Association Rules

Proseminar Data Mining

Bertan Ege Teberci
Fakultät für Informatik
Technische Universität München
Email: teberci@in.tum.de

***Abstract*— Association rules are one of the most researched fields in data mining and used to make important inferences based on the co-occurance of items in a dataset. Through the generation of association rules, data analysts can extract useful information out of databases. In this paper, the definition and generation process of association rules are presented, as well as their importance and application methods in various areas and some major algorithms for generating association rules such as AISM, SETM and Apriori.**

I. Introduction

While the size of databases is growing rapidly through constant inputs of new actions and transactions, the need for finding ways and algorithms to analyse datasets arises simultaneously. In order to get some meaningful or useful results out of a dataset during a data mining process, certain techniques and procedures are used to generate helpful implications. One of the most significant ones of these techniques is association rule mining, which aims to produce some inference rules out of a dataset to find out occurance patterns between items in the dataset.

Many business enterprises use datasets consisting of the transactions their customers create, in order to make managerial decisions on which products to buy and whom to target. On the one hand, having access to huge size of information gained through their customers might make business managers be able to make decisions based on quantitative analyses on the basket data, which basically consists of items bought by a customer over a period of time [1]; but on the other hand, it is quite challenging and toilsome to make inferences out of the stored data due to abundance of irrelevant relations. To overcome this challenge, association rules between various items are used to reduce the overall complexity of the data set by composing certain relations between items into more generalized rules.

II. Definition

Association rules were first introduced in [1]. Frequent relationships or common relation patterns between a set of items in a data set can be identified through them and these relationships are based on co-occurrence of the items. [2] Extracting all such rules has a significant importance for marketing and customer segmentation applications by giving the ability to find out common buying patterns of the customers. [3]

The problem of finding association rules in a database and the formal definition of them can be explained in the following way : Let $I = \{i_1, i_2, i_3, i_4, ...\}$ be the set of all items in a basket data and $T = \{t_1, t_2, t_3, t_4, ...\}$ be the set of all transactions. Each transaction t_i represents a subset of items selected from I. Besides, let X and Y be subsets of items, i.e. $X, Y \subset I$. We can say that a transaction t_i contains X or Y, if X or Y is also a subset of t_i, so $X \subseteq t_i$ or $Y \subseteq t_i$. An association rule describes an implication of the form $X \implies Y$ where X and Y are disjoint sets, so $X \cap Y = \emptyset$. [3] X is defined as the antecedent of the rule, where Y is the consequent. In contrary to the its general usage in logic, the inference sign in the rule indicates not certainty but a relatively high, more precisely higher than a predefined threshold, probability of occurence of the itemset Y, if the itemset X is already present.

To illustrate and explain the formal definition, let's take the transaction list shown in Table 1 as an example :

TABLE I
Transaction List

TID	Basket
1	Bread, Butter, Egg
2	Butter, Egg, Milk
3	Butter
4	Bread, Butter

On the first column of the list, TID represents the unique identifier of the given transaction, where on the second column the purchased items are given. At first glance, it is easy to notice a pattern between bread and butter, as well as bread and eggs, since whenever a customer buys bread or egg, he also buys butter as well. In other words, we can generate an association rule $Bread \implies Butter$ or $Egg \implies Butter$ from the given transaction list.

There are two measures being used to identify the strength or interestingness of an association rule : support and confidence. The support is the ratio of the number of all the transactions t_i that contain $X \cup Y$ to the number of transactions in T, formally :

$support(X, Y) = \frac{|\{t_i \in T | X \cup Y \subseteq t_i\}|}{|T|}$

The confidence is the ratio of the number of all the transactions t_i that contain $X \cup Y$ to the number of all the transactions that contain X, formally :

$confidence(X, Y) = \frac{|\{t_i \in T | X \cup Y \subseteq t_i\}|}{|\{t_j \in T | X \subseteq t_i\}|}$

Support is a substantial measure to consider, because a very low support might lead to incorrect implications due to lack of sufficient information in the data set. That's why support can be used to eliminate weak or insufficiently substantiated rules by giving how often the items in the rule appears in the transactions.

