Finding the boundaries of attributes domains of quantitative association rules using abstraction- A Dynamic Approach

RANJEET KUMAR
Department of Information & Communication Technology, Manipal Institute of Technology, Manipal, Karnataka-576104.
INDIA

PREETHAM KUMAR
Department of Information & Communication Technology, Manipal Institute of Technology, Manipal, Karnataka-576104.
INDIA

ANANTHANARAYANA V.S
Department of Information Technology, National Institute of Technology, Surathkal, Karnataka.
INDIA

Abstract: - The Need to extract the association rules from the pool of varied data, having gained increased momentum in the field of data mining, necessitates the discovery of methods to process multi-dimensional data, and find the qualitative or quantitative association rules from it by considering all the relevant fields in an efficient manner In this paper we propose an efficient and novel algorithm for finding the boundaries of attributes domains dynamically. It first builds an abstraction, called Multi-Variate Tree**,** in single scan of the database. During this construction the boundaries of domains of (quantitative) attributes are identified dynamically. These identified attributes with boundary values which are frequent are then used for finding association rules.

Key-Words: - MVT, boundary values of attributes, dynamic approach.

1 Introduction

Whenever we talk of data-mining, we are referring to a huge amount of data [1]. Under the given scenario, in which a single scan of the database itself seems to be a gigantic task, multiple scans of the database cause too much of overhead, in terms of memory consumption and computation-time. In certain cases, it can prove too demanding for the system to meet the desired end.

With this end in view, it becomes highly imperative that we devise techniques that need smaller number of scans of the database, and consume less memory space for generation of the desired result. Discovery of association rules being the canonical task of data mining, it goes without saying that, simply finding the frequent patterns does not necessarily serve the purpose, as it is equally important to know its association with other relevant information fields [1,2,3,4], that have high relevance with the corresponding transaction, in order to get a comprehensive result that encompasses all the associated key areas of the transaction. Of course we have the decision tree [1] algorithm which associates relevant information of a transaction with items bought, only if the class label attribute is present in the database.

In this paper we are introducing a novel method for discovering the frequent item-sets, and the boundary values of the attribute fields associated with them. Here, we construct a tree called Multi-Variate Tree (MVT), which stores not only the item number and its count, but also the minimum and the maximum values of the other relevant information fields i.e. attributes, associated with that item in the transactional item-set. Finally, we can mine the association between item-sets and the other attributes of the transaction [4,5], in terms of the range of values of the various attributes associated with a frequent item-set, in single scan [6] of the MVT.

2 Construction of Multi-Variate Tree, and Finding Boundary Values of Attributes for Item-sets

The algorithm is divided into two main phases. The first phase is to build an efficient tree, called Multi-Variate Tree**,** in a single scan of database[6] using the items in each transaction**.** In the second step, MVT is scanned once, using a recursive function, to obtain the quantitative association rule between frequent item-sets and the corresponding information fields i.e. attributes. The construction of MVT, and the mining of quantitative association rule from the MVT are outlined as follows.

Structure of a Node in MVT

Each **node** in the MVT has following two parts, as shown in Figure 1:

{ *item part:*
inum: item number
count: support count
clink : link to child nodes
slink :link to sibling nodes
ilink: link to next node, having same item-number
relevant-information part:
Each Relevant-info node has the structure:
{ **range**[2] : // stores min/max values
flink : link to next Relevant-info node.
}
}

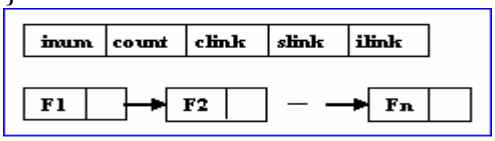

Figure 1. Node structure in a MVT

2.1 Algorithm for Construction of MVT

Input: D, a transaction database, in which items are considered in an increasing order of item numbers.

Output: The MVT

Create a root node of the MVT and label it as "root". Let each transaction be represented by **Trans** [**P**| **F**], where **P** is the set of items bought, and **F** is the set of relevant information Fields, with **F*i*** as its i^{th} field.
Initialize **Trancount**, the total no. of transactions in the database, to 0.
For each transaction **Trans** in database do the following:
Increment **Trancount** by 1.
Call **insert_tree(Trans)**, which performs the operations stated as follows:
Let **p** point to the first item in the set **P**, **F*i*** be the i^{th} field in the set **F**, and **T** denote the current node (starts with root node).

