A Study of Data Reduction Using Multiset Decision Tables

Uday Seelam and Chien-Chung Chan
Department of Computer Science
University of Akron
Akron, OH 44325-4003
chan@uakron.edu

Abstract

In rough set theory, observations of objects in a domain of interest are stored in a decision table where each row denoting one object. Objects with same description are duplicated. Duplications may be reduced by using information multisystems, which can be further transformed into Multiset Decision Tables (MDT). In this paper, we have demonstrated the efficacy of MDT when dealing with very large data sets. Experimental results based on the well-known Intrusion Detection System (IDS) data set show that the size of MDT is only 1/3 of the original decision table when all features are used. It could be further reduced to 1/7 when a set of 7 features is used. We also showed that the running time of generating an MDT is faster than generating a C4.5-like decision tree based on the MS SQL server 2000.

1. Introduction

Rough set theory introduced by Pawlak [1, 2] provides sound mathematical foundation for developing data analysis tools. Observations of objects in a domain of interest are stored as decision tables, which are special kinds of information systems introduced by Pawlak [1, 2]. Dealing with large data sets using decision tables has been regarded as a main challenge. To alleviate this issue, the concept of information multisystems was introduced by Grzymala-Busse [3]. The basic idea is to use multiset representation of decision tables. Intuitively, a large data set with many duplicated observations can be effectively represented by multiset-based approach. To facilitate the development of data analysis tools, an information multisystem can be further transformed into a multiset decision table introduced by Chan [4]. In this paper, we have used the well-known Intrusion Detection System data set [5] to show the effectiveness of multiset approach to the representation of decision tables. We have developed SQL stored procedures to transform decision tables into Multiset Decision Tables (MDT). Space complexity is measured by file size in terms of Mega bytes and number of records in the files. Time complexity is measured by using SQL Profiler. We have compared the running time for generating MDT tables with the time to generate C4.5-like decision trees [6] using data mining tools available on the MicroSoft SQL server 2000. The results are very much in favor of our stored procedures for generating MDT. It is not a fair comparison, because decision tree tools include further reduction of input data sets into decision tree classifiers. Because decision tree tools are known for its efficiency in generating tree classifiers, the purpose of our comparison is to show that there is room and it is possible to develop classifier generating algorithms based on MDT that are as efficient as decision tree approach.

The paper is organized as follows. In Section 2, we briefly review related concepts of information multisystems and MDT. In Section 3, we formulate the problem of transforming a decision table into MDT and discuss the development of SQL stored procedures for the task. Experimental results are presented in Section 4. Conclusions are given in Section 5, followed by references.

2. Related Concepts

The notion of information systems was introduced by Pawlak to represent knowledge about objects in a domain. An information system is a pair S = (U, A) where U is a nonempty finite set called the universe, and A is a nonempty finite set of attributes. Each attribute in A is associated with a set of values called domain of the attribute. Objects in U are described by values of attributes in A. In other words, each object is represented by a vector of attribute values. Therefore, a vector of attribute values may denote more than one object. Decision tables are special cases of information systems. In a decision table there is a designated attribute called *decision attribute* and another set of attributes are called *condition attributes*. Information multisystems were introduced by Grzymala-Busse [3]. The basic idea is to represent an information system using *multisets* [7].

0-7695-3032-X/07 $25.00 © 2007 IEEE
DOI 10.1109/GrC.2007.90

Object identifiers represented explicitly in an information system is not represented in an information multisystem. More precisely, an *information multisystem* is a triple $S = (Q, V, \tilde{Q})$, where Q is a set of attributes, V is the union of domains of attributes in Q, and $\tilde{Q}$ is a *multirelation* on $\bigtimes_{q \in Q} V_q$. Let M be a multiset, and let e be an element of M whose number of occurrences in M is w. The sub-multiset $\{w \cdot e\}$ will be denoted by $[e]_M$. Thus M may be represented as union of all $[e]_M$'s where e is in M. A multiset $[e]_M$ is called an *elementary multiset* in M. The empty multiset is elementary. A finite union of elementary multisets is called a *definable multiset* in M. Let P be a subset of Q, a *projection of* $\tilde{Q}$ *onto* P is defined as the multirelation $\tilde{P}$, obtained by deleting columns corresponding to attributes in $Q - P$. Note that $\tilde{Q}$ and $\tilde{P}$ have same cardinality. A *multipartition* χ on a multiset X is a multiset $\{X_1, X_2, \ldots, X_n\}$ of sub-multisets of X such that