On the other hand, confidence indicates the reliability of a rule. In other words, confidence measurement helps on estimating the conditional probability of the rule. Thus, it is also an important measure to consider during association analysis.

III. Association Rule Mining

Nowadays, while we are still ascending to the peak of the Big Data era; considering the size of databases being used, it is very hard to discover association rules in a large database in optimum ways. Therefore, some certain protocols must be set and various algorithms must be built in order to overcome the complexity and the performance concerns on mining of association rules.

Formally, the problem of mining association rules can be defined as follows : Find all the rules with minimum support and confidence values in a transaction set T, where thresholds for minimum support and confidence are set as minsup and minconf. [4]

While analyzing a database to generate association rules between items, there are two forms of additional constraints to consider as explained in [1] :

1. Syntactic Constraints : During the discovery of new rules, one should be interested only in rules that have a specific item $i_A \in I$ in the antecedent and $i_C \in I$ in the consequent.

2. Support Constraints : The support of newly discovered rules must be satisfiable. In other words, the support must be above a certain threshold which makes it certain that the rule is worth to consider.

Considering these constraints, we may separate the problem of mining all association rules in a database into two subproblems :

1. Finding all itemsets with a support value above the minimum support, these itemsets are also called large or frequent itemsets [4].

2. Generation of the desired rules by using the large itemsets, where each subset of every large itemset must satisfy the minimum confidence [2].

A. Large Itemset Generation

There are several algorithms which can be used to generate large itemsets. Despite the differences in their way of finding frequent itemsets, in general they are all based on similar principles.

A data set with n items can include up to 2^n large itemsets. The challenge on generating large itemsets is to determine the support count for each 2^n candidate itemsets. In order to do this, we need to check for each transaction whether the candidate is contained in the transaction or not and if it is, then to increment the support count. To perform this operation, a lattice structure can be used to list all possible itemsets [4] as shown in Figure 1, where each possible candidate k-itemset in a database with 4 items a, b, c and d is listed :

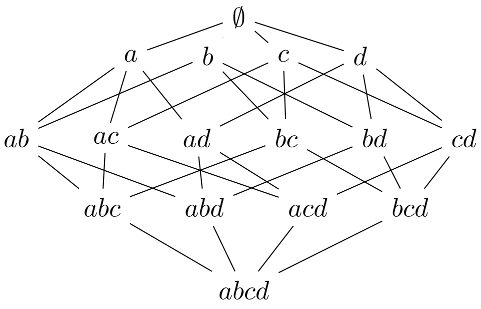

Fig. 1. Lattice Structure

Generating large itemsets with by listing all possible large itemsets as seen in the lattice structure and then calculating the support count for each candidate is very costly and redundant, because it requires in worst case totally 2^i combinations where i is the number of items in the itemset I. These combinations actually correspond to all subsets of the set of items in I [1]. But the redundancy is that in most of the cases we do not need to check the support count for all the candidates, since some of them might be eliminated through various techniques which are formalized and composed in following algorithms :

1) AIS: The AIS algorithm is the first algorithm that is published in [1] aiming to generate all itemsets in a database. It is targeted to discover qualitative rules, but limited to only one item in the consequent part. In other words, only the rules in form $X \implies i_c$ can be generated where X is a subset of the item set I and i_c is a single item in I.

The AIS algorithm makes multiple passes over the entire database, where a frontier set is used to keep track of the candidate itemsets to be extended. During each pass, the support for candidates is measured by scanning all transactions. In the first pass only single items are considered as candidate, therefore the algorithm counts the support for only individual items. Based on the minimum support count the items below the minimum value or in short the small itemsets are eliminated. During the second pass the support count for candidate 2-itemsets, which are generated through the extension of large 1-itemsets with other items in the transaction, are accumulated and checked whether they are above the support threshold. This process iterates for each $k+1$-itemset until any of them becomes empty [5]. Algorithm 1 shows the pseudo code for the template of the algorithm.

