Multidimensional Association Rules on Tensors

Ryohei Yokobayashi, Takao Miura
Hosei University
Japan
ryohei.yokobayashi.2j@stu.hosei.ac.jp
miurat@hosei.ac.jp

ABSTRACT: *In this work, we propose a framework suitable for multidimensional data mining based on tensor. A Tensor Data Model (TDM) provides us with high order data structure and naive description for information retrieval. Among others, we discuss multidimensional rule mining here. Generally, association rule mining (or extraction of association rules) concerns about co-related transaction records of single predicate, and hard to examine the ones over multiple predicates since it takes heavy time- and space- complexities. Here we show TDM allows us to model several operations specific to multidimensional data mining yet to reduce amount of description.*

Keywords: Tensor Data Model, Multidimensional Association Rules, Data Mining

Received: 21 April 2018, Revised 28 May 2018, Accepted 3 June 2018

DOI: 10.6025/jcl/2018/9/3/106-119

1. Introduction

Recently there happens a variety of sensor-devices and LSI memory and we may have many chances to access huge amount of the relevant information easily and quickly through internet. However, because of the wide variety of information, it is not easy to explore them consistently and efficiently. We hardly extract useful knowledge and these information disappear immediately very often.

Techniques to extract these knowledge have been investigated and proposed, they are called *data mining*. Because of its intuitiveness, *association rule mining* has been paid much attention[1]. Association rule mining concerns about co-related transaction records of single predicate, but few about multiple predicates. This research topic is called *multidimensional association rule mining*[3]. Clearly the larger dimensions we have, the more useful information we can extract[7]. That is, the problem gets hard about multiple predicates, because it takes heavy time- and space- complexities. For example, we may have usual association rules: Whenever A's stock goes up, so does B's. And we like to estimate that, whenever A's stock goes up, so does B's *in a few days*. In this case we like to extend the information by adding temporal information. To do that, we extend each stock price with timestamp so we have (*A,* 10*,* 180331) of 3 dimensional tensor structure "company, price, time".

In this investigation we introduce a *Tensor Data Model* (TDM)[13]. TDM is advantageous because we can model multiple aspects (in terms of ranks) from single view so that we can extract multi-dimensional association rules and also we can examine whether the processes and results are really useful or not in an intuitive manner. Several operators to TDM allow us to retrieve, manipulate, transform and modify the contents, Similar to relational algebra or OLAP operators including embedded operations. Here we define several operators for data mining on TDM as well as for data retrieval by which we can model them in an rather abstract manner.

This work contributes mainly to the following 3 points:

(1) We propose a collection of operators for data mining on TDM.

(2) We discuss how to extract multi-dimensional association rules on TDM.

(3) We examine the space complexity.

The rest of the paper is organized as follows. In section 2 we describe TDM, the data description and data manipulation. In section 3, we discuss general frameworks of multidimensional data mining. Section 4 contains how to extract multi-dimensional association rules on TDM. . Section 5 contains an experimental results to see the effectiveness of this idea and we conclude this investigation in section 6.

2. Tensor Data Models

2.1 Describing High Order Data

First of all let us define a *tensor X* as a multidimensional array of data whose elements (or *cells*) are referred by multiple indices. The number of indices is called *order* of the tensor. Formally, given data structure $\mathcal{T}$ which is a real-values $\mathcal{R}$, we have a tensor X of order N as:

$$X = \{X_{e_1,..,e_N} | e_j \in I_j, j = 1,..,N\} \subseteq \mathcal{T}^{I_1 \times I_2 \times ... \times I_N}$$

Here $X_{e1, .., eN}$ is a cell in $\mathcal{T}$ indexed by $e_1, .., e_N$, $e_j \in I_j = \{1, .., |I_j|\}$. The k-th index I_k is called k-th dimension and k is *mode* of X. We also say $X \in \mathcal{T}[I_1, I_2, ..., I_N]$. Note there exists no duplicate value with same indices $e_1, .., e_N$, that is, X is functionally determined on $I_1 \times I_2 \times ... \times I_N$. A (conventional) vector is a tensor of order 1 and a matrix of order 2. A *high* order tensor means the one of $N \geq 3$ usually.