$$\sum_{i=1}^{n} X_i = X$$

where the *sum of two multisets* X and Y, denoted $X + Y$, is a multiset of all elements that are members of X or Y with the number of occurrences of each element e in $X + Y$ is the sum of the number of occurrences of e in X and the number of occurrences of e in Y. Follow from [3], classifications are multipartitions on information multisystems generated with respect to subsets of attributes. Specifically, let $S = (Q, V, \tilde{Q})$ be an information multisystem. Let A and B be subsets of Q with $|A| = i$ and $|B| = j$. Let $\tilde{A}$ be a projection of $\tilde{Q}$ onto A. The subset B generates a multipartition B_A on $\tilde{A}$ defined as follows: each two i-tuples determined by A are in the same multiset X in B_A if and only if their associated j-tuples, determined by B, are equal. The mulitpartition B_A is called a *classification on* $\tilde{A}$ *generated by* B.

The idea of multiset decision tables (*MDT*) was first introduced in [4]. Let $S = (Q = C \cup D, V, \tilde{Q})$ be an information multisystem, where C are condition attributes and D is a decision attribute. A multiset decision table is an ordered pair $\boldsymbol{A} = (\tilde{C}, C_D)$, where $\tilde{C}$ is a projection of $\tilde{Q}$ onto C and C_D is a multipartition on $\tilde{D}$ generated by C in $\boldsymbol{A}$. We will call $\tilde{C}$ the *LHS* (Left Hand Side) and C_D the *RHS* (Right Hand Side).

A decision table is shown in Table 1 where U has 12 objects, the condition attributes are $\{C_1, \ldots, C_4\}$ and the decision attribute is D. The corresponding information multisystem of Table 1 is shown in Table 2.

Examples

Table 1. A decision table.

U	C_1	C_2	C_3	C_4	D
1	False	False	False	False	Neptune
2	False	False	False	False	Neptune
3	False	False	False	False	Neptune
4	True	False	False	False	Normal
5	True	False	False	False	Normal
6	False	False	False	False	Teardrop
7	True	False	False	False	Normal
8	False	False	False	False	Neptune
9	False	False	False	False	Teardrop
10	False	False	False	False	Neptune
11	False	False	False	False	Neptune
12	True	False	False	False	Normal

Table 2. An information multisystem.

C_1	C_2	C_3	C_4	D	W
False	False	False	False	Neptune	6
True	False	False	False	Normal	4
False	False	False	False	Teardrop	2

An MDT derived from Table 2 is shown in Table 3, where d_1, d_2, and d_3 denoting the decision values Neptune, Normal, Teardrop, respectively and w_1, w_2, and w_3 are numbers of occurrences for the decision values, respectively.

Table 3. MDT derived from Table 2.

C_1	C_2	C_3	C_4	W	d_1	d_2	d_3	w_1	w_2	w_3
False	False	False	False	8	1	0	1	6	0	2
True	False	False	False	4	0	1	0	0	4	0

3. Generation of MDT

From the above example, it is clear that MDT representation is more compact than the information multisystem, where duplicate vectors of condition attributes values are kept whenever they are associated with more than one decision value. However, all vectors are unique in MDT. The association with multiple decision values can be encoded as Boolean vectors denoted by the decision columns in MDT. In addition, MDT supports the computation of lower and upper approximations of subsets directly and the derivation of basic probability assignment function of the Dempster-Shafer theory as shown in [4]. In the following, we will discuss how to generate MDT from decision tables efficiently. The efficacy will be presented in Section IV.

