A MATRIX MODEL FOR MINING FREQUENT PATTERNS IN LARGE DATABASES

DIVYA BHATNAGAR

Department of CSE, Sir Padampat Singhania University,
Udaipur, Rajasthan, 313601, India
divyabbb@yahoo.co.in

KAMALRAJ PARDASANI

Department of Mathematics, Maulana Azad National Institute of Technology,
Bhopal, Madhya Pradesh, 462051, India
kamalrajp@rediffmail.com
http://www.krpardasani.com

Abstract:
This paper proposes a model for mining frequent patterns in large databases by implementing a matrix approach. The whole database is scanned only once and the data is compressed in the form of a matrix. The frequent patterns are then mined from this compressed database which brings efficiency in data mining, as the number of database scans is effectively less than two. The computation time is reduced as some of the patterns are mined simultaneously and searching is minimized. Appropriate mathematical operations are designed and performed on matrices to achieve this efficiency.

Keywords: Association rules, Boolean Matrix, Data Mining, Frequent Patterns, Itemsets

1. Introduction

Data mining has been recognized as promising field of database research due to its wide and significant applications in industry. It is a key step in the knowledge discovery process in large databases (Han et. al.2001). It consists of applying data analysis and discovery algorithms that under limitation of acceptable computational efficiency produce a particular enumeration of patterns over the data (Fayyad et al. 1996).

Due to massive amount of data generated from business transactions, there arose a need for an efficient algorithm to discover new interesting patterns from these databases in order to derive new information and knowledge for effective decision making. One of the important problems of data mining is to discover association rules from the database. A key component in association rule mining problem is to find all frequent itemsets. Many algorithms have been implemented for finding frequent patterns for data mining. In large databases the problem of mining frequent patterns gets multifold. Since the database needs to be scanned several times, efficient algorithms are required for mining frequent patterns. One of the important developments in area of association rule mining was development of Apriori (Agrawal et al. 1994) algorithm. It was improved by partition (Sarasere et al. 1995) and sampling (Toivonen et al. 1996), but both of these approaches were inefficient when the database was dense. PASCAL (Bastide et al.) is an optimization of Apriori but when all the candidate patterns are candidate key patterns, then the algorithm behaves exactly like Apriori. A huge calculation and a complicated transaction process are required during the two procedures. Therefore, the mining efficiency of the Apriori-like algorithms is very unsatisfactory when transaction database is very large (Hanbing et al. 2007).The matrix model proposes the most optimized approach with only one database scan and fast computation of frequent patterns. The algorithm transforms a transaction database into a boolean matrix stored in bits (Hanbing et al. 2007). This Boolean matrix is then further reduced to another matrix that is much smaller than the original one. This helps perform computations in a much fast and efficient manner, minimizing memory usage to a very large extent as compared to many other popular algorithms.

2. Proposed Model

A matrix model has been proposed for mining frequent patterns in large databases. In this approach, the database is scanned only once. The number of interesting items and transactions determines the size of the matrix. Columns represent the items and the rows represent the transactions. The presence or absence of items in a transaction is stored as 1 or 0 respectively in a matrix initially. This data compressed in the form of matrix, is now the basis for mining all frequent patterns. The support counts of all 1-itemsets and 2-itemsets are mined simultaneously from this initial matrix, and stored in another matrix called reference matrix. Each item R[i][j] of this matrix used as reference matrix in later stages is the support count of the itemset {i,j} where i and j are the positions of rows and columns representing the interesting items. The diagonal elements of this matrix are the support count of all 1-itemsets and the elements lying above the diagonal are the support counts of all 2-itemsets. This matrix further acts as a reference for mining all the other itemsets. Using this reference matrix, all the frequent 1-itemsets and frequent 2-itemsets along with their supports are stored in matrices F1 and F2 respectively. Using apriori property, only the frequent kth patterns are joined to form k+1th itemets. Since all the k+1th itemsets are formed out of frequent kth items, all the proper subsets of k+1th item are frequent except the 2-itemset that is newly formed in the process of joining the two kth items. The support of this new subset is then read from the reference matrix. The minimum support of the three subsets i, e., the supports of the two kth items and the newly formed subset is then compared with the minimum support. If it is equal to or greater than the minimum support specified, it is stored in the matrix of frequent k+1th itemset Fk+1 along with its support. For example, let I1 and I2 be two items from the matrix of k-itemsets. I1 and I2 are joined if I1[1]=I2[1] and I1[2]=I2[2] and …. I1[k-2]=I2[1-2] and I1[k-1]<I2[k-1]. The support of {I1[k-1], I2[k-1]} is read from the reference matrix and if the minimum of the supports of {I1}, {I2} and {I1[k-1], I2[k-1]} is >= minsup, the itemset I1join I2 is frequent. This process is continued till all the frequent patterns are mined.