For instance, we show amount of monthly sales per item in a matrix format as in a figure 1 (left). A tensor D of order 3 may contain amount of monthly sales per item and store as illustrated in a figure 1 (right). The tensor of order 3 is defined over $STORE \times ITEM \times MONTH$ where $STORE = \{store1, store2\}$, $ITEM = \{item1, item2, item3\}$ and $MONTH = \{January, July\}$.

2.2 Basic Operations for Tensor Data

Let us introduce several operators over tensors[9][13]. We define *projection, selection*, *add, subtract*, *product* over tensors and *multiply* over tensor and matrix. Moreover, for the purpose of data mining, we introduce *transform* of orders, *reduction* and

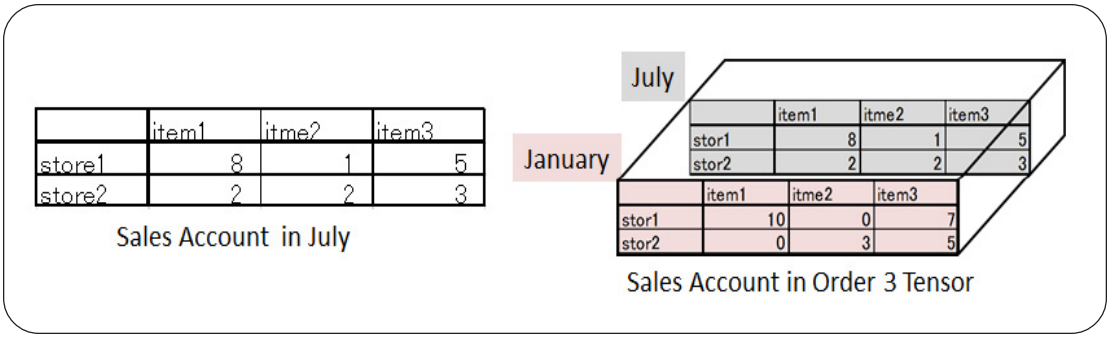

Figure 1. Amount of Sales

recover of dimensions, *vectorize* and some other operators.

Let $X \in \mathcal{T}[I_1, I_2, ..., I_N]$. First we define two database operators, *projection* and *selection*. Let $A, B, .., C$ be distinct modes or modes indexed by dimensions. *Projection* of X over $A,B, .., C$, denoted by $Y = \pi_{A, B,.., C}(X)$, is a subset of X defined as :

$$\begin{aligned} \pi_{A,B,..,C}(X) \quad = \quad & \{Z_{j_1,j_2,..,j_c}\}, j_1 \in A, j_2 \in B, ..., j_c \in C \\ & Z_{j_1,..,j_c} = X_{i_1,..,i_N}[j_1, ., j_c] \\ & \text{and } \ X_{i_1,..,i_N} \in X \end{aligned}$$

Note $Y \in \mathcal{T}[A, B, ..., C]$ and that there might be several cells on $j_1, ., j_c$ in which case we can't define a tensor consistently[1].

Selection of X with conditions C, denoted by $Y = \sigma_C(x)$, is a subset of X defined as :

$$Y = \sigma_C(X) = \{X_{i_1,...,i_N}\} \quad C(X_{1_1,...,i_N}) = true$$

Note that $Y \in \mathcal{T}[I_1, I_2, ..., I_N]$ and that the condition C holds for $I_1 \times ... \times I_N$. We might have an empty tensor if no index value satisfies the condition.

For example, we can obtain the matrix in the figure 1 (left) by $\sigma_{MONTH} = \text{“}July\text{”}(D)$. The combination provides us with a new tensor in the table 1 by $D_3 = \pi_{STOTE, ITEM}(\sigma_{MONTH} = \text{“}July\text{”}(D))$. Let us note that $D_{1,2} = \pi_{STOTE, ITEM}(D)$ can't be well-defined as a tensor of order 2, because there exist 2 values, 10 and 8, at $(store1, item1) \in STOTE \times ITEM$.