The extension of frontier set during each pass in the algorithm is built upon the estimation on the frequency of the extended part, in other words, whether the extended itemset is expected to be large or small. Candidate itemsets are generated from ther frontier itemsets by extending them with other items present in the tuple, where an itemset that is expected to be small is extended no more [1].

Let's take the database shown in *Table 1* as an example and apply the AIS algorithm on it where minimum support is set on 1/2. As seen in Table 2, 3, 4 and 5; AIS algorithm generates many candidate itemsets C_k that turn out to be small in the further passes.

Algorithm 1 AIS

```
procedure GenerateLargeItemsets
    $largeSet \leftarrow \emptyset$
    $frontierSet \leftarrow \{\emptyset\}$
$loop$:
    while $frontierSet != []$ do
        $candidateSet \leftarrow []$
        for all $t \in\ T$ do
            for all $f \in\ frontierSet$ do
                if $f \in\ t$ then
                    $extendedCandidates \leftarrow Extend(f, t)$
                    for all $c \in\ candidateSet$ do
                        if $c \in\ extendedCandidates$ then
                            $c.count \leftarrow$ c.count+1
                        else
                            $c.count \leftarrow 0$
                            $candidateSet \leftarrow$ candidateSet+c
        $frontierSet \leftarrow []$
        for all $c \in\ candidateSet$ do
            if $c.count/d.size > minsup$ then
                $largeSet \leftarrow$ largeSet+c
            if $ShouldUseAsFrontier(c)$ then
                $frontierSet \leftarrow$ frontierSet+c
```

TABLE II

C_1

Itemset	Support
Bread	0.5
Butter	1.0
Egg	0.5
Milk	0.25

TABLE III

C_2

Itemset	Support
Bread, Butter	0.5
Bread, Egg	0.25
Bread, Milk	0
Butter, Egg	0.5
Butter, Milk	0.25
Egg, Milk	0.25

This algorithm might have a big importance on association rule mining due to its pioneer role in the field, but it has also some disadvantages performance-wise. The main problem of the algorithm is that it can generate many small candidates [3]. The restriction having only one single item in the consequent is another drawback of the AIS algorithm.

2) SETM: The SETM was introduced in [6] and aims to use SQL to calculate large itemsets. [7] In this algorithm, each itemset is stored in a tuple with form $< TID, itemset >$ where TID refers to the unique identifier of the relevant transaction. As in the AIS algorithm, the SETM algorithm also makes multiple passes over the entire database and during the first pass the counting of the support counts of individiual items is performed. In following passes the candidate itemsets are generated by extending large itemsets of the previous pass. While generating the itemsets, the algorithm also saves a copy of the candidates together with TID of the generated transaction in a sequentiel structure, which makes the usage of relational merge-join operation and the separation of the candidate generation and the support counting processes possible. [7] At the end of each pass, the support count of the candidate itemsets is measured through the aggregation of the sequential structure.

TABLE IV

C_3

Itemset	Support
Bread, Butter, Egg	0.25
Bread, Butter, Milk	0
Bread, Egg, Milk	0
Butter, Egg, Milk	0.25

TABLE V

C_4

Itemset	Support
Bread, Butter, Egg, Milk	0

The main problem of this algorithm is the size of the required space due to the necessity to store a large number of TIDs. Besides, no memory management technique is introduced in the SETM algorithm which makes it hard to handle buffer overflow issues while running this algorithm.

3) Apriori: The Apriori algorithm was firstly introduced in [3] and is by far the most well-known association rule algorithm. This algorithm differs from the AIS and SETM algorithms in the way of generating candidate itemsets and selecting candidates for support counting. [2] Considering the fact that any subset of a large itemset must be large, the Apriori algorithm calculates the support count for only the itemsets found large in the previous pass. This gives us the opportunity to reduce the number of generated candidate items by making it possible to generate k-itemsets by joining large $k-1$-itemsets.

The algorithm collects first the large 1-itemsets. In the further passes, candidate k-itemsets C_k are generated from the large L_{k-1} itemsets found in the previous pass by using the apriori-gen function, which will be covered further on this section. After that, the algorithm scans each transaction in the database and extracts all the itemsets that are contained in the given transaction from the generated C_k itemsets. At the end of the pass support counting is performed for only these extracted itemsets.

