Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Madrid, Spain, February 15-17, 2006 (pp417-422)

Some algorithms for data table (re)ordering using Monotone Systems

LEO VÕHANDU, REIN KUUSIK, ANTS TORIM, EIK AAB, GRETE LIND
Department of Informatics
Tallinn University of Technology
Raja 15, 12618 Tallinn
ESTONIA

Abstract: - We present here some algorithms for data table reordering to give it more informative form. All these algorithms are Monotone System algorithms and have been created in our department within several years. Main aim of these algorithms is to summarize entire data table with one "average" object called best decision (BD) and to open the hidden inner structure of the data table. The BD object is defined here as one that gives the maximal value to the weight function. At first we give a review of a computationally simple weight function to define BD which does not account for the dependencies between the attributes and we describe a method named Scale of Conformity to implement it. We describe several table reordering techniques based on that scale. Finally we define a BD as a branch in the decision tree and introduce a weight function that takes attribute dependencies into account. We present some examples to demonstrate effectiveness of such techniques.

Key-Words: - Data table, Data mining, Monotone systems theory, Decision tree, Best decision, Algorithms

1 Introduction

Data mining approaches like rule-sets [1], decision trees [8] and clustering [2] describe data tables with structures that are quite comprehensive put not as compact as traditional descriptive statistics – mean, median, mode. Traditional descriptive statistics however are calculated independently for every attribute and don't account for correlations between values of different attributes. "Average" object composed from values calculated independently for each attribute might not exist in the data table at all.

In our paper we will discuss at first the task of representing the data table with one "average" row that we call the best decision (BD) and describe a method named Scale of Conformity to implement it. The concept of the BD is inspired by decision trees [8] and monotone systems theory [4, 5, 7]. It was first described in [3], however it was not presented to a wider international audience. In contrast to the Quinlan's ID3 algorithm [8], we are not interested in the entire decision tree but only in the one branch of it – the best decision. On the scale of conformity [10] we define BD and review its shortcomings. Then we describe several data table reordering techniques (minus technique, plus technique, mixed technique) which have been created in our department using the scale of conformity as a weight function and also the monotone systems theory. We discuss the results of described methods and describe how to build a monotone system to the given data table. After that we give our definition for the BD for dependent attributes case. We present also an algorithm that uses the concept of a potential to prune the search-space as described in [3].

2 Problem Formulation

BD was originally intended as a formal way to select one from possibly conflicting recommendations of experts [3]. It is however applicable to any table containing discrete data as a way to describe it with one "average" row. We present a computationally simple (O(MN) operations) but quite effective scale of conformity approach [10] and review its shortcomings. Then we describe several data table reordering techniques using the scale to discover typical and fuzzy parts of the data (data mining task). After that we present another best decision approach on decision trees.

2.1 Best Decision Measured by the Scale of Conformity

The problem of finding the BD can be defined in many ways. We will use so-called scale of conformity [10]. Conformity as a measure for an object is calculated by a transformation where instead of the attribute value we use its frequency (so-called frequency transformation). For every row in the data table we calculate the sum of all attribute-

value frequencies. This sum is the conformity weight for that row. The greater the conformity of a row the more typical its attribute values for the data table are. According to that principle the row with the greatest conformity becomes BD.

2.2. Example of Scale of Conformity

Table 1. Initial data table and its frequency table

	A1	*A2*	*A3*	*A4*	*A5*	*W(O)*
O1	1	2	2	2	2	12
O2	2	1	2	1	1	20
O3	2	1	2	1	1	20
O4	1	1	2	1	2	16
O5	2	2	1	2	1	14
O6	2	1	1	1	1	18
Attribute's value						
1	2	4	2	4	4	
2	4	2	4	2	2	

Weights of objects (rows) are in the last column W(Oi). For example: W(O1)=2+2+4+2+2=12

As we see, rows 2 and 3 are the BD-s.

3 Data table reordering techniques using the scale of conformity

We present here several reordering techniques called minus technique, plus technique, mixed technique. All they apply Monotone System Theory [7] and as a weight function they all use conformity.

3.1 Building a Monotone System on a data table

Let us have a data table X(N,M), where every element Xij can have a discrete value from range hj=0,1,...,Kj-1.