	item1	**item2**
store1	8	1
store2	2	2

Table 1. Projection and Selection

Let $X, Y \in \mathcal{R}[I_1, I_2, ..., I_N]$ and we introduce numerical operators over tensors[9][13]. Let us define $X + Y$ (*add*) as an element-wise operation. We can define $X - Y$ (*subtract*), $X * Y$ (*multiply*) and X/Y (*divide*) in a similar manner:

$$\begin{aligned} X + Y \quad & Z_{i_1,..,i_N} = X_{i_1,..,i_N} + Y_{i_1,..,i_N} \\ X - Y \quad & Z'_{i_1,..,i_N} = X_{i_1,..,i_N} - Y_{i_1,..,i_N} \\ X * Y \quad & Z'_{i_1,..,i_N} = X_{i_1,..,i_N} * Y_{i_1,..,i_N} \\ X/Y \quad & Z'_{i_1,..,i_N} = X_{i_1,..,i_N} / Y_{i_1,..,i_N} \end{aligned}$$

We also introduce numerical functions *Max, Min*:

$$\begin{aligned} Max(X,Y) \quad & Z_{i_1,..,i_N} = \max(X_{i_1,..,i_N}, Y_{i_1,..,i_N}) \\ Min(X,Y) \quad & Z'_{i_1,..,i_N} = \min(X_{i_1,..,i_N}, Y_{i_1,..,i_N}) \end{aligned}$$

Given $X, Y \in \mathcal{T}[I_1, I_2, ..., I_N]$, *Inner Product* $\langle X, Y \rangle$ (or, $X \cdot Y$) is defined recursively as a real-value over element-pairs:

$$\langle X, Y \rangle = X \cdot Y = \sum_{i=1}^{I_1} \sum_{i=2}^{I_2} ... \sum_{i=N}^{I_N} X_{i_1,i_2,...,i_N} \times Y_{i_1,i_2,...,i_N}$$

[1]Note *null* is different from “0” or blank, for instance.

Norm of X, $||X||$, is defined as $\sqrt{X \cdot X}$. Note the inner product generates a multidimensional correlation over two tensors.

Let $x_{(j)} \in R^{I_j}, j = 1, .., N$ is a tensor of order 1 (or a vector), *composition* of $\{ x_{(1)}, .., x_{(N)} \}$, denoted by $x_{(1)} \otimes , ..., x_{(N)}$ is a tensor $X \in R[I_1, .., I_N]$:

$$X = x_{(1)} \otimes \ ... \otimes x_{(N)}$$

Here $X_{i_1 ... i_N} \in X$ is defined as $x_{(1)i_1} \times ... \times x_{(N)i_N}$. We call X as a composed tensor of $x_{(j)} \in R^{I_j}, j = 1, .., N$.

If X, Y have composed tensors such that $X = x_{(1)} \otimes ... \otimes x_{(N)}$ and $Y = y_{(1)} \otimes ... \otimes y_{(N)}$ respectively, we define *Product* (or, outer product) of tensors X, Y, denoted by $Z = X \otimes Y \in \mathcal{T}[I_1, I_2, ..., I_{N_X N_Y}]$, as follows:

$$X \otimes Y = \{ Z_{i_1, .., i_{N_X N_Y}} \}$$
$$Z_{i_1, .., i_{N_X N_Y}} = x_{(1)i_1} ... x_{(N_X)i_{N_X}} y_{(1)i_1} ... y_{(N_Y)i_{N_Y}}$$

For instance, let us define two tensors X and Y .

$$X = \sigma_{MONTH="January"}(D)$$
$$Y = \sigma_{MONTH="July"}(D)$$

Then, the inner product $X \cdot Y$ means (extended) sales correlation of *January* and *July*.

As for relationship among a tensor X, a matrix M, a vector V and a scalar c, let us consider a high order tensor X as a collection of vectors, called *j-th fiber*, if we think of the tensor X whose entries are fixed expect I_n: $X_{i1}, ... , i_{n-1}, *, i_{n+1}, .. , i_N$. A figure 2 contains 3 types of fibers given a tensor of order 3 along each mode. They are intuitively called mode-1 fiber (column fiber, $n = 1$), mode 2 (row fiber, $n = 2$) and mode 3 (tube fiber, $n = 3$) respectively.