Algorithm 2 shows the overall functioning of the algorithm :

The generation of new candidates in each pass is performed by a specific function called $AprioriGen$. It takes as argument L_{k-1}, the large $k-1$-itemsets found in the previous pass, and returns the set of all large k-itemsets. The function consists of the two following operations :

Algorithm 2 Apriori

```
procedure GenerateLargeItemsets
    $L_1 \leftarrow$ large 1-itemset
    for $k = 2; L_{k-1} \neq \emptyset; k++$ do
        $C_k \leftarrow AprioriGen(L_{k-1})$
        for all $t \in\ D$ do
            $C_t \leftarrow Subset(C_k, t)$
            for all $c \in\ C_t$ do
                $c.count \leftarrow c.count + 1$
        for all $c \in\ C_k$ do
            if $c.count \geq minsup$ then
                $L_k \leftarrow L_k + c$
    return $\cup_k L_k$
```

1. Candidate Generation
2. Candidate Pruning

During the first step, L_{k-1} is joined with itself to obtain the candidate set C_k. In the second step, all the itemsets $c \in C_k$ which have some $k-1$-subsets that are not in L_{k-1}, are deleted from the candidate set. The remaining large k-itemsets are returned at the end.

Algorithm 3 illustrates how the generation and pruning processes are performed :

Algorithm 3 AprioriGen

```
procedure GenerateCandidates($L_{k-1}$)
    insert into $C_k$
    select $p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}$
    from $L_{k-1} = p, L_{k-1} = q$
    where $p.item_1 = q.item_1$ and ... and $p.item_{k-2} = q.item_{k-2}$ and $p.item_{k-1} < q.item_{k-1}$
    for all $c \in\ C_k$ do
        for all $(k-1) - subsetss \in\ c$ do
            if $s \notin\ L_{k-1}$ then
                delete $c$ from $C_k$
```

While applying *Apriori* algorithm on the database with a minimum support of 0.5, following candidate itemsets C_k and large itemsets L_k are generated :

TABLE VI
FIRST PASS

$C_1 Itemsets$	Support	$L_1 Itemsets$	Support
Bread	0.5	Bread	0.5
Butter	1.0	Butter	1.0
Egg	0.5	Egg	0.5
Milk	0.25		

TABLE VII
SECOND PASS

$C_2 Itemsets$	Support	$L_2 Itemsets$	Support
Bread, Butter	0.5	Bread, Butter	0.5
Butter, Egg	0.5	Butter, Egg	0.5
Bread, Egg	0.25		

As seen in the example, the number of candidates generated throughout the application of the algorithm is reduced in comparison with the AIS algorithm, which makes $Apriori$ algorithm more efficient and convenient on generating large itemsets.

4) Apriori-TID: The Apriori-TID algorithm shows many similarities with the Apriori algorithm and was also introduced in [3]. It uses the same generation function *AprioriGen* to determine candidate itemsets in the beginning of each pass. Contrary to Apriori, this algorithm does not scan the entire database to calculate support counts before the first pass. Instead of this, the set $\overline{C_k}$ is used where each member is in a tuple form $< TID, X_k >$ and each X_k is a candidate large k-itemset with the identifier TID [3].

At first, $\overline{C_1}$ refers to the entire database where each item i is conceptually replaced by i and $\overline{L_1}$ is the superset of all large 1-itemsets. After that, C_2 is generated by using $apriori_gen$ function and then $\overline{C_1}$ is scanned. During the scanning of $\overline{C_1}$, entries of $\overline{C_2}$ corresponding to transaction t are obtained by selecting members of C_2 which are present in t [2]. At the end of the pass, L_2 is obtained by choosing the candidates from C_2 with a support count above the minimum support. This process is iterated for $k > 1$ until a candidate itemset L_{k-1} is found to be empty.

