Mining Frequent Sequential Pattern Using Binary Matrix Approach in Large Databases

P.Sowjanya, M.Kalyani
Assistant Professor, Dadi Institute of Engineering & Technology
sowjanyap@diet.edu.in , kalyanim@diet.edu.in

Introduction

The process of discovering patterns from often occurring ordered events or subsequences is known as sequential pattern mining. For instance, "Customers who buy a Canon digital camera are likely to buy an HP colour printer within a month." Sequential patterns are helpful for shelf layout and advertising in retail data. Sequential patterns may also be used in this sector, as well as telecoms and other companies, for focused marketing, client retention, and a variety of other duties. Sequential patterns can also be used in other contexts, such as network intrusion detection, weather forecasting, production processes, and Web access pattern analysis. Be aware that the majority of works on sequential pattern mining focus on categorical (or symbolic) patterns, while numerical curve analysis typically falls under trend analysis .Due to the vast range of practical applications, the subject of mining Frequent Sequential Patterns (FSPs) from deterministic databases has received a lot of interest in the scientific community. For instance, FSPs can be employed in mobile tracking systems to categorise or group moving objects, and in biological research, FSP mining aids in the discovery of connections between gene sequences.

Think of a store that sells a variety of goods. The management of the supermarket must typically make business decisions such as what to put on sale, how to create coupons, how to arrange products on shelves to maximise profit, etc. An strategy that is frequently utilised to raise the calibre of such decisions is the analysis of historical transaction data. However, up until recently, the computer could only save global data about the total sales over a certain amount of time (a day, a week, a month, etc.). The development of bar-code technology has made it possible to record so-called basket data, which lists the things bought each transaction. Transactions using basket data types don't always include products purchased at the exact same moment. It might include products that a customer has acquired over time. Examples are the recurring purchases made by book or music club members.

Numerous organisations have gathered enormous amounts of this data. The migration of these data sets to database systems, which are often stored on tertiary storage, is happening very slowly. The fact that present database systems do not offer the essential capability for a user interested in taking benefit of this information is one of the primary causes of the limited success of database systems in this field. Sequential pattern mining has been extensively investigated in the literature in the context of deterministic data, and numerous algorithms, including as Apriori, PrefixSpan, SPADE, FreeSpan, and GSP, have been suggested to handle this problem.

Here, the new methodology is put up against the Apriori algorithm. The Apriori algorithm, which R. Agrawal and R. Srikant first proposed in 1994, is a key one for mining frequent itemsets for Boolean association rules. Because the method makes use of previous knowledge of common itemset qualities, the name of the algorithm is based on this fact. andidate item generation is required. Finding regular patterns will take a long time using this method.

Let's define certain terms before we talk about sequential pattern mining. Suppose I = fI1, I2, ::: The set of all items is Ipg. A nonempty set of things is known as an itemset. An ordered list of events is a sequence. It is written as (e1e2e3 _ _ _el), where event e1 comes first, then e2, then e3, and so on. An element of s is another name for event e j. An event in the context of consumer purchase information is a shopping trip during which a customer made purchases at a certain retailer. Thus, the event is an itemset, or an unordered list of the items the buyer bought while travelling. The event is denoted by (x1x2____xq), where xk is an integer. If an element only contains one item, the brackets are eliminated for clarity; hence, element (x) is represented as x. Let's say a customer visited the shop multiple times to purchase items. For the client, a sequence is created by these sequential events. In other words, the consumer purchased the products in s1 first, then s2 afterwards, and so forth. An item can appear more than once in distinct events of a sequence, but it can only do so once in one event of the series. The length of a sequence is determined by how many times each item appears in the sequence. A series of length l is known as a l-sequence.. A subsequence of another sequence, b = (b1b2 _ _ _bm), is a sequence a = (a1a2 _ _ _an),

A Review of The Books

In a sequence data set, where a sequence records an ordering of occurrences, sequential pattern mining looks for common subsequences. We can examine the sequence in which regularly purchased things are purchased, for instance, using sequential pattern mining. Customers might typically purchase a PC first, then a digital camera, then a memory card, for instance. Apriori is the fundamental algorithm for identifying frequent itemsets in sequential pattern mining.

A level-wise search is an iterative strategy used by Apriori to investigate (k+1)-itemsets using k-itemsets. By searching the database, adding up each item's count, and gathering the items that meet the minimal support, the set of frequent1-itemsets is first discovered. The set that results is known as L1. When no more frequent k-itemsets can be located, L1 is then used to find L2, the collection of frequent 2-itemsets, which is then used to find L3, and so on. Each Lk must be located via a complete database scan. The Apriori property, an important attribute, is utilised to condense the search space in order to increase the effectiveness of the level-wise production of frequent itemsets.

A priori requirement: A frequent itemset's non-empty subsets must all also be frequent.