An MDT can be easily constructed from a decision table but proper care should be taken while populating the MDT. One simple way to generate a MDT from a decision table is to generate an intermediate table by applying SQL group-by operation on a decision table. Then, it is followed by a row by row updating of the target MDT. However, the row by row updating on MDT in the database causes major slow down of the task. In order to overcome the drawback in the simple approach, another intermediate table is created which is in the encoded form of the condition attributes in the decision table. Encoding the values of condition attributes helps the procedure to perform better while populating the MDT with decision values and weights. The encoding is done by taking distinct values of the condition attributes and assigning them a unique ID which acts like encoded ID for that particular set of condition attributes. After the encoded table is constructed, an intermediate table consisting of the encoded id, the decision value and the count or weight of the decision value is constructed and populated. This table is the information multisystem represented in another form with encoded ID's. Then, the MDT is constructed and populated. One important aspect while populating MDT is updating the MDT table. Updates on MDT should be done in bulk as row by row updating involves lots of interactions with the database table and is very time consuming. All the rows which are affected with a decision value are kept in track and are updated at once onto the table. The number of updates on the table will depend on the size of the domain of a decision attribute.

We used three temporary tables for implementing the MDT generating procedure and a split function. The split function takes delimited string and delimiter as parameters and returns a table with splitted substrings.

In the following, we will call each row of condition values in a decision table as *condition vector*. The first table, 'dt_encode', contains encodings for condition vectors appeared in a decision table. Each row in the 'dt_encode' table consists of a unique condition vector and its corresponding encoded id, which is an integer, generated by a program.

Another temporary table for information multisystem is 'wt_dtable' which contains the encoded ids of condition vectors and the decision values along with the count of decision values i.e., number of times a decision value is repeated for a particular encoded id of the condition vector. This table is an encoded weighted decision table.

Last temporary table used by our stored procedure is 'dom_dec' which holds all the distinct values of the decision attribute. This helps us to write dynamic SQL statements.

The three temporary tables are used to generate required data for updating the decision and weight columns of an MDT, which is updated in bulk for each decision value, i.e., all the records corresponding to a particular decision value are updated on to the table at once. Using a cursor called 'Decision_Cursor' we traverse through each decision value of the decision attribute. Form the update statement using dynamic strings in which the decision values are either 1 or 0 and the weights are summed up for each decision value. The update statement is executed before the cursor fetches for the next value of the decision attribute. The detail steps are outlined in the following pseudo code.

Procedure Bulk_Updating

Inputs:

MDT table with condition attributes and encoded ID populated,
Wt_table which contains encoded ID, decision values and their weights,
Table containing decision values D = {D1, D2, D3 …, Dn}

Outputs:

MDT table populated with decision and weight columns (d1, d2, d3…., dn) and (w1, w2, w3…., wn)

```
begin
    Construct dynamic Update string: update MDT set;
    Declare a cursor named as Decision_Cursor and Open it
    for (each decision value in D) do
      Join tables MDT and wt_table on encoded ID and
      distinct decision value;
      begin
          if (wt_table(Decision Value)=D1)
          then   {  d1:= 1;
                    w1:= w1+wt_table(weight of D1);
                 }
           else  {  d1:= d1;    w1:=w1; }
          if (wt_table(Decision Value)=D2)
          then   {  d2:= 1
                    w2:= w2+wt_table(weight of D2);
                 }
          else   {   d2:= d2;    w2:=w2; }
          ……
          ……
          if (wt_table(Decision Value)=Dn)
          then   {dn:= 1;
                   wn:= wn+wt_table(weight of Dn);
                 }
          else   {   dn:= dn;    wn:=wn; }
          for (each value of D) do
              Update MDT;
      end; // for loop
      Close Decision_Cursor;
end; //  procedure Bulk_Updating
```

4. Experimental Results

The dataset that we used for evaluating our MDT generating procedure is the IDS dataset made available by MIT Lincoln labs, which was downloaded from UC Irvine ML repository [5]. The data set has been used in many studies [8, 9, 10, 11]. The total number of attributes in the Intrusion Detection System Dataset is 42. Out of these attributes, there are 41 condition attributes and 1 decision attribute. Among the condition attributes, 34 attributes are

numerical attributes and 7 are categorical attributes. The domain of the decision variable is 16 for this dataset. The total file size is 772 MB and the total number of records is 4,826,656.