Algorithm 4 shows how the algorithm works :

Algorithm 4 AprioriTID

```
procedure GenerateLargeItemsets
    $L_1 \leftarrow$ large 1-itemsets
    $\overline{C_1} \leftarrow$ database D
    for $k = 2; L_{k-1} \neq \emptyset; k++$ do
        $C_k \leftarrow AprioriGen(L_{k-1})$
        $\overline{C_k} \leftarrow \emptyset$
        for all $t \in\ \overline{C_{k-1}}$ do

            $$C_t = \{c \in\ C_k \mid (c - c[k]) \in t.set - of - itemsets \wedge (c - c[k-1]) \in t.set - of - itemsets\}$$

            for all $c \in\ C_t$ do
                $c.count \leftarrow c.count + 1$
            if $C_t \neq \emptyset$ then
                $\overline{C_k} \leftarrow \overline{C_k} + < t.TID, C_t$
        $L_k \leftarrow c \in\ C_k \mid c.count \geq minsup$
    return $\cup_k L_k$
```

B. Rule Generation

Along with the generation of the candidate itemsets, how to extract association rules from a given large itemset is another subproblem of the association rule analysis. From each large k-itemset I_k, we can potentially extract upto $2^{k-1} - 2$ association rules by removing of empty antecedents and consequents.

Given a large k-itemset I_k, all potential rules which can be extracted from the itemset can be formalized through partitioning of I_k into two non-empty subsets X and $I_k - X$ and creating an inference in form $X \implies I_k - X$. Since all such rules have already met the minimum support, simply because they are generated from a large itemset, all we need to do to generate association rules from I_k is to check all the potential rules and select the ones satisfying the minimum confidence.

For example, given a large itemset $I_3 = \{a, b, c\}$, there are six candidate association rules that can be generated :

$a \implies b, c$
$b \implies a, c$
$c \implies a, b$
$a, b \implies c$
$a, c \implies b$
$b, c \implies a$

1) Confidence-Based Pruning: To inspect all the potential rules from each large itemset in a database is not efficient. Thus, confidence-based pruning technique can be used to decrease the number of candidate rules which leads to an enormous progress on efficiency. To eliminate some of the potential rules, confidence-based pruning uses the following theorem :

If the confidence of a rule $X \implies I_k - X$ is below the confidence threshold, then the confidence of any rule $X' \implies I_k - X'$ with $X' \subset X$ must be below the threshold as well.

To prove this theorem, consider the rules $X \implies I_k - X$ and $X' \implies I_k - X'$ with $X' \subset X$. The confidence of these rules is equal to $|I_k|/|X|$ and $|I_k|/|X'|$. Since X' is a subset of X, $|X'| \geq |X|$ and therefore the rule $X' \implies I_k - X'$ cannot have a higher confidence than $X \implies I_k - X$ [4].

2) Rule Generation in Apriori Algorithm: For rule generation, the *Apriori* algorithm uses a level-wise technique by considering the number of items in the consequent. Firstly, all the rules with only one item in the consequent are generated. Afterwards, candidate rules are generated from these rules and then confidence values of candidates are counted. While counting the confidence values; if a low-confidence rule $X \implies I_k - X$ is found, all the rules $X' \implies I_k - X'$ with $X' \subset X$ are eliminated without inspection by using the confidence-based pruning technique.

A pseudo-code for the rule generation process in *Apriori* algorithm is shown in *Algorithm 5* :

Algorithm 5 RuleGeneration

```
procedure GENERATERULES
    for all $large k - itemset l_k, k \geq 2$ do
        $H_1 \leftarrow i \mid i \in\ l_k$ APRIORI-GENRULES($l_k, H_1$)

procedure APRIORI-GENRULES($l_k, H_m$)
    $k \leftarrow |l_k|$
    $m \leftarrow |H_m|$
    $rules \leftarrow \emptyset$
    for $k \geq m + 1$ do
        $H_m \leftarrow AprioriGen(H_m)$
        for all $h \in H_m$ do
            $conf \leftarrow |l_k|/|l_k - h|$
            if $conf \geq minconf$ then
                $rules \leftarrow rules + ((l_k - h) \implies h)$
            else
                $H_m \leftarrow H_m - h$
        $m \leftarrow m + 1$
    return $rules$
```

IV. Other Application Methods of Association Rules

Although the origins of association rules are generally based on market basket analysis, there are other various application areas where association rules can be applied to extract useful information from huge datasets. Some of these areas are :

A. Medical Diagnosis

Association rules can be used to assist doctors and physicians on identifying the illness of their patients. The noise in training examples and unreliable tests makes diagnosis a practically hard process to handle. This may result in hypotheses with unsatisfactory prediction accuracy which might lead to critical failures on risky medical applications [8].