2.1. *Proposition*

Let $\alpha, \beta \in$ Fk i.e., α and β be two frequent k-itemsets with α = (i1,…, ik_1, ik), β = (i1,…, ik_1, i*k) and ik < i*k. Now, $\alpha \cup \beta$ = (i1,….,ik_1,ik,i*k) is the candidate frequent k+1-itemset. Now, $\alpha \cup \beta$ has a subset $\gamma = (\alpha - \beta) \cup (\beta - \alpha)$ i.e., {ik, i*k} whose support was not taken into consideration while calculating the supports for α or β. Therefore, its support is to be fetched from the reference matrix. Hence, sup($\alpha \cup \beta$)=min[sup {α}, sup {β}, sup {γ}] is the support of $\alpha \cup \beta$. For example, let α={B,C,E} and β={B,C,F} be the two frequent 3-itemsets. Then (α-β)={E} and (β-α)={F}. Then α and β forms the candidate 4-itemset{B,C,E,F}. In this process, the newly formed subset γ is{E,F}. The support of {E,F} is read from the reference matrix R and the support of {B,C,E,F}=min [{B,C,E},{B,C,F},{E,F}]. The detailed method is presented below.

2.2. *Method*

2.2.1 Transforming Transaction Database to Boolean Matrix

The mined transaction database is T, having n transactions and d items. Let T={T1,T2,…,Tn} be the set of transactions and I={I1,I2,…,Id}be the set of items. We set up a Boolean matrix Mn*d. Scanning the transaction database T, if item Ij is in transaction Ti, where 1=j=n,1=i=d, the element value of Mij is '1,' otherwise the value of Mij is '0.'.

2.2.2. Generating the set of frequent 1 and 2-itemsets

The Boolean matrix Mn*d is scanned and support counts of all 1 and 2-itemsets are computed and stored in a reference matrix Rd*d.

2.2.3. Pruning the reference matrix

The Boolean matrix Mn*d is scanned and support counts of all 1 and 2-itemsets are computed and stored in a reference matrix Rd*d.

2.2.4. Generating the set of frequent k-itemsets

Frequent k-itemsets are generated as per the proposition mentioned above. The proposed algorithm MatPat discovers all frequent patterns.

3. Algorithm

Algorithm: MatPat (Finds frequent patterns by counting inference using matrix)
Input: Reference matrix, R; minimum support threshold, *min_sup.*

Output: F1,…,Fn, all frequent itemsets in T.
Method:
CreateR(T);
GetF1(R); // Create frequent 1 patterns
GetF2(R); // Create frequent 2 patterns
GetFn(R); // Create frequent 3 to n patterns

Procedure GetF1
Input: Reference matrix, R; minimum support threshold, *min_sup.*
Output: F1
Method:
for each row in R
add row to F1 // row is the item
add R[row][row] to F1 // R[row][row] is support

Procedure GetF2
Input: Reference matrix, R; minimum support threshold, *min_sup.*
Output: F2
Method:
for each row in R
for each column >row
if (R[row][col]>=min-sup) // Get frequent 2 patterns.
{
sup2=R[row][col]
add itemset to F2
add sup(itemset) to F2
}

Procedure GetFn (Add frequent 3 to n pattrens)
Input: Reference matrix, R; min. support threshold, *min_sup.*
Output: F3,….Fn.
Method:
for each k-itemset from k=3 to n
{ for each itemset l1 ∈ Fk-1
for each itemset l2 ∈ Fk-1
if (f1 [1] = f2 [12]) ^ (f1 [2] = f2 [2]) ^…..^ (f1 [k-2] = f2 [k-2]) ^ (f1 [k-1] < f2 [k-1]) then
supk= min(sup{f1},sup{f2},sup{f1[k-1],f2[k-1]});
if (supk>=min-sup)
add itemset to Fk
add sup(itemset) to Fk }

