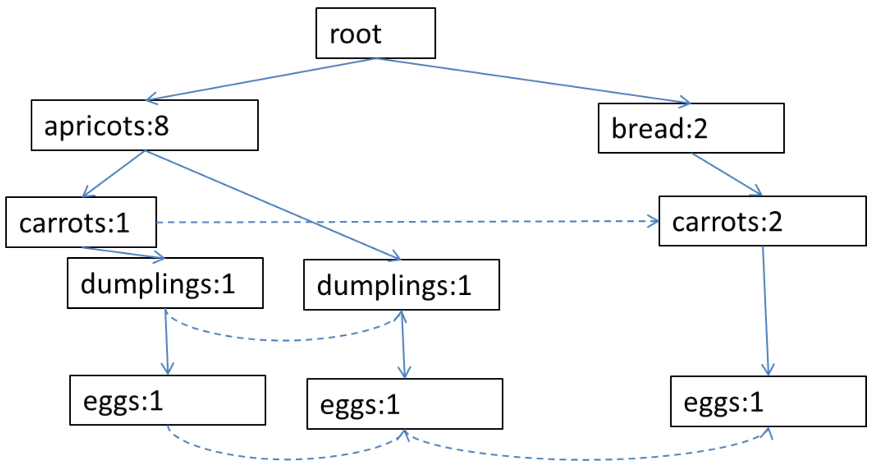


Figure 2: Prefix path subtree for the eggs item.

Now it must be checked whether eggs are a frequent item. To do so count the occurrences of the eggs item in the subtree (along the dotted lines) and sum the frequencies. In this case the count = 3. If the user defined parameter $\varphi \geq 2$ then {eggs} are considered a frequent item. Now that eggs have been extracted as a frequent item, use the prefix path subtree to find all the itemsets ending in {eggs}. See Figure 3 for an example of how FP-Growth creates these itemsets.

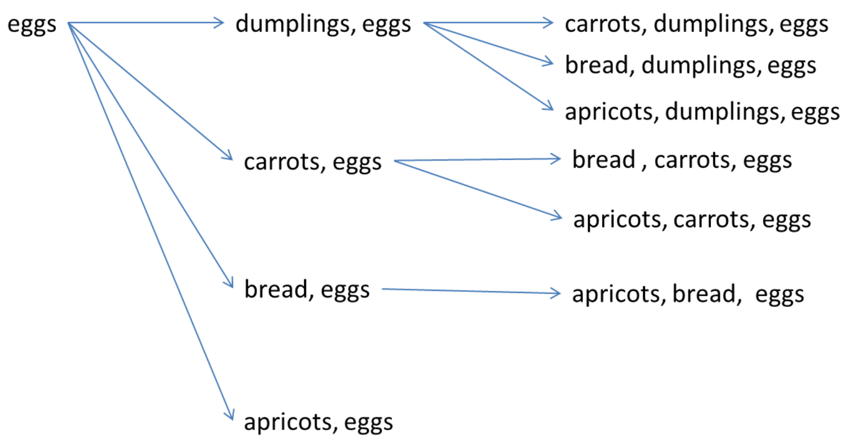
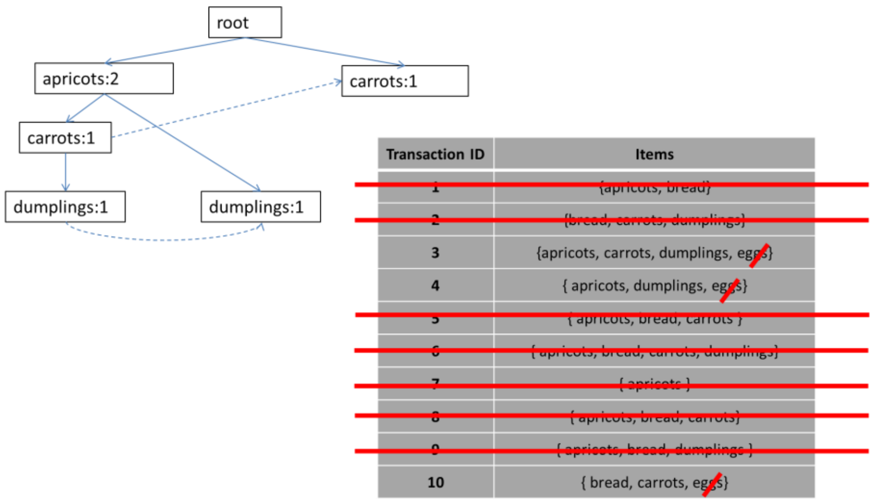


Figure 3: Frequent itemsets evaluation ending in eggs.