Unlike market basket analysis, applying association rules on medical diagnosis requires a division of the items into two categories : diagnostics and symptoms. Therefore, the association between symptoms and diagnostics reveals cause-effect relationships which can be used to obtain more accurate diagnoses given certain symptoms.

B. Protein Sequences

Proteins are one of the most substantial molecular complexes in the cellular machinery of any organism [9]. They are basically amino acid sequences with a unique 3-dimensional structure. The functioning of a protein heavily depends on the sequence and slight differences in the sequence might lead to enormous changes on the structure and the functioning. There are still so many undiscovered aspects on the nature of proteins, however it is now generally believed that amino acid sequences are not random and association rules application is a powerful and convenient tool to decipher the relational patterns in proteins [10]. Association rules are applied on protein sequences to find out whether an amino acid is present or absent in a protein, based on the presence of another amino acid.

C. Credit Card Businesses

Identification of the preferences of different credit card customer groups has a big importance for the banks to make market targeting decisions [11]. Assocation rules are one of the techniques used by banks to discover customer behaviour patterns. The key point is that, unlike market basket analysis, credit card purchases involve different datasets and thus, different association rule analysis techniques are used on credit card businesses.

V. Conclusion

Mining association rules is one of the most used techniques in data mining [2]. It has an important role in market basket analysis and therefore is a powerful tool for businesses to make strategic decisions on targeting their customers. Besides, there are many other fields where application rules can be used to generate inference rules on the co-occurance of the items in a database.

Although association rule generation is in first look a simple process and brings no performance-related issues while applying on relatively small datasets, important efficiency concerns emerge if the working dataset is in huge size. Thus, certain techniques and algorithms are introduced to make it possible to generate association rules efficiently without using too much time and memory space.

References

[1] A. S. R. Agrawal, T. Imielinski, “Mining association rules between sets of items in large databases,” in *ACM SIGMOD 1993 Internat. Conf. on Management of Data, Washington DC*, 1993, pp. 207–216.

[2] L. G. M. H. Dunham, Y. Xiao and Z. Hossain, “A survey of association rules,” in *Technical Report, Southern Methodist University, Department of Computer Science, Technical Report TR 00-CSE-8*, 2000.

[3] R. S. R. Agrawal, “Fast algorithms for mining association rules in large databases,” in *20th Int. Conf. on Very Large Data Bases, Santiago de Chile, Chile*, 1994, pp. 487–499.

[4] P.-N. Tan, M. Steinbach, and V. Kumar, “Association analysis: Basic concepts and algorithms,” pp. 327–414, 01 2005.

[5] S. V. C. Trupti A. Kumbhare, “An overview of association rule mining algorithms,” *International Journal of Computer Science and Information Technologies*, pp. 927–930, 2014.

[6] M. Houtsma and A. Swami, “Set-oriented mining for association rules in relational databases,” in *11th IEEE International Conference on Data Engineering*, 1995, pp. 25–34.

[7] R. Srikant, “Fast algorithms for mining association rules and sequential patterns,” 1996.

[8] N. L. D. Gamberger and V. Jovanoski, “High confidence association rules for medical diagnosis,” in *IDAMAP99*, pp. 42–51.

[9] C. Branden and J. Tooze, “Introduction to protein structure,” 1991.

[10] D. A. Rajak, “Association rule mining- applications in various areas,” 01 2008.

[11] R.-C. Wu, r. s. Chen, and C.-C. Chen, “Data mining application in customer relationship management of credit card business,” in *In Proceedings of 29th Annual International Computer Software and Applications Conference*, vol. 2, 08 2005, pp. 39 – 40.