1. Choose a suitable weight function $\pi(X_{ij})$.

2. Choose activities applicable to elements ('+' or '-') and a rule to recalculate weights.

3. Applicable activity ('+' or '-') and the rule for recalculation of weights together have to guarantee the system to be monotone. I.e. if we apply an activity ('+' or '-') to an element a $\in$ X it causes the weight $\pi_X(b)$ of all elements b $\in$ X which are connected to a, to change in the same direction ('+' increases, '-' decreases the weight). If a and b are not connected then the weight does not change.

3.2 Minus technique

"Minus technique" is a simple method to order a N*M data matrix [10]. Below we will shortly describe the algorithm. First we order the rows and then the columns. (To reorder the columns we can transpose the matrix and use the same algorithm again.) As a result of ordering we can easily see typical and fuzzy parts of the data.

Assume that we have a data matrix X(N, M), i=1,…,N, j=1,…,M. Every element Xij has a discrete value from an interval [1,K].

Algorithm:

S1. Calculate frequencies FT(t,j) for every attribute's values t=1,2,…,Kj in columns j, where j=1,…,M

S2. For every row i=1,2,…,N find the sums (weights) W(i) = Σ FT(t, j), j=1,…,M

S3. Find R = min W(i); remember i

S4. Eliminate row i from the matrix

S5. If there are yet rows in the matrix then goto S1 else to S6

S6. Reorder matrix rows in the order of elimination

S7. End

3.3 Example (of the minus technique)

To demonstrate that algorithm we are using data from table 1.

Table 2. Order of elimination of rows (6 iterations)

	It1	*It2*	*It3*	*It4*	*It5*	*It6*
O1	12					
O2	20	19	17	14	10	
O3	20	19	17	14	10	5
O4	16	13	13			
O5	14	12				
O6	18	18	15	13		

Table 3. Order of elimination of attributes (5 iterations)

	A1	*A2*	*A3*	*A4*	*A5*
It1	12	20	16	20	18
It2		18	14	18	18
It3		16		16	14
It4		12		12	
It5				6	

Table 4. Meanings of attributes' values

attribute \ value		1	2
A1	gender	Female	Male
A2	has a flat	Yes	No
A3	education	Higher	Secondary
A4	activeness	Yes	No
A5	has a car	Yes	No

Table 5. Reordered data matrix

	A1	*A3*	*A5*	*A2*	*A4*	*W(Oi)*
O1	1	2	2	2	2	12
O5	2	1	1	2	2	12
O4	1	2	2	1	1	13
O6	2	1	1	1	1	13
O2	2	2	1	1	1	10
O3	2	2	1	1	1	5
W(Aj)	12	14	14	12	6	

Those object conformity values at the moment of elimination are written to the orders of reordered table data table. As we can see, the reordered data matrix is more informative and is easier to interpret.

3.4 Plus technique

"Plus technique" is another simple method for N*M data matrix ordering [6, 10]. Below we will shortly describe the algorithm and give a short example. Its complexity is $O(N^2M)$ operations. First we order the rows and then the columns. As a result we can again easily see typical and fuzzy parts of the data matrix.

Assume that we have a data matrix X(N, M), i=1,…,N, j=1,…,M. Every element Xij has a discrete value from an interval [1,K].

Algorithm:

S1. Calculate frequencies FT(t,j) for every attribute's values t=1,2,…,Kj in columns j, where j=1,…,M
S2. For every row i=1,2,…,N find the sums (weights) W(i) = Σ FT(t, j), j=1,…,M
S3. Find R = max W(i); remember i
S4. Eliminate row i from the matrix
S5. Increase frequencies of the eliminated row i elements by one: FT(t,j):= FT(j,t)+1
S6. If there are yet rows in the matrix then goto S2 else to S7
S7. Reorder matrix rows in the order of elimination
S8. End

To reorder the columns we can transpose the matrix and use the described algorithm again.

3.5 Example (of the plus technique)

To demonstrate that technique we are using data from table 1.

