ISSN:0975-9646

An Efficient Frequent Pattern Mining using Boolean Matrix and Transaction Reduction Technique

S.Tejaswi[#1], D.Suresh Kumar[*2], Ch.Sita Kameswari[#3]

[#1]. S.Tejaswi, Student of M.Tech(CSE), Baba Institute Of Technology and sciences

*[*2].D.Suresh Kumar, Asst.Professor, Department of Computer Applications,*
Gayatri Vidya Parishad College for Degree And PG Courses(A)

[#3].Ch.Sita Kameswari, Professor, Dept of CSE, Baba Institute Of Technology and sciences

Abstract: Association Rule Mining is one of the most important functionality in data mining, by using that we can find out frequent patterns. Here we propose a novel method of finding frequent patterns by using Boolean matrix for transactional data set and finding the support count by using "Logical AND" operation and here we can also reduce the transactions by the assumption " A Transaction which do not contain frequent k-items can not contain frequent k+1-items in it. So such transactions can be removed from the transactional data set. So that we can save the memory space and reduce the computational time to generate frequent patterns.

Keywords: Boolean Matrix, Logical And, Transaction Reduction, Association Rule Mining

INTRODUCTION:

Association Rule Mining is very important functionality in Business Intelligence. To impove our business we need to follow some strategies.
Strategies:

(i) We have to sell more than one item to every customer in their every visit
(ii) We have to verify which items are frequently purchased by the customers
(iii) We need to find out the strongly associated items to put them in a package.

METHODOLOGY:

Here we propose a novel methodology to find out frequent patterns by using Boolean Matrix and Transaction Reduction Techniques.
Here first we have to convert our transactional data set into Boolean matrix . Then by using Logical And operation we can find out the support count for each item, i.e.., how we can generate c1. Here min_sup=2. So the items which satisfy min_sup will be sending to L1 like Apriori. For C2 (candidate 2-itemset) we have to do the join operation for L1 with L1. Then we will get C2 subsets. To find the support count for C2 we have to use the Logical And operation between the items in subsets. For ex., if (i1,i2) is a subset then we should do the logical AND between i1 and i2 in all transactions in which both should be present.
The Logical And Operation displays 1 when both are existing and displays 0 in remaining cases. Here we should find the sum where it displays 1(it will be the support count for c2). So finally we can find the support count for all the subsets in C2. The items which does not satisfy the support count will be eliminated.

TRANSACTION REDUCTION APPROACH:

Here after C2 set generated, we have to verify which subsets are eliminated. Now We will remove such transactions from transactional data set which contains the eliminated subsets in it. Now those transaction will be removed from our transactional dataset. Now onwards we have to consider this new transactional data set in which we eliminate some transactions. By using this transaction reduction approach we can find out the frequent patterns within short time and we can also save memory and by this we can also find out best frequent pattern than apriori.

TRANSACTIONAL DATA SET:

TID	List_of_item_IDs
T100	T-shirts,Trousers,Trackpants
T200	Trousers, Shorts
T300	Trousers, Shirts
X T400	T-Shirts, Trousers, Shorts
T500	T-Shirts, Trousers, Shirts
T600	Trousers, Shirts
T700	T-Shirts, Shirts
X T800	T-Shirts, Trousers, Shirts, Trackpants
T900	T-Shirts, Trousers, Shirts

I1:T-shirts
I2: Trousers
I3:shirts
I4:Shorts
I5:Trackpants

Boolean Matrix :

T_ID	**I1**	**I2**	**I3**	**I4**	**I5**
T100	1	1	0	0	1
T200	0	1	0	1	0
T300	0	1	1	0	0
T400	1	1	0	1	0
T500	1	1	1	0	0
T600	0	1	1	0	0
T700	1	0	1	0	0
T800	1	1	1	0	1
T900	1	1	1	0	0
	6	**8**	**6**	**2**	**2**

C1

Item	Support_count
I1	6
I2	8
I3	6
I4	2
I5	2

L1

Item	Support_count
I1	6
I2	8
I3	6
I4	2
I5	2

C2

Item	Support_count
I1,I2	5
I1,I3	4
I1,I4	1 X
I1,I5	2
I2,I3	5
I2,I4	2
I2,I5	2
I3,I4	0 X
I3,I5	1 X
I4,I5	0 X

After removing T400 and T800 new Transactional Data Set as D_{new}

T_ID	**I1**	**I2**	**I3**	**I4**	**I5**
T100	1	1	0	0	1
T200	0	1	0	1	0
T300	0	1	1	0	0
T500	1	1	1	0	0
T600	0	1	1	0	0
T700	1	0	1	0	0
T900	1	1	1	0	0

$C2_{(new)}$

Item	Support_count
I1,I2	3
I1,I3	3
X **I1,I5**	**1**
I2,I3	4
X **I2,I4**	**1**
X **I2,I5**	**1**

L2

Item	Support Count
I1,I2	3
I1,I3	3
I2,I3	4

C3

Item	Support Count
I1,I2,I3	2

L3

Item	Support Count
I1,I2,I3	2

Algorithm:

STEP1: Convert the Transactional Data Set to Boolean Matrix by placing 1 in the transaction where the item is existent. Place 0 when the item is not exist in transaction.

STEP2: At the End of all Transactions count the number of binary 1's for each item(i.e.. support count of C1).

STEP3: Send the items from C2 to L2 for those items in C1 which satisfy the given min_sup(here min_sup=2)

STEP4: To generate C2=L1ΛL2(Logical And Operations). After the Logical And Operations calculate the sum of binary 1's for C2 which will be the support count for C2. Send the subsets to L2 which satisfy the minimum support count. Eliminate transactions from transactional data set which contains the subsets which does not satisfy the given min_support count.

STEP5:: After removing some transactions consider a new transactional data set as $D_{new.}$ Now onwards we consider this new transactional data set for further process.

STEP6:: Similary to generate c3 find the 3-item set after joining l2 with l2. For example if we want to find out the support count of (i1,i2,i3) we should find out the Logical And between the subsets (i1,i2) Λ(i2,i3) Λ(i1,i3)

STEP7: Repeat the above steps until we get the final frequent pattern.

CONCLUSION:

This paper proposes a new methodology which is a refinement to apriori algorithm, by using this approach we can find out the frequent patterns within short time since we use transaction reduction technique which reduces the memory space as well as time complexity so that we can find out the frequent patterns more efficiently than apriori algorithm.

REFERENCES:

[1].http://paginas.fe.up.pt/~ec/files_0506/slides/04_AssociationRules.pdf
[2].http://arxiv.org/ftp/arxiv/papers/1403/1403.3948.pdf
[3]. http://www.cscjournals.com/docs/IJCSR10-01-01-010.pdf
[4].http://www.ijeat.org/attachments/File/v4i2/B3627124214.pdf
[5].http://airccse.org/journal/ijdms/papers/4512ijdms05.pdf