We implemented the MDT generating procedure using MS SQL stored procedure and ran on MS SQL Server 2000. The running time for the SQL Stored procedure was taken using the SQL Profiler. SQL Profiler helps us to know the exact timing of when the procedure started executing and at what time the procedure completed the execution. It also gives the detailed execution times step by step as the procedure will have multiple updates and the timing for these updates can also be known. As all our data were in tables, they were converted to text files and the sizes of the files were noted down. The conversion of tables into text files was done using BCP (Bulk Copy Program) in SQL. Using this single statement query the tables can be transferred to text files whose size can be known easily.

Table 4. Experimental results using 42 attributes.

Size of Decision Table	Records in DT	Time of MDT	Size of MDT	Records in MDT	Time of Decision Tree
1.5 MB	10,000	9 sec	1.09 MB	5,724	7 sec
4 MB	25,000	11 sec	2.84 MB	16,643	12 sec
8 MB	50,000	22 sec	6.7 MB	41,286	22 sec
12 MB	75,000	44 sec	10.7 MB	65,751	37 sec
16 MB	100,000	91 sec	14.9 MB	89,988	57 sec
40 MB	250,000	449 sec	39.6 MB	233,889	581 sec
80 MB	500,000	768 sec	58.4 MB	307,400	1252sec
120 MB	750,000	1313 sec	89.7 MB	444,884	2574sec
160 MB	1,000,000	1948 sec	117 MB	582,437	2858sec
320 MB	2,000,000	2311 sec	150 MB	730,644	4159sec
640 MB	4,000,000	2636 sec	193 MB	911,227	6523sec
772 MB	4,826,656	3144 sec	224 MB	1,031,520	7286sec

In the first experiment, we used all 42 attributes from the IDS data set. In addition to running our MDT generator, the decision tree data mining tool on MS SQL server was applied to both data sets. We evaluated 12 different sizes of decision tables. The experimental results are shown Table 4 and Figures 1, 2, and 3.

Table 4 shows the size of original decision table, number of records in the decision table, execution time for generating MDT from the given decision table, size of the MDT generated, number of records in the MDT, and the time to generate a decision tree.

From the above table, few conclusions can be drawn: (1) When input data set grows larger, the reduction in MDT becomes more significant, as shown in Figure 1. (2) The number of records in the MDT is much smaller than the number of records in the original dataset, as shown in Figure 2. (3) The execution time of the MDT using stored procedure is better when compared to the MS decision trees for larger datasets, as shown in Figure 3.

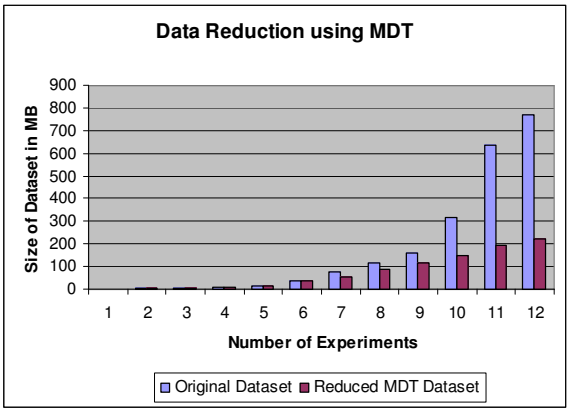

Figure 1. Reduction in data set size.

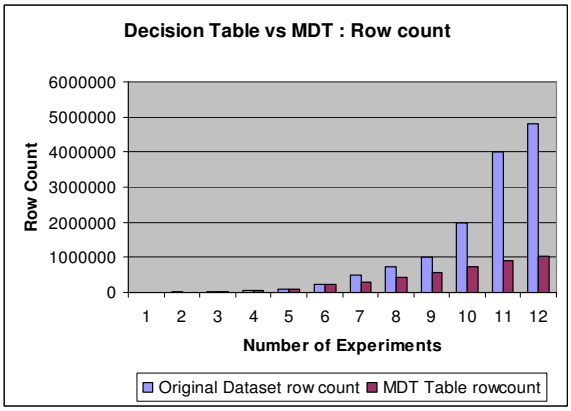

Figure 2. Reduction in number of records.