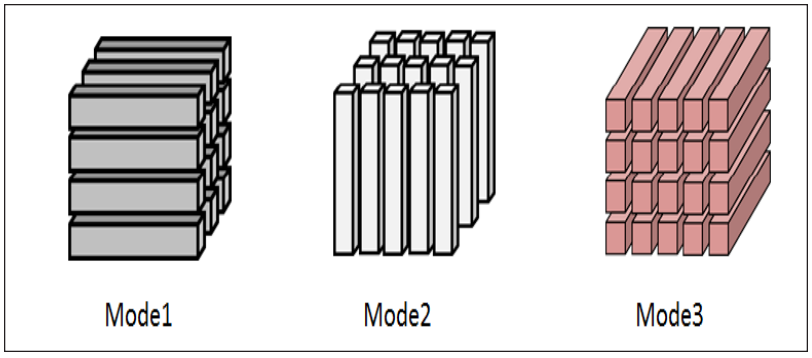

Figure 2. Fiber and Mode

For *matricization* of a tensor, we give the formalization by using mode-n fibers[9]. As illustrated in a figure 3, we consider the two-dimensional slices where the horizontal slice (all the mode 1 fibers), lateral slice (all the mode 2 fibers) and frontal slice (all the mode 3 fibers) as matrices from a tensor of order N.

Let us define *unfold* or *mode-n* matricization. A tensor X can be represented when we unfold the tensor into a matrix. The unfolding X along mode n, denoted by $X_{(n)}$, is a matrix of dimension $I_n \times (I_{n+1} ... I_N I_1 ... I_{n-1})$ considering all the mode-n fibers as vectors. For example, each column of $X_{(n)}$ is a column of X along the n-mode. Note the matrices are denoted by $X_{(1)}, .., X_{(N)}$, of the size $I_n \times (I_{n+1} ... I_N I_1 ... I_{n-1}), n = 1, 2, .., N$.

For example, let us show examples of mode-1/2/3 matricizations in a table 2 applied to the figure 1.

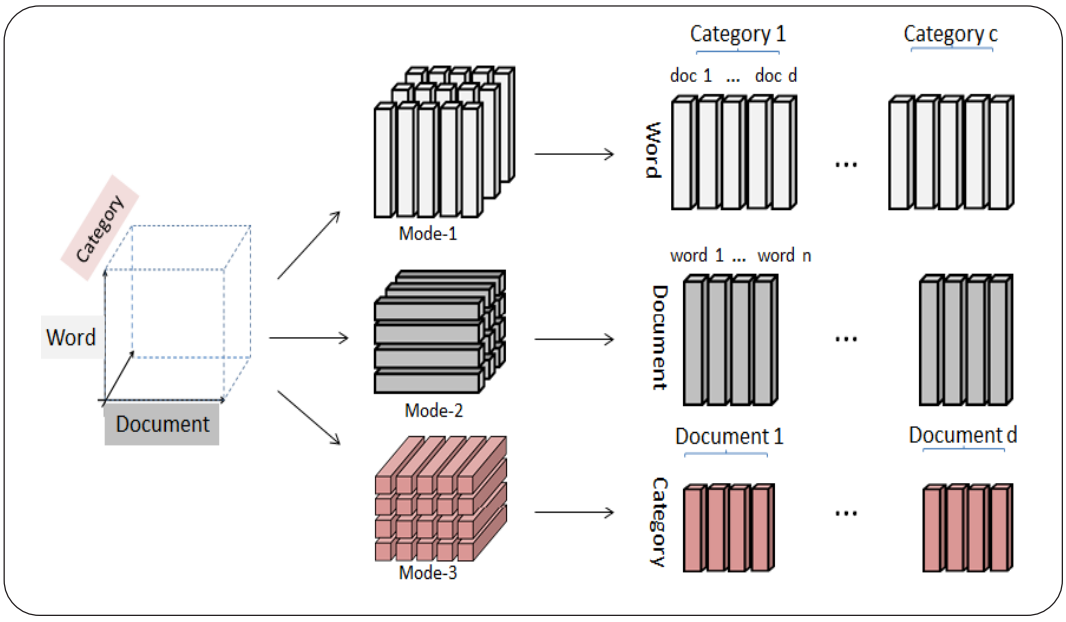

Figure 3. Matricization using Mode-n Fibers

	Jan.item1	Jan.ite2	Jan.ite3	Jul.ite1	Jul.ite2	Jul.ite3
store1	10	0	7	8	1	5
store2	0	3	5	2	2	3

	st1.Jan	st1.Jul	st1.Jan	st2.Jul
item1	10	8	0	2
item2	0	1	3	2
item3	7	5	5	3

	ite1.st1	ite1.st2	ite2.st1	ite2.st2	ite3.st1	ite3.st2
January	10	0	0	3	7	5
July	8	2	1	2	5	3