Procedure CreateR (To create a reference matrix R).
Input: Transaction matrix, M ;
Output: R, frequent itemsets in T.
Method:
for each row t in M //t is a row in M
for each col i in M //i is a column in M
{
if (M[t][i]==1)
{
R[i][i]++;
for each col+1 j in M
if (M[t][j]==1)

```
R[i][j]++;
}
} // R[i][j] is the support
```

4. Running Example

We illustrate the algorithm on the following dataset for minsup = 3/5(Hongyan et al.). Generate a matrix M of size 5 by 6 from T as shown below.

T

Item	Transaction_id
A	1, 3, 5
B	2, 3, 4, 5
C	1, 2, 3, 5
D	1
E	2, 3, 4, 5
F	1, 2, 3, 4, 5

M

A	B	C	D	E	F
1	0	1	1	0	1
0	1	1	0	1	1
1	1	1	0	1	1
0	1	0	0	1	1
1	1	1	0	1	1

Fig. 1. Transaction Database and corresponding Boolean Matrix

Generate a reference matrix R from M as shown below by scanning all the rows of M. Each time we visit a row, we compute the support counts of 1 and 2-itemsets appearing in that row. For example, in the first row, if the first column has a 1, then it adds 1 in R[1][1] incrementing the support count of A. Then the subsequent columns are visited and if an item is present, it is supposed to be present with A and the support of this item with A is incremented in R.

R

	A	B	C	D	E	F
A	3	2	3	1	2	3
B		4	3	0	4	4
C			4	1	3	4
D				1	0	1
E					4	4
F						5

Fig.2. Reference Matrix

Delete D as it is infrequent. Now since there are 5 frequent 1-items, we get a square matrix R of size 5 by 5. Now all other patterns will be mined from this compressed data.

R

	A	B	C	E	F
A	3	2	3	2	3
B		4	3	4	4
C			4	3	4
E				4	4
F					5

Fig. 3. Reduced Reference Matrix

The diagonal elements in R store the support counts for frequent 1-itemsets. Searching these diagonal elements, we delete the row and column for infrequent 1-itemset. Thus frequent 1-itemset F1 = ({A},{B},{C},{E},{F}) and F2 = ({A,C}, {A,E}, {B,C}, {B,E}, {B,F}, {C,E}, {C,F},{E,F}). For mining we use row and column positions to read the support of an item. Thus support count of items in F1 are read from R as sup(A) = R[1][1], sup(B) = R[2][2], sup(C) = R[3][3], sup(E) = R[4][4], and sup(F) = R[5][5]). Similarly, supports counts of 2-itemsets are computed in F2. Therefore, F1 and F2 are stored as follows.

F3

I1	I2	I3	Sup
1	3	5	3
2	3	4	3
2	3	5	4
3	4	5	3

>>

Itemset	Sup
A C F	3
B C E	3
B C F	4
C E F	3

Fig. 4. F1 and F2 showing all frequent 1- itemsets and 2- itemsets respectively.

For example, first two rows of F2 are be joined to get a 3-itemset {A,C,F} i.e., {1,3,6}. The proper subsets of {1,3,6} are {1,3},{1,6} and {3,6}. The support counts of {1,3} i.e., {A,C} and {1,6} i.e., {A,F} are known and now the support of only {3,6}i.e., sup (C,F) is to be found. Since, the sup(C,F) is stored in the reference matrix as R[3][6], it is read from R which is 4. Therefore, min(sup({1,3},sup{1,6}, sup{3,6}) is the support count for {1,3,6}. As it comes out to be 3, which is equal to the min-sup, the 3-itemset {1,3,6} is frequent and stored in F3 along with its support. In this way we get F3 as follows:

F1

Item	Sup
1	3
2	4
3	4
4	4
5	5

>>

Item	Sup
A	3
B	4
C	4
E	4
F	5

F2

I1	I2	Sup
1	3	3
1	4	3
2	3	3
2	4	4
2	5	4
3	4	3
3	5	4
4	5	4

>>

Itemset	Sup
A C	3
A F	3
B C	3
B E	4
B F	4
C E	3
C F	4
E F	4

Fig. 5. F3 showing all frequent 3-itemsets.