Repeat until **P** is non-empty
{
If **T** has a child **N** such that
N.item-number=p.item-number, then
increment **N**'s **count** by 1.
For each **F*i*** in **F**,
if *N.[Fi].min-value* is greater than the *Trans.[Fi].value*, update *N.[Fi].min-value* with *Trans.[Fi].value*, else if *N.[Fi].max-value* is less than the *Trans.[Fi].value*, update *N.[Fi].max-value* with *Trans.[Fi].value*;
Else
Create a new node **N**, and initialize its count to 1, and set *N.[Fi].min-value* and *N.[Fi].max-value* to *Trans.[Fi].value*.
If **N** is the only child node of **T**, then
T's **child-link** (*clink*) is linked to **N**;
Else link **N1** (T's youngest child node) to **N**, using **N1**'s **sibling-link** (*slink*)
Set **T=N** and increment **p** to point to next item in the item-set **P**.
}

2.2 Finding item-sets and the boundary values of attributes associated with them

Input: MVT.

- **min_sup**, the minimum support count threshold

- **min_isetsize**, the minimum size of the itemset.

Output: The complete set of frequent item-sets, and the range of values of information fields, i.e. attributes, associated with them, depending on the min_sup.

Method:

Scan the entire MVT using a recursive function for depth-first-scan. Each time we come across a leaf node or a node representing a branching point in the MVT, take the set of all the intermediate items between the root node and that node, inclusive of that node. Also, for the item-set thus obtained, if the conditions of **min_sup** and **min_isetsize**, project the range of values of the information field, i.e. min-max values of each of the information fields **F*i***, contained in the node corresponding to the item having highest item number in that item-set.

3 Illustration

Apply the above algorithm to a sample database, shown in Table 1, where we have the information about the age, basic salary and hra (human resource allowance) of each customer, and the various items bought by him/her. The corresponding tree structures, in different stages are shown in the Figure 2 through Figure 4. Figure 4 shows the complete tree structure after processing the 10th transaction.

The nodes in the figures showing MVT are shown numbered, " (node number", for the purpose of illustration, and only the item number in the nodes are shown. Other information fields have not been shown individually in each of the nodes of MVT, in order to keep the figures manageable and neat. Instead, the range of values of attributes common to an itemset in the tree has been shown by a separate ellipse, as shown in Figure 2 and Figure 3.

Table 1. Sample database

Age	Basic	Hra	Items
43	4886	2777	1,2,4,5,6,7,8,9
28	6567	2429	1,2,3,4,5,6,7,8,9
44	12537	7198	2,3,4
51	12980	5956	6
30	10996	7281	1,3,4,6,7,8,9
23	9857	4124	1
55	12814	7367	6,7,8
18	11276	3178	1,2,3,4,7,9
49	4012	2226	4
49	4795	6570	2,4,9

In Table 1, the age is in years, basic salary in rupees and hra (human resource allowance) in rupees. Age, basic and hra form the relevant information part of the node, where as the item numbers and their count comprise the item part of the node.

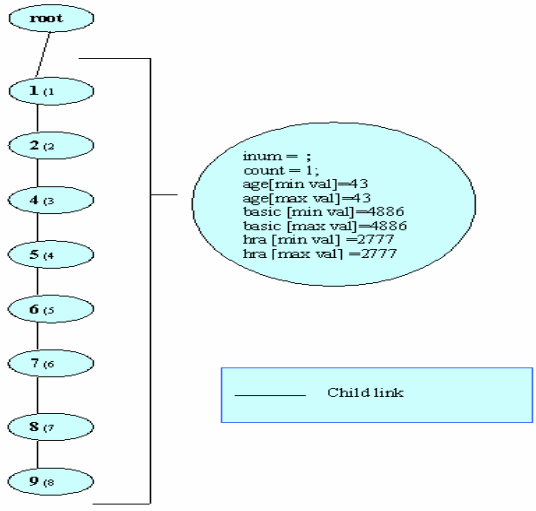

Figure 2. Tree structure after 1st transaction

The tree structure after processing of the first transaction (age=43; basic=4886; hra=2777; items: 1,2,4,5,6,7,8,9) is shown in Figure 2. The MVT starts with a root node, and the nodes, bearing the item number, are created in the order of increasing item number. Since each of the nodes, bearing the respective item number, are newly created, the minimum and the maximum value for each of the associated information fields- age, basic, hra- are same. In this case, each of the nodes numbered

from 1 to 8, has the following values in their information fields.
age[min]=age[max]=43,
basic[min]=basic[max]=4886,
hra[min]=hra[max]=2777,
and the count value of each of the items is 1.