Table 6. Order of elimination of rows (objects) (6 iterations)

	It1	*It2*	*It3*	*It4*	*It5*	*It6*
O1	12	13	14	14	17	19
O2	20					
O3	20	25				
O4	16	19	22	24		
O5	14	16	18	21	21	
O6	18	22	26			

Table 7. Order of elimination of columns (attributes) (5 iterations)

	A1	*A2*	*A3*	*A4*	*A5*
It1	12	20	16	20	18
It2	14		18	26	22
It3	16		20		26
It4	16		24		
It5	18				

Table 8. Reordered data matrix

	A2	*A4*	*A5*	*A3*	*A1*	
O2	1	1	1	2	2	20
O3	1	1	1	2	2	25
O6	1	1	1	1	2	26
O4	1	1	2	2	1	24
O5	2	2	1	1	2	21
O1	2	2	2	2	1	19
	20	26	26	24	18	

In our case reordering results are the same for minus and plus techniques. In general it does not have to be so.

3.6 Mixed technique

It is a new technique not presented earlier. It uses minus technique to find a "teacher" as the closest row (column) to the eliminated one.

Algorithm starts like minus technique:

S1. Calculate frequencies FT(t,j) for every attribute's values t=1,2,…,Kj in columns j, where j=1,…,M
S2. For every row i=1,2,…,N find the sums (weights) W(i) = Σ FT(t, j), j=1,…,M
S3. Find R = min W(i); remember i
S4. Eliminate row i from the matrix

S5. If there are yet rows in the matrix then goto S6 else to S12

After the first iteration it continues like plus-technique:

S6. Nullify the frequency table FT

S7. Increase frequencies of the eliminated row i elements by one:
FT(t,j):= FT(j,t)+1

S8. For every row i=1,2,…,N find the sums (weights) W(i) = Σ FT(t, j), j=1,…,M

S9. Find R = max W(i); remember i

S10. Eliminate row i from the matrix

S11. If there are yet rows in the matrix then goto S7 else to S12

S12. Reorder matrix rows in the order of elimination

S13. End

3.7 Example (of mixed technique)

Table 9. Order of elimination of rows

	It1	*It2*	*It3*	*It4*	*It5*	*It6*
O1	12					
O2	20	1	4	9		
O3	20	1	4			
O4	16	3				
O5	14	2	2	4	6	9
O6	18	0	2	6	10	

Table 10. Order of elimination of columns

	A1	*A2*	*A3*	*A4*	*A5*
It1	12	20	16	20	18
It2		2	2	2	0
It3			4	8	4
It4			6		8
It5			10		

Table 11. Final reordered table

	A1	*A2*	*A4*	*A5*	*A3*	
O1	1	2	2	2	2	12
O4	1	1	1	2	2	3
O3	2	1	1	1	2	4
O2	2	1	1	1	2	9
O6	2	1	1	1	1	10
O5	2	2	2	1	1	9
	12	2	8	8	10	

Mixed technique always finds the closest member (row or column) to the just eliminated one. For this reason the reordered table is very interesting, it shows so called "evolutionary" way of developing/changing the "teacher". We can see the main changes in that way.

To use described reordering methods there are no serious problems if the number of rows and columns is small (tens or hundreds of attributes and objects). If the data matrix is huge then it is harder to see the patterns and it means that we need some other methods for pattern mining.

3.8 Best decision using attribute dependencies

Conformity scale approach does not take into account the dependencies between the attributes. That can lead to situations where the best decision does not feel intuitively appropriate. Let's examine the following example. Let us have a data table *T*:

Table 12. Example.

	A1	*A2*	*A3*	*A4*	*A5*	*A6*	*A7*
O1	1	1	1	1	3	3	3
O2	1	1	1	2	3	3	3
O3	1	1	1	2	2	2	2
O4	2	2	2	1	2	2	2
O5	2	2	2	1	2	2	2

In our example the number of rows | *T* | is five. We denote the set of attributes in the table *T* by A_T and the set of possible values for an attribute *a* by *Dom(a)*. In our example A_T = *{A1, A2,..., A7}* and *Dom(A1) = {1, 2}*. We call pair *(a, v)* where *a* is an element of A_T and *v* is an element of *Dom(a)* element of decision. For each element *(a, v)* we can calculate its frequency $\pi_T((a, v))$ in data table *T*.

Table 13. Frequencies.