Table 2. Mode-*n* Matricization, $N = 1, 2, 3$

Now let us define formally *mode-n product* or tensor-matrix multiplication. Given a tensor $X \in \mathcal{T}[I_1, I_2, ..., I_N]$ and a matrix $M \in \mathcal{T}_{Jn} \times In$, we say Y is a tensor of the mode-n product, $Y = X \times_n M$ in $\mathcal{T}[I_1, I_2, ., I_{n-1}, J_n, I_{n+1}, .., I_N]$ where $j_n = 1, .., J_n$:

$$X \times_n M = \{Y_{i_1,..,i_{n-1},j_n,i_{n+1},..,i_N}\}$$
$$Y_{i_1,..,i_{n-1},j_n,i_{n+1},..,i_N} = \sum_{i_n=1}^{I_n} X[i_1,..,i_{n-1},i_n,i_{n+1},..,i_N]M[j_n,i_n]$$

By the definition we mean $Y(n) = MX(n)$ and also we have some properties:

$$X \times_m A \times_n B = X \times_n B \times_m A, \quad m \neq n$$
$$X \times_n A \times_n B = X \times_n (BA)$$

One important special case is *tensor-vector multiplication*. Given a tensor $X \in \mathcal{T}[I_1, I_2, ..., I_N]$ and a vector (or one row matrix) $V \in T_{In}$, the multiplication, $Y = X \times_n V$, is a tensor in $\mathcal{T}[I_1, I_2, ., I_{n-1}, I_{n+1}, .., I_N]$ where

$$X \times_n V = ((Y_{i_1,..,i_{n-1},i_{n+1},..,i_N}))$$
$$Y_{i_1,..,i_{n-1},i_{n+1},..,i_N} = \sum_{i_n=1}^{I_n} X[i_1,..,i_{n-1},i_n,i_{n+1},..,i_N] \cdot V[i_n])$$

This means Y becomes one order smaller than X where the Y value at i_1, .., i_{n-1}, i_{n+1}, .., i_N becomes inner-product with V. We also introduce a *normalized* multiplication $Y = X \times^{o}_{n} V$, which means the multiplication normalized by the fiber lengths.

$$X \times_n^{\circ} V = ((Y_{i_1,..,i_{n-1},i_{n+1},..,i_N}))$$
$$Y_{i_1,..,i_{n-1},i_{n+1},..,i_N} =$$
$$(\sum_{i_n=1}^{I_n} X[i_1,..,i_{n-1},i_n,i_{n+1},..,i_N] \cdot V[i_n])/L$$
$$L = \sqrt{(\sum_{i_n=1}^{I_n} X[i_1,..,i_{n-1},i_n,i_{n+1},..,i_N])^2)||V||}$$

A *tensor-scalar multiplication* can be defined in a straightforward manner. Given a tensor $X \in \mathcal{T}\ [I_1, I_2, ..., I_N]$ and a scalar $c \in R$, let us define $Z = cX$ as $Z_{i1}, ..., {}_{iN} = cX_{i1}, .., {}_{iN}$.

For example, we multiply the tensor $D^{2\times3\times2}$ in the figure 1 by a matrix $M^{2\times3}$, and by a vector V^3 both in mode 2:

$$D \times_2 M = D \times_2 \begin{pmatrix} 1 & 3 & -1 \\ 2 & -4 & 5 \end{pmatrix} = \left(\begin{array}{cc|cc} 3 & 4 & 6 & 5 \\ 55 & 13 & 37 & 11 \end{array}\right)$$

$$D \times_2 V = D \times_2 (\ 1 \quad 3 \quad -1\ ) = \begin{pmatrix} 3 & 4 \\ 6 & 5 \end{pmatrix}$$

Table 3. Matrix/Vector Multiplication

2.3 Data Mining on Tensor Data

Here we introduce 2 operators, *vectorize* and *shift*, mainly for data mining on tensor, and count operations.

Each cell in a tensor may contain a real value so that often we need to transform this cell to a bit value to indicate a given cell is empty or not, and to keep a bit vector along one dimension.