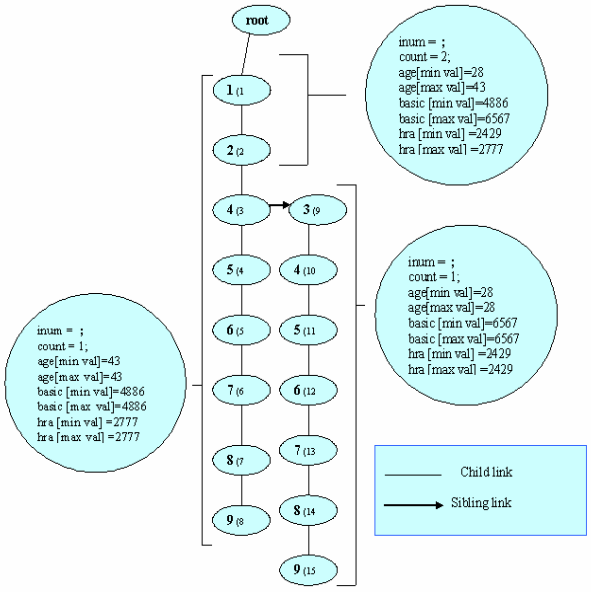

Figure 3. Tree structure after 2nd transaction

After the processing of the second transaction (age=28; basic=6567; hra=2429; items:1,2,3,4,5,6,7,8,9), the MVT has the structure as shown in Figure 3. Here, since item number 1 was already present under the root, we increment its count value by 1, and update its information field as follows. As 28 is not greater than the stored age[max] value of 43, we leave age[max] value unchanged. However, 28 is lesser than the stored age[min] value of 43. so, age[min] value is updated to a new minimum value of 28. Next, the basic salary of rupees 6567 is greater than the basic[max] value , but not lesser than the basic[min] value. Hence, basic[max] value is updated to 6567, where as basic[min] remains unchanged at 4886. Similarly hra[min] is updated to 2429. Again, the node containing item number 2 already exists as a child of node1. Therefore, its count is incremented by 1, and its associated information fields are updated in a similar manner.

Thus, the values contained in the relevant information fields of node 1 and node 2, after 2nd transaction, are as follows:
age[min]=28, age[max]=43;
basic[min]=4886, basic[max]=6567;
hra[min]=2429, hra[max]=2777;
The next item in the transaction is 3. As node number 2 has no child with item number 3, a new node with item number 3 is created under node number 2, as a sibling node of node number 3 having item number 4, with count=1, and other information field values as:
age[min]=age[max]=28;
basic[min]=age[max]=6567;
hra[min]=hra[max]=2429;
Similarly, other nodes are created for the rest of the items in the 2nd transaction. After processing the 2nd transaction, the structure of MVT, and the values contained in its nodes are shown in the Figure 3. Figure 3 also shows the sibling link (*slink)* between the child-nodes of the node, having node number as 2.

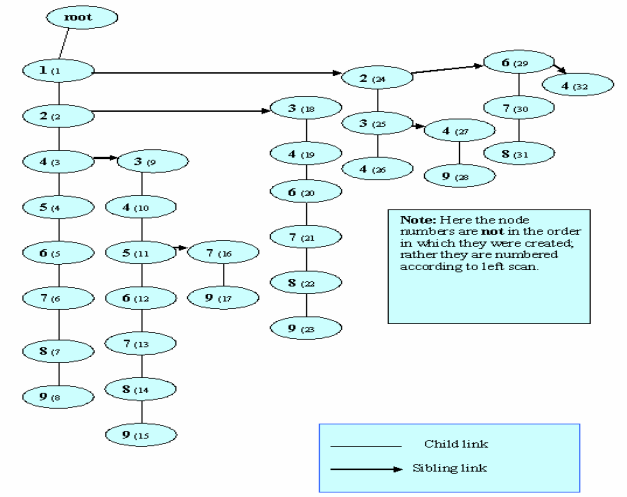

Figure 4. Tree structure after 10th transaction

The complete structure of MVT, after the 10th transaction, is shown in Figure 4. The values stored in the nodes after 10th transaction are shown in Table 2.