Attribute's value	*A1*	*A2*	*A3*	*A4*	*A5*	*A6*	*A7*
1	3	3	3	3	0	0	0
2	2	2	2	2	3	3	3
3	0	0	0	0	2	2	2

Each object (row) in the data table is a set of elements:

$$\mathbf{O = \{ (a_1, v_1), (a_2, v_2),..., (a_n, v_n)\}}$$

Objects weight according to the scale of conformity is sum of its elements frequencies:

$$W(O)=\pi_T((a_1, v_1))+\pi_T((a_2, v_2))+...+\pi_T((a_n, v_n)) \quad (1)$$

Table 14. Weights of objects in the scale of conformity.

	A1	*A2*	*A3*	*A4*	*A5*	*A6*	*A7*	*W(O)*
O1	3	3	3	3	2	2	2	18

O2	3	3	3	2	2	2	2	17
O3	3	3	3	2	3	3	3	**20**
O4	2	2	2	3	3	3	3	18
O5	2	2	2	3	3	3	3	18

Table 15. The best decision (by the scale of conformity).

(O3)	1	1	1	2	2	2	2

We can see that the object identified as the best decision is not very typical, as there is only one instance of it. In this case the scale of conformity approach does not seem trustworthy. That kind of result was caused by not taking into account the dependencies between the attributes. How should we behave when we assume mutual dependency between the attributes? Below we describe a suitable approach. We present its result (the best decision) to show the difference.

Table 16. The best decision (by another approach)

(O4, O5)	2	2	2	1	2	2	2

Previous methods defined certain ordering for a data set. We now describe a novel method that summarizes data table with one row of the table called best decision (like arithmetic mean summarizes set of numbers).

Concept of the best decision as described here was first formulated in [3] and described in detail in [9]. Decision can be described as one branch in a decision tree. Element of the decision (node of decision tree) is an attribute-value pair (*a, v*). First element defines a sub-table, second element defines a sub-table of the sub-table and so on. Informally, weight of a decision is sum of rows over those recursive sub-tables. Formal definition follows.

We denote the data table T after the selection of an element *(a, v)* by *T \ (a, v)* and define it as a sub-table of *T* that contains all the rows of *T* where value of an attribute *a* equals *v* and all the columns of *T* except the column *a*.

Table 17. Initial data table T

	A1	*A2*	*A3*	*A4*	*A5*
O1	1	2	2	2	2
O2	2	1	2	1	1
O3	2	1	2	1	1
O4	1	1	2	1	2
O5	2	2	1	2	1
O6	2	1	1	1	1

For example if *T* is Table 17 then *T \ (A2, 1)* is:

Table 18. T \ (A2, 1)

	A1	*A3*	*A4*	*A5*
O2	2	2	1	1
O3	2	2	1	1
O4	1	2	1	2
O6	2	1	1	1

Definition 1. Let M be the number of columns in table T. Ordered set $I_T = \langle (a_1, v_1), (a_2, v_2), \ldots, (a_n, v_n) \rangle$ that contains *n* elements from table *T* and where no attribute a_i occurs twice is a decision. If $n = M$ then I_T is a complete decision, if $n < M$ then I_T is a partial decision.

One complete decision for Table 17 is *⟨(A1, 2), (A3, 1), (A5, 2), (A4, 1), (A2, 1)⟩.*

Definition 2. For a decision $I_T = \langle (a_1, v_1), (a_2, v_2), \ldots, (a_n, v_n) \rangle$ we define its weight $W(I_T)$ as follows:

frequency of the element $FT((a_1, v_1))$: number of rows where attribute *a* has value *v.*

$$W(I_T) = FT((a_1, v_1)) + W(I \setminus (a_1, v_1)_{T \setminus (a1, v1)}), \text{ if } M > 1$$
$$W(I_T) = FT(a_1, v_1), \text{ if } M = 1 \quad (2)$$

For Table 17:

$W(\langle (A1, 2), (A3, 1), (A5, 2), (A4, 1), (A2, 1) \rangle) = 4 + 2 + 2 + 1 + 1 = 10$-

Definition 3. The best decision for the table *T* is a decision I_T, with greatest weight. That is, for any decision I'_T in table *T* the condition $W(I_T) \geq W(I'_T)$ holds. We denote the best decision for the table *T* by bI_T.