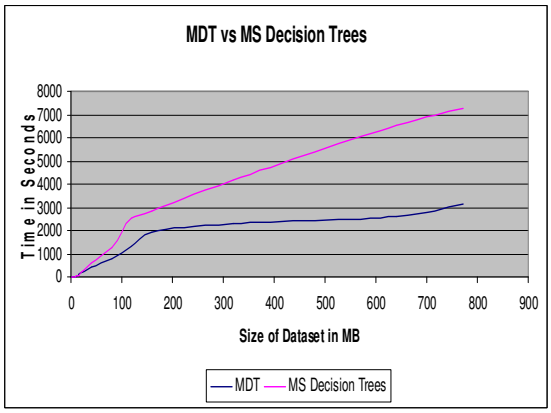

Figure 3. Running time of MDT versus Decision Tree.

In our second experiment, we have manually selected 7 attributes which are considered the most important ones. The selected features are columns 12, 13, 15, 17, 19, 21

and 22 in the original IDS data set. The decision attribute remains the same which is column number 42. Experimental results are shown in Table 5 and Figures 4, 5, and 6.

Table 5. Experimental results using 7 attributes.

Size of Decision Table	Records in DT	Time of MDT	Size of MDT	Records in MDT	Time of Decision Trees
0.38 MB	10,000	0.5 sec	0.08 MB	2	1 sec
0.93 MB	25,000	1 sec	0.25 MB	6	3 sec
1.79 MB	50,000	2 sec	0.77 MB	17	6 sec
2.68 MB	75,000	3 sec	1.26 MB	26	9 sec
3.56 MB	100,000	4 sec	1.36 MB	26	14 sec
8.86 MB	250,000	10 sec	3.26 MB	54	53 sec
17.9 MB	500,000	16 sec	6.32 MB	83	74 sec
26.9 MB	750,000	27 sec	10.2 MB	116	88 sec
35.7 MB	1,000,000	50 sec	13.1 MB	149	141 sec
71.4 MB	2,000,000	95 sec	18.8 MB	204	334 sec
142.8 MB	4,000,000	191 sec	21.3 MB	231	706 sec
172 MB	4,826,656	286 sec	24.3 MB	253	1013 sec

It can be observed that the time taken for executing the procedure to generate MDT is a quarter of the time taken to generate a decision tree as shown in Figure 4. Moreover the number of records is reduced from almost 5 million to just 250 records, as shown in Figure 5. The size of the generated MDT is almost 1/7th of the original decision table for the complete dataset with 7 condition attributes and one decision attribute, as shown in Figure 6.

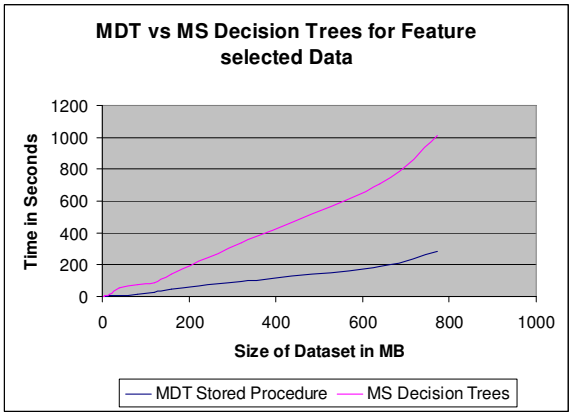

Figure 4. Running time of MDT versus Decision Tree over 7 attributes.

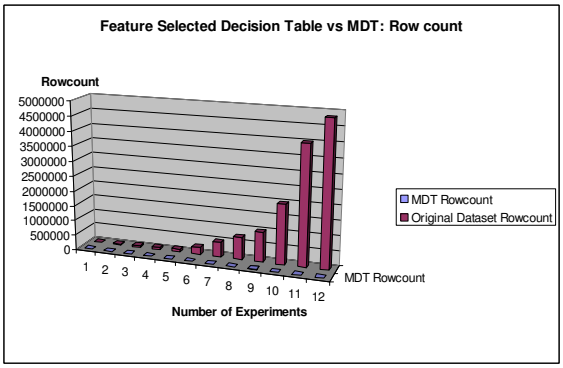

Figure 5. Reduction in record counts using 7 attributes.