Table 2. Values contained in the nodes of the MVT shown in Figure 4

Node no.	inum	count	Age-range	Basic-range	Hra-range
1	**1**	5	18-43	4886-11276	2429-7281
2	**2**	3	18-43	4886-11276	2429-3178
3	**4**	1	43	4886	2777
4	**5**	1	43	4886	2777
5	**6**	1	43	4886	2777
6	**7**	1	43	4886	2777
7	**8**	1	43	4886	2777
8	**9**	1	43	4886	2777
9	**3**	2	18-28	6567-11276	2429-3178
10	**4**	2	18-28	6567-11276	2429-3178
11	**5**	1	28	6567	2429
12	**6**	1	28	6567	2429
13	**7**	1	28	6000	2000
14	**8**	1	28	6000	2000
15	**9**	1	28	6000	2000
16	**7**	1	18	11276	3178
17	**9**	1	18	11276	3178
18	**3**	1	30	10996	7281
19	**4**	1	30	10996	7281
20	**6**	1	30	10996	7281
21	**7**	1	30	10996	7281
22	**8**	1	30	10996	7281
23	**9**	1	30	10996	7281
24	**2**	2	44-49	4795-12537	6570-7198
25	**3**	1	44	12537	7198
26	**4**	1	44	12537	7198
27	**4**	1	49	4795	6570
28	**9**	1	49	4795	6570
29	**6**	2	51-55	12814-12980	5956-7367
30	**7**	1	55	12814	7367
31	**8**	1	55	12814	7367
32	**4**	1	49	4012	2226

Now, let find the item-sets and the boundary values of attributes associated with the item-sets. Consider once again the MVT after the 2nd transaction as depicted in Figure 4.

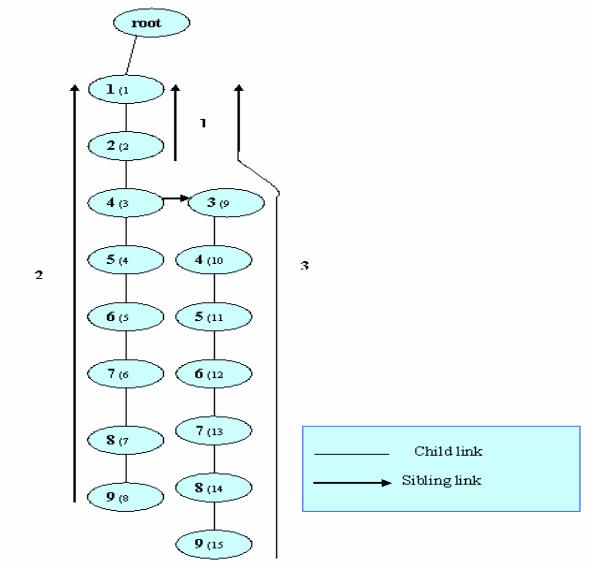

Figure 5. Mining the item-sets for the MVT shown in Figure 2.

Recall that the algorithm, for finding the quatitative association between item-sets and the boundary values of attributes associated with them, requires that we consider the item-set between the root node and a node that happens to be either a leaf node or a branching point, inclusive of that node, and project the range of values of relevant information field as contained in the last item of that item-set, i.e. the item having the highest item number. The lines in Figure 5 denote the span of item-sets. While scanning the MVT shown in Figure 5, we find that the node number 2 is a branching point. So, starting from node number 2, we move up till we come to the root node, as shown by line number1. The item-set thus obtained is {1,2}. The relevant information contained in node number2, pertaining to item number 2 of the item-set obtained, is displayed as follows (refer to Figure 3 for values after 2nd transaction).

Count=2;
Age-range= 18-43;
Basic-range=4886-6567;
Hra-range=2429-2777;

Proceeding this way, we come across node number 8, a leaf node. In Figure 5, line 2 gives the item-set corresponding to node number 8, and the result for the same is as follows:

{1,2,4,5,6,7,8,9}
count=1;
Age-range= 43;
Basic-range=4886;
Hra-range=2777;

Similarly, line 3 gives the following result:

{1,2,3,4,5,6,7,8,9}
count=1;
Age-range=28;
Basic-range=6567;
Hra-range=2429;

Using the method described above, we can find the result for the MVT, obtained after processing all the ten transactions of the sample database, shown in Figure 4. Table 3 shows the complete set of result for the MVT, shown in Figure 4.

Table 3. Item-sets and the boundary values of attributes associated with them.

Itemset	count	Age-range	Basic-range	Hra-range
1	5	18-43	4886-11276	2429-7281
1,2	3	18-43	4886-11276	2429-3178
1,2,4,5,6,7,8,9	1	43	4886	2777
1,2,3,4,5,6,7,8,9	1	28	6567	2429
1,2,3,4,7,9	1	18	11276	3178
1,3,4,6,7,8,9	1	30	10996	7281
2	2	44-49	4795-12537	6570-7198
2,3,4	1	44	12537	7198
2,4,9	1	49	4795	6570
6	2	51-55	12814-12980	5956-7367
6,7,8	1	55	12814	7367
4	1	49	4012	2226

We also have the count value of the item-sets in the result, besides the range of values of relevant information fields i.e. attributes associated with that item-set. As such, we can project only those tuples in the result, whose count value satisfy the minimum support threshold.