Here, {B, C, E} and {B, E, F} form the new itemset {B, C, E, F}. The supports of {B, C, E} and {B, E, F} is known. The support of only the new 2-itemset {E, F} is required. It is then read from R. Following the previous procedure, we get F4 as shown below.

F4

I1	I2	I3	I4	Sup
2	3	4	5	3

>>

Itemset	Sup
B C E F	3

Fig. 6. F4 showing frequent 4- itemset.

Since there is only one pattern in F4, no further patterns are generated.

5. Conclusion

In this paper, a frequent pattern mining model based on matrix is proposed. The model scans the transaction database only once. Since it stores all transaction data in bits, it needs less memory space and can be applied for mining large databases. We evaluate the designed algorithm MatPat against Apriori and PASCAL. In both of these algorithms, a database pass is made whenever a support count is required. Unlike PASCAL where database is to be accessed until all the new candidate patterns are not key patterns, MatPat accesses the database only once. Also, in PASCAL, if the data is weakly correlated, the number of frequent patterns is small as compared to the total number of patterns, and in most cases all frequent patterns are also key patterns (Bastide et al.).

In this matrix model the support of all the frequent 1-itemsets and 2-itemsets is counted in only the first database scan after which the database will not be scanned any more. The support counts of all the higher itemsets will be read from the available reference matrix. Unlike PASCAL, in case of weakly correlated datasets, the matrix model does not change its behavior.

Let us look at the time complexity of the apriori algoithm. Considering the row-wise storage where $T \approx d2n\tau$, and including the dependence on the data size we get the time complexity of apriori: $T = O(d2n)$. In case of the column-wise storage scheme $T = O(dn)$ (Markus et al.). MatPat generates all frequent 1 and 2–itemsets simultaneously by examining all n transactions. All other frequent k-patterns, $n>k>2$ are generated from the reference matrix of size m, where m is the total number of frequent 1-items. Therefore, the time complexity of MatPat is O(n). The following table compares MatPat with Apriori and Pascal.

Table 1. A comparative chart

Measures	Apriori	PASCAL	MatPAt
No. of database passes	4	2	1
Support counts from database	19	10	0
Time Complexity	$O(d_2n)$ / O(dn)	O(dn)	O(n)

MatPat discovers the patterns faster and performs very efficiently. It is easy to implement and it can give very good results especially in large databases. Some important areas where this algorithm can be applied are data stream mining and incremental data mining. It can also be implemented on distributed databases and parallel processing can be applied to make it faster. Further, the storage requirement can be reduced if the algorithm generates the reference matrix directly from the database thus eliminating the need of transaction matrix. Thus the proposed model presents a novel and efficient technique for mining frequent itemsets in large databases.

References

[1] Han, J., Kamber, M.: Data mining: *Concepts and Techniques*. Morgan Kaufmann Publishers, San Francisco (2001).

[2] Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.:Mining frequent patterns with Counting Inference, SIGKDD Explorations Vol. 2 , Issue 2

[3] Hongyan Liu, Jiawei Han, Dong Xin, Zheng Shao: Top-Down Mining of Frequent Patterns from Very High Dimensional Data

[4] Sarasere A.,Omiecinsky E., and Navathe S. : An efficient algorithm for mining association rules in large databases. In Proc. 21st VLDB, pages 432-444, Sept. 1995.

[5] Pujari A.K. : Data Mining:*Techniques*: Universities Press.

[6] Fayyad U.,Piatetsky-Shapiro G., Smyth P., and Uthrusamy R (Eds.). Advances in Knowledge Discovery and Data Mining. AAAI press, Menlo Park,CA, ISBN:0-262-56097-6, pages:611,1996.

[7] Agrawal R. and Srikant R. : Fast algorithms for mining association rules in large databases. In Proc. 20th VLDB, pages 478- 499, Sept. 1994.

[8] Toivonen H. L sampling large databases for association mining rules . In Proc. 22nd VLDB, pages 134-145, Sept. 1996.

[9] Hanbing Liu1 and Baisheng Wang : An association rule mining algorithm based on a boolean matrix. In Data Science Journal, Volume 6, Supplement, 9 September 2007.

[10] Markus Hegland : The Apriori Algorithm - a Tutorial. March 30, 2005 9:7 WSPC/Lecture Notes Series: 9in x 6in. heg05a.