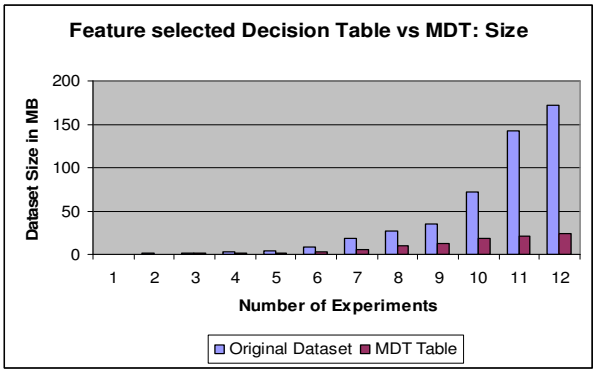

Figure 6. Reduction in data set size using 7 attributes.

5. Conclusions

In the paper, we presented a SQL implementation of stored procedure for transforming information multisystems into Multiset Decision Tables. We have demonstrated the effectiveness of using MDT for data reduction by evaluating the performance on a large IDS data set. The running time for generating MDT compared well to the generating of C4.5-like decision trees on MS SQL server 2000. Our results suggest that MDT can provide a compact representation for applying other reduct generating algorithms and rule learning algorithms.

Further optimization of the stored procedure may be achieved by tuning the queries. The execution time could be further improved by reducing the number of updates on the table. The procedure made use of the SQL cursor construct may be improved by better implementation.

6. References

[1] Pawlak, Z., "Rough sets: basic notion," *Int. J. of Computer and Information Science* 11, 344-56, (1982).

[2] Pawlak, Z., *Rough Sets: Theoretical Aspects of Reasoning about Data*, Kluwer, 1991.

[3] Grzymala-Busse, J.W., "Learning from examples based on rough multisets," *Proc. of the 2nd Int. Symposium on*

Methodologies for Intelligent Systems, Charlotte, North Carolina, October 14-17, pp. 325-332, 1987.

[4] Chan, C.-C., "Learning rules from very large databases using rough multisets," *LNCS Transactions on Rough Sets*, (J. Peters, A. Skowron, J.W. Grzymala-Busse, B. Kostek, R.W. Swiniarski, M. Szczuka Eds.), Vol. 1, pp. 59 – 77, Springer –Verlag Berlin Heidelberg, 2004.

[5] Hettich, S. and Bay, S. D. (1999). *The UCI KDD Archive [http://kdd.ics.uci.edu*]. Irvine, CA: University of California, Department of Information and Computer Science.

[6] Quinlan, J.R., *C4.5: Programs for machine learning*. San Francisco, Morgan Kaufmann, 1993.

[7] Knuth, D.E., *The Art of Computer Programming. Vol. III, Sorting and Searching*. Addison-Wesley, 1973.

[8] Chebrolu, S., A. Abraham, J.P. Thomas, "Feature detection and ensemble design of intrusion detection systems," *Computer & Security* (2005) 24, 295-307.

[9] Mukkamala, S**.** and Sung, A.H., (2002) "Feature Ranking and Selection for Intrusion Detection Systems Using Support Vector Machines," *url = "citeseer.ist.psu.edu/583136.html*".

[10] Sung, A.H.; Mukkamala, S., "Identifying important features for intrusion detection using support vector machines and neural networks," 2003 Symposium on Applications and the Internet Workshops (SAINT 2003), 27-31 January 2003 - Orlando, FL, USA, Proceedings. IEEE Computer Society 2003, ISBN 0-7695-1873-7, 209 – 216.

[11] Wenke Lee, Salvatore J. Stolfo, Kui W. Mok, "A Data Mining Framework for Building Intrusion Detection Models," *sp*, p. 0120, 1999 IEEE Symposium on Security and Privacy, 1999.