To vectorize a tensor $X \in \mathcal{T}\ [I_1 \times I_2 \times ... \times I_N]$ where $I_k = \{1, ..., |I_k|\}$, we define $Y = \overrightarrow{(X)}$ as $y_{i1...iN} = 1$ if $x_{i1}...i_N) \neq 0, = 0$ otherwise as illustrated in a figure 4.

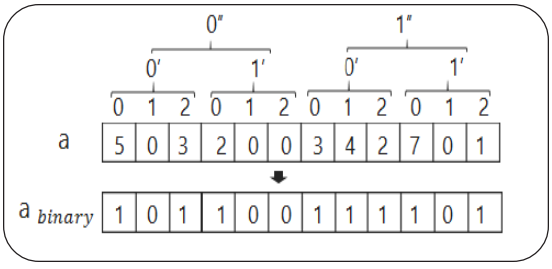

Figure 4. Vectorizing a tensor

A *shift(i)* operator causes shifting elements along a dimension i in a tensor X. That is, *Shift* operator causes one step shift to the right along i-th dimension as illustrated in a figure 5.

$$Z = Shift(X,i) \quad \pi_{i_2,..,i_N}(z_{i_1,i_2,..,i_N}) = \pi_{i_2,..,i_N}(x_{i_1,i_2,..,i_{N-1}})$$
$$= \pi_{i_1}(z_{i_1,i_2,..,i_N}) = 0$$

Finally let us note that we can count all the elements using some combination of the operators. Let Y be a vector E where

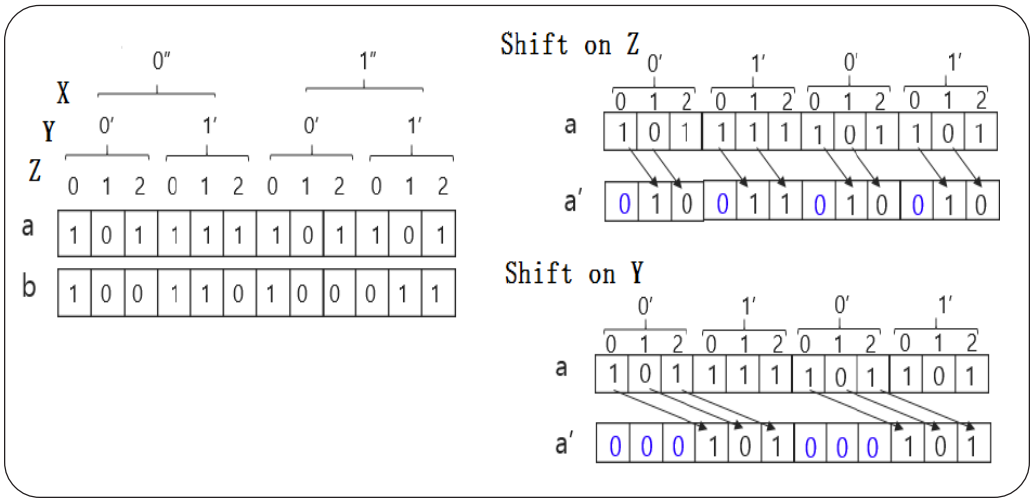

Figure 5. Tensot Shift

all the cells contain 1. Given a tensor X of order 1, $X^{n\times m} \times Y^{m\times 1} = (Z^{n\times 1})$ contains the number of "1" in X, or *count*, as illustrated below:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^t = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

3. Extracting Association Rules on Tensors

Market basket analysis is a typical example for big-data application such as sales information of supermarkets where we might consider transaction data (or an *item set*) as a unit of logical processing. Among others, an *association rule* has been put much attention so far, the rule means a significant conditional probability $P(XY|X)$, denoted by $X \Rightarrow Y$.

After formalizing the issue, we show some extension of the approach for multidimensional sets of items and discuss why we get difficulties here.

3.1 APRIORI

In 1993, Agrawal[1] has proposed an effective algorithm named *APRIORI*, to obtain association rules from a collection D of itemsets over items $\mathcal{I} = \{i_1, i_2, \cdots, i_s\}$: $D \subseteq 2^{\mathcal{I}} - \{\phi\}$. Let us show some example of transactions (and their itemsets) on the left of a table 4.