Every iteration in this case makes use of itemset. Finding a frequent itemset will therefore take a long time throughout each cycle. We can now suggest a new strategy that allows us to represent an item's binary representation. In this study, we provide a novel method for identifying recurring patterns called the Binary Matrix approach. By employing this strategy, we are able to both identify common patterns and reduce the time complexity associated with doing so. As a result, the research presented a brand-new approach for mining the most common item sets first using the Binary Matrix of often occurring item sets of length 1. The algorithm's main goal is to build a Binary Matrix with frequent length-1 item sets as row headings and transaction record IDs as columns. The transaction record either contains or does not contain the associated frequent length-1 item set, depending on the value of "1" or "0" in the matrix.

Binary Matrix

The server or service provider that mines cypher databases using association rules to locate the most common item sets. As a result, the research presented a brand-new approach for mining the most common item sets first using the Binary Matrix of often occurring item sets of length 1. The fundamental idea behind the approach is to construct a Binary Matrix with frequent length-1 item sets as row headings and transaction records IDs as column headings. The transaction record either contains or does not contain the matching frequent length-1 item set, depending on the value of "1" and "0" in the matrix. The number of values must then be calculated.

Creation of recurring patterns

Counting the number of individual items in all transactions, such as, can be used to create an initially frequent one item set. Consider item "a." Count the number of "1"s that are present in all transactions opposing item "a." The total count of item "a" is 3 because it is present in transactions 1, 2, and 6. If the count is equal to or higher than the minimal threshold value or support count (2 in our example), the item can be considered to be frequent.

Common 'n' item set

We can use the same method to continue looking for frequent items until we find sets of two, three, or n items. Take into account the two item sets "a, b"; after checking the relevant items opposite "a, b" (both of which should be set to "1"), the count is "1".Transactions 1 and 6 in the table above both have "1" in the places for a and b, thus the count is 2. Because our minimum support count value is 2, "a, b" is now a frequent item. Using the same procedure, you may find the other frequent patterns.

Methodology

The suggested method makes use of data mining to uncover sequential, regular patterns. The binary matrix approach is a new strategy this study proposes for identifying recurrent patterns. This method will enable the discovery of frequent patterns while lowering the temporal complexity of frequent item discovery. There are certain fundamental ideas present, including

Formal Model

Let I equal I1, I2,... I'll be a collection of binary qualities known as things. Let T be a transactional database. Each transaction t is represented as a binary vector, with t[k]= 1 indicating that t purchased the item Ik and t[k]= 0 indicating that it did not. Each transaction is represented by a single tuple in the database. Give X a set of a few items from I. In order for a transaction t to satisfy X, t[k] must equal 1 for all items Ik in X.

Support Restrictions

These limitations relate to the volume of transactions in T that a rule can support. The percentage of transactions in T that fulfil the union of elements in the rule's consequent and antecedent is known as the rule's support. Support shouldn't be mistaken for assurance. Support equates to statistical significance, whereas confidence measures how strong the rule is. In addition to statistical significance, another rationale for support limitations is that, for commercial reasons, we are typically only interested in rules with support over a certain minimal threshold. If the support is insufficient, the rule is not worth considering or is just less desired (it may be taken into consideration later).

Create any and all combinations of goods that have minimum support for a fractional transaction over that threshold. All additional combinations that fall short of the requirement are referred to as tiny itemsets.

Analysis

The main goal of the technique is to build a Boolean matrix (TABLE I) with frequent length-1 itemsets as row headings and transaction record IDs as column headings. There are just two types of values in the matrix: 1 and 0, which indicate whether the transaction record has the associated frequent length-1 itemset or not. The count of columns with the same number of values 1 must then be done, as well as the number of values 1 in each column.The number of values 1 in the column may, in accordance, be the size of the maximum frequent itemset, or vice versa, depending on whether the count of such columns is greater than the minimum support.

As a result, certain values will be calculated, each of which may be the maximum frequent itemsets length. The support of each candidate itemset will then be determined using the Binary matrix, and a collection of candidate itemsets used for extracting maximum frequent itemsets will then be constructed from frequent length-1 itemsets. The candidate itemset is common if the support is more than the minimal support, and vice versa. According to the nonempty sub-sets of frequent itemsets still being frequent, all frequent itemsets will then be removed from the maximum frequent itemsets.

Generally speaking, the new algorithm's three guiding ideas are as follows:

Making a Binary Matrix Using Frequent Length-1 Itemsets as a Guide:

When the transaction database is scanned for the first time, all the frequent length-1 itemsets will be formed from it, and for each frequent length-1 itemset, all the IDs of the transaction records comprising it need to be noted in one array. Then, for each frequent length-1 itemset, a corresponding Binary array with a length equal to the number of transaction records in the database will be constructed. '0' and '1' are the only values present in each array. In the associated Binary n array, the value is 1 if the transaction record has a lot of length-1 itemsets, and vice versa. Finally, a Binary matrix will be built using all of the Boolean arrays with often occurring length-1 itemsets.