4 Experimental Results

We tested this methodology against the standard IBM synthetic data[7], that contained wide range of item numbers (more than 900) in the transaction. To this data, we appended the three attribute fields namely age, basic salary and hra, using data generating programs based on randomization functions in order to ensure that the final test data remained unbiased. The data contained 49,100 transactions. The number of rules obtained under different values of constraints such as minimum support and minimum size of item-sets is shown in Table 4. By increasing the support threshold and minimum size of item-sets, we can filter out the desired rules to get the result showing the boundary values of attributes for only those item-sets that satisfy the minimum threshold criteria.

Table 4. Number of itemsets obtained with IBM data[7] for different values of support threshold and minimum item-set size

Minimum support count %	Minimum size of itemset	Number of Rules obtained
0.002	1	53573
0.004	1	11291
0.006	1	6428
0.008	1	4437
0.01	1	3403
0.012	1	2782
0.004	2	10947
0.004	3	6012
0.004	4	2228
0.004	5	1087
0.004	6	680
0.004	7	518
0.01	5	286

Some of the rules obtained for a support threshold of 0.01% and minimum itemset-size limit of 5 are shown in Table 5 as follows.

Table 5. A Sample of results obtained with IBM data[7] for support threshold of 0.01 % and min. Itemset-size limit of 5

Item numbers of itemsets	Count %	Age-range	Basic-range	Hra-range
19,556,770,811,868,879	0.016	24-59	577913216	2363-6276
8,73,118,467,569	0.06	12-59	4183-13374	2052-7928
8,328,379,380,680	0.226	10-59	4046-1398	2002-7892
8.328,379,380,608,717	0.208	10-59	4046-13978	2002-7892
8.328,379,380,608,717,733	0.2	10-59	4046-13978	2002-7892
8.328,379,380,608,717,809	0.18	10-59	4046-13978	2002-7892
8.328,379,380,608,717,733,809,965	0.174	10-59	4046-13978	2002-7892
8.328,379,380,608,717,733,965	0.012	14-48	5871-12002	3878-7248
8,328,379,608,717	0.012	12-48	4692-11093	2121-7760
8,109,129,319,939	0.024	20-59	4141-12811	3083-7508

5 Conclusion and Future work

The data structure obtained through this algorithm is MVT, which can be used to represent a large database in a compact form. MVT can be constructed using single database scan. We have used it here for quantitative association between the item-sets and the boundary values of attributes associated with them. Our research reinforces the following: (1) While mining an extremely large transaction database, we should discourage the use of algorithms that require multiple scans of the massive database or create huge data structures. What is required is a method for developing compact data structure, in minimum number of scans of database, that can store entire relevant information in a compact form to obtain the desired association rule. (2) In order to obtain a comprehensive result pertaining to a transaction, stress should be laid upon an approach that takes into account all the relevant information fields of the transaction and tries to discover a rule based on them.

After having obtained promising result from the IBM synthetic data, next we are going to obtain the live data from a Cancer Hospital, and use this algorithm to obtain quantitative rules pertaining to various diagnostic parameters that can help in determining the chances of cancer, as well as those pertaining to the treatment process.

References: -

[1] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufman, San Francisco, CA, 2001.

[2] J. Han, J. Pei, and Y. Yin. *Mining frequent patterns without candidate generation.* In *ACM-SIGMOD*, Dallas, 2000.

[3] R. Hemlata, A. Krishnan, C. Scenthamarai, R. Hemamalini. *Frequent Pattern Discovery based on Co-occurrence Frequent Tree.* In Proceeding ICISIP-2005.

[4] R. Agarwal, T. Imielinski and A. Swami. *Mining Association Rules between Sets of Items in large Databases* in Proc. 1993- *ACM-SIGMOD* Int. Conf. Management of Data, pages 207-216, Washington D.C. May 1993.

[5] Ramakrishnan Srikant and Rakesh Agarwal. *Mining Generalized Association* Rules in Proc. of the 21st Int'l Conference on very large database, Zurich, Switzerland, December 1995.

[6] Ananthanarayana V. S, Subramanian, D.K., Narasimha Murthy, M- *Scalable, Distributed and Dynamic Mining of Association Rules using PC-Tree*; pp559-566, HIPC, Bangalore, INDIA, 2000.

[7] IBM/Quest/Synthetic data.