APRIORI algorithm allows us to extract *association rules*, denoted by $a \Rightarrow b$, which means, if any itemset contains an item a, the itemset also contains b, and if the itemsets containing both a and b appear frequently. Formally we define notions of *support* and *confidence*. Assuming there happen N transactions, let $C(X)$ be the number of transactions containing X.

$$support(X) = \frac{C(X)}{N}$$
$$confidence(Y \mid X) = \frac{C(X,Y)}{C(X)}$$

Given two threshold values σ and ρ called *minimum support* and *minimum confidence* respectively, we extract all the pairs X, Y of itemsets such that $support(X, Y) \geq \rho$ and $confidence(Y|X) \geq \rho$. Note that $X \subseteq Y$ means $C(X) \geq C(Y)$, called *anti monotonicity*. And APRIORI take an advantage of this property to avoid heavy computation. We compute $C(Y)$ only if all the $X \subseteq Y$ satisfy $C(X) \geq \sigma$. Otherwise we don't need to examine it. Similarly, once we have $X \Rightarrow Y$ which means $C(XY)/C(X) \geq \rho$, we must have $X \Rightarrow Z$ if $XZ \subseteq XY$. This is because $C(XZ)/C(X) \geq C(XY)/C(X) \geq \rho$.

3.2 Multidimensional Association Rules

There has been some investigation [7] proposed so far for the purpose of multidimensional association rule mining. In this

subsection, we transform transactions into 2 dimensional data where we introduce a notion of *distance* to extract (2- dimensional) association rules. Let Δ and Δ' be two points in the 2 dimensional space that are indexed by v and u respectively. The points are denoted also by $\langle v \rangle$ and $\langle u \rangle$. We define *relative distance* (Δ, Δ') as $u - v$ to the two points. If we assume to start at an origin 0, or $\langle 0 \rangle$, we can think about relative distance of Δ, denoted by $(0, \Delta)$, as v.

Let us assume that Δ contains an item i_k, denoted by $\Delta(i_k)$. Or we may assume Δ contains a transaction T_k, denoted by $\Delta(T_k)$. Finally we mean all the possible items by I_e and all the possible itemsets by D_e.

Lu[7] introduces a notion of distance into transactions to extract temporal association rules as in a table 4.

Transaction	Items	(Extended) Transaction	(Extended) Items
T_1	a,b,c	$\Delta_0(T_1)$	$\Delta_0(a), \Delta_0(b), \Delta_0(c)$
T_2	c,d,e	$\Delta_1(T_2)$	$\Delta_1(c), \Delta_1(d), \Delta_1(e)$
T_3	a,b	$\Delta_2(T_3)$	$\Delta_2(a), \Delta_2(b)$
T_4	a,b,c,e	$\Delta_3(T_4)$	$\Delta_3(a), \Delta_3(b), \Delta_3(c), \Delta_3(e)$
T_5	b,c,d,e	$\Delta_4(T_5)$	$\Delta_4(b), \Delta_4(c), \Delta_4(d), \Delta_4(e)$
T_6	a,b,e	$\Delta_5(T_6)$	$\Delta_5(a), \Delta_5(b), \Delta_5(e)$

Table 4. Transactions and Items

Starting at Δ_0, we see, for instance, an item a in T_1 (or exactly an extended item $\Delta_0(a)$ in $\Delta_0(T_1)$) with *several* distances 0, 2, 3 and 5. Clearly it is hard to assume that each extended item preserves all the possible distances with its extended transaction.

To extract efficiently yet consistently distance relationship about extended items and transactions, they apply *maxspan* algorithm to this case[5]. The maxspan value (*ms*) is a threshold which we should observe items and distance within the value. Let us note that we should extend notions of support and confidence of itemsets X, Y :

$$support' = \frac{\text{Number of transactions within } ms \text{ containing } X,Y}{\text{Number of total transactions within } ms} \quad (1)$$

$$confidence' = \frac{\text{Number of transactions within } ms \text{ containing } X, Y}{\text{Number of transactions within } ms \text{ containing } X} \quad (2)$$

Let us show E-Apriori [7] to extract 2-dimensional association rules in Algorithm1. Once we take counts on itemsets with each distance, we can obtain 1-itemsets