Definition 1: The corresponding Boolean array of every frequent length-1 itemset Im[N] is BT1, BT2,..., BTn (1nN), where Im is the mth frequent length-1 itemset, N is the quantity of transaction records in the database, Tn is the ID of the nth transaction record, and BTn's value is 0 or 1 only.

Definition 2: The binary array with N dimensions of the mth frequent length-1 itemset, Im[N], is the Boolean matrix of frequent length-1 itemsets IM*N, where M is the total number of frequent length-1 itemsets.

Extraction of Boolean Matrix's Most Frequent Itemsets
The Binary matrix's columns each correspond to a single transaction record. If the field has a value of 0, it signifies that the associated transaction record has the associated frequent length-1 itemset, and vice versa. As a result, each column's number of values 1 denotes the presence of a certain number of often occurring length-1 itemsets in the accompanying transaction record. The number of value 1 may be the size of the maximum frequent itemset, or vice versa, if the number of transaction records with the same number of value 1 is greater than the minimum support.

As a result, a set of values will be obtained, each of which may be the maximum frequent itemset's length. A series of candidate itemsets will then be formed from frequent length-1 itemsets according to each of the values in descending order, and the support of each candidate itemset could be computed using the Binary matrix of frequent length-1 itemsets. Each candidate itemset is frequent if its support is more than its minimal support, and vice versa. Finally, the size of the candidate itemset, which is the length of the maximum frequent itemset, is needed if the maximum frequent itemsets obtained from the collection of candidate itemsets are not empty. If not, you must carry out the prior action again to check the subsequent step.

If not, the previous process must be repeated in order to check each subsequent value up until the maximum frequent item sets are not empty. The maximum length of a frequent itemset is one if all the maximum frequent itemsets are empty.

Definition 3: An array called Max[n] is used to store values, each of which may be the length of the most frequent itemset, and where n is the array's size.

Definition 4: Since the set of candidate itemsets for the most frequent itemsets C is composed of IM1, IM2,..., and IMn, the corresponding Binary matrix for CMn*N is composed of IM1[N], IM2[N],..., and IMn[N], where IMn is a candidate itemset.

Definition 5: Support(C) = IM1[N] And IM2[N] for candidate itemset C. Also, IMn[N]. In Fig. 3, the logical Binary operator "And" is demonstrated between two binary arrays of potential itemsets. If a value of 0 is present, the computation will be 0, and vice versa.

Creating the Maximum Frequent Itemsets and All the Frequent Itemsets

According to the nonempty subsets of frequent itemsets still being frequent, all the frequent itemsets might be derived from all the maximum frequent itemsets. And Definition 5 might be used to determine the support for each frequent itemset. Finally, all of the powerful association rules may be extracted from all of the common itemsets.

Result

Using binary data, the binary matrix technique can find recurring patterns. In terms of pattern count and time length, the new approach will be contrasted with the traditional, or Apriori, approach. Now that certain patterns have been provided, the frequent patterns for both algorithms are shown depending on the minimal support count. In terms of pattern count and time duration, the results will next be analysed. As a result, the outcome was the same in terms of pattern count, but the new strategy required less time.

The specified sequential pattern can first be stored as a dataset, after which we can browse it from a list and add transactions to a specific dataset. The output screen for this browsing dataset will then appear. Once the dataset has been chosen, the output will provide a list of the item sets that are included in the chosen dataset.

The screen with the minimum support count is then displayed after choosing next. If we choose the execute button, which will show the frequent patterns for the Apriori and binary matrix techniques, the minimum support count will be modified. We can obtain the most frequent item sets and their lowest support count by selecting the Apriori algorithm.

Then, after pressing the binary matrix button, we can access the binary matrix, where we can then find the most frequent item sets and their minimal support count. After selecting the analysis button, the analysis of those frequent patterns will be shown. The pattern count and time duration for those frequently occurring itemsets will then be shown.

Then, we can compare the most popular item sets from the two ways. Therefore, for the identical data, the patterns count was the same for both techniques. However, the amount of time will depend on how long it takes to obtain the most frequent item sets with that little assistance.

Conclusion

For the specified sequential patterns, a novel approach is put forth in this study to identify frequently occurring item sets. Here, a binary matrix is being used to mine for recurring patterns. Then, both Apriori and Binary matrix algorithms mine the often occurring patterns. Experimental findings demonstrate our novel approach's superior performance versus Apriori. Thus, an effective frequent pattern mining approach and a comparison of the conventional Apriori algorithm serve as the conclusion of our research. This method reduces the amount of time needed to mine for common patterns. In terms of temporal complexity, the experimental analysis of the Binary matrix approach yields more effective results than Apriori.

Required References