The best decision and its weight for Table 17 is,

$W\langle (A2, 1), (A4, 1), (A5, 1), (A1,2), (A3,2) \rangle = 4 + 4 + 3 + 3 + 2 = 16$.

Our definition for the best decision has several interesting properties:

- It takes into the account dependencies between the attributes.
- Row described by the best decision is guaranteed to exist in data table.
- The order of elements in the best decision is significant and informative. First (attribute, value) pairs represent greater part of the weight than later elements and describe frequent and correlated (attribute, value) pairs. In the scale of

conformity approach that information is not always available.

- Best decision represents a branch in the decision tree [8]. It can be thought of as representing properties of a typical row in the order of importance.

3.8.1 Brute-force algorithm

A brute-force algorithm for the depth-first search can be described as follows. An efficient algorithm is described in [9]. Recursive function *BestDecision* has three parameters: table (or sub-table) *T*, current best decision *bI* and partial decision under construction *I*. It returns the decision with the greatest weight from two options:

- The best decision that can be constructed by extending partial decision I with elements form the table *T*. That is always $I + bI_T$.
- Current best decision *bI*.

Call for the first level of recusion has the form *BestDecision (T, ⟨⟩, ⟨⟩)* and we assume that $P(\langle\rangle) = 0$.

BestDecision(*T, bI, I*):
(End of recursion) If $M = 1$:
Find *(a, v)* with greatest *FT ((a, v))*
If $P(I + \langle(a, v)\rangle) > P(bI)$, then set $bI \leftarrow I + \langle(a, v)\rangle$
(Recursion) If $(M > 1)$:
For each *(a, v)* in *T*:
Set $bI \leftarrow$ BestDecison$(T \setminus (a, v)), bI, I + \langle(a, v)\rangle)$
(Return) Return the decision *bI*

4 Conclusion

We presented some methods developed by our group in several years to find the best representative of data table and to discover typical and fuzzy parts of the data (data mining task). They are very simple to implement and effective to use. Data table reordering methods give to the researcher more preliminary information to define work hypothesis, and to test them for example with bootstrapping methods. Results we get are not something abstract and separated from data, we can see them always and they are "close" to our data.

References:

[1] Cohen, W. W. Fast effective rule induction. *Machine Learning: Proceedings of the Twelfth International Conference*, 1995, pp. 115-123.

[2] Kaufman, L. and Rousseeuw, P. *Finding groups in data: An Intorduction to cluster analysis*. NewYork: Wiley, 1990.

[3] Kuusik, R. Application of Theory of Monotonic Systems for Decison Trees Generation. *Transactions of Tallinn Technical University,* No. 705, 1989, pp. 47-58.

[4] Kuusik, R., Lind, G. Generator of Hypotheses – an Approach of Data Mining Based on Monotone Systems Theory. *International Journal of Computational Intelligence,* vol.1, 2004, pp. 49-53

[5] Kuusik, R., Lind, G., Võhandu, V. Data mining: pattern mining as a clique extracting task. *Proceedings of the Sixth International Conference on Enterprise Information Systems*, Vol. 2, Porto, Portugal, April 14-17, 2004, pp. 519-522

[6] Kuusik, R., Lind, G., Võhandu, V. Frequent pattern mining as a clique extracting task. *The 8th World Multi-Conference on Systemics, Cybernetics and Informatics*, July 18-21, 2004 - Orlando, Florida, USA, SCI 2004 Proceedings, Vol. IV, pp.425-428.

[7] Mullat, I. Extremal Monotone Systems. *Automation and Remote Control*, No 5, 1976, pp. 130-139 (in Russian)

[8] Quinlan, J. R. Induction of Decision Trees, *Machine Learning*, (1), 1986, pp. 81-106

[9] Torim, A., Kuusik, R. Problem and Algorithms for Finding the Best Decision, In *WSEAS Transactions on INFORMATION SCIENCE and APPLICATIONS*, Issue 9, Volume 2, September 2005, pp. 1462-1470

[10] Võhandu, L., Fast Methods in Exploratory Data Analysis. In *Transactions of TTU*, No 705, 1989, pp. 3-13