Conditional FP-tree

The main issue now is to identify which of these itemsets are more relevant. In order to do this FP-growth uses the prefix path subtree to create an FP-tree around the eggs item. This is called a *conditional FP-tree* (for the eggs item). To do this, the eggs item is removed from the prefix path sub tree. This is done because eggs are no longer needed. The goal is to see what happens in the prefix of the transactions, or to see what is frequent simultaneously with the eggs item. Figure 4 illustrates such a tree for the eggs item.



Transaction ID	Items
~~1~~	~~{apricots, bread}~~
~~2~~	~~{bread, carrots, dumplings}~~
3	{apricots, carrots, dumplings, ~~eggs~~}
4	{ apricots, dumplings, ~~eggs~~}
~~5~~	~~{ apricots, bread, carrots }~~
~~6~~	~~{ apricots, bread, carrots, dumplings}~~
~~7~~	~~{ apricots }~~
~~8~~	~~{ apricots, bread, carrots}~~
~~9~~	~~{ apricots, bread, dumplings }~~
10	{ bread, carrots, ~~eggs~~}

Figure 4: Conditional FP-Tree for the eggs item.

The interesting thing is that at this point it is only necessary to count. So now the count = 2 for dumplings, so {dumplings, eggs} may be called a frequent itemset. Similarly, {apricots, dumplings, eggs} is also a frequent itemset. To find out whether the {carrots, eggs} pattern is a frequent itemset, the prefix path ending in {carrots, eggs} must be found. The same procedure as before is applied recursively and the prefix path is then found in the following prefix path (Figure 5).

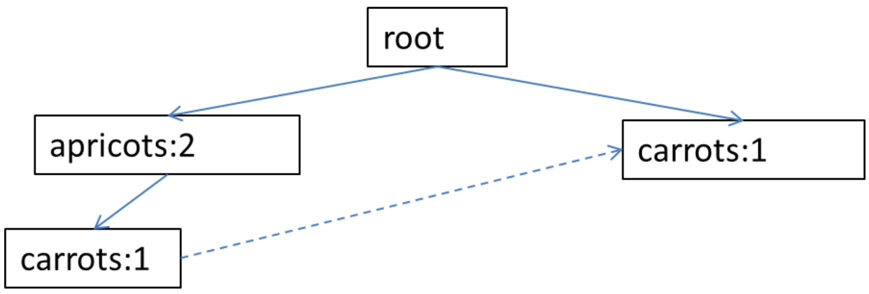


Figure 5: {carrots, eggs} prefix path.

Thus the count ≥ 2 with respect to carrots. After applying this procedure a number of times recursively, for this dataset the following itemsets are found to be the frequent ones:

Suffix	Frequent Itemset
{eggs}	{eggs}, {eggs, dumplings}, {apricots, dumplings, eggs}, {carrots, eggs}, {apricots, eggs}
{dumplings}	{dumplings}, {carrots, dumplings}, {bread, carrots, dumplings}, {apricots, carrots, dumplings}, {bread, dumplings}, {apricots, bread, dumplings}, {apricots, dumplings}
{carrots}	{carrots}, {bread, carrots}, {apricots, bread, carrots}, {apricots, carrots}
{bread}	{bread}, {apricots, bread}
{apricots}	{apricots}

As previously stated, FP-growth has a number of advantages with respect to Apriori, in particular in that it only requires two steps to define the general FP-tree to start the rule mining procedure, as has been illustrated. It is also much faster than Apriori in the rule mining task.

Disadvantages of FP-growth

FP-growth also has some *disadvantages.* First of all, the FP-tree may be expensive to build, since if the dataset is big it may not fit in memory. Secondly the *support* metric can only be calculated once the tree has been created. This effectively means that in datasets in which the tree cannot be built, the support also cannot be calculated, meaning that the data analyst should resort to something simpler, like, for example, the Apriori algorithm previously explained.

3 Assignment Bundle: Frequent Itemset Mining

Students should work on the assignment bundle section on Frequent Itemset Mining (LU07: Frequent Itemset Mining in RapidMiner) and complete the related exercises.

References

1. R. Agrawal and R. Srikant (1994): **Fast algorithms for mining association rules in large databases,** *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, Santiago, Chile, September 1994*, pp. 487-499.

2. J. Han, H. Pei, and Y. Yin**: Mining Frequent Patterns without Candidate Generation**, *Proc. Conf. on the Management of Data (SIGMOD'00, Dallas, TX)*, New York, NY: 1-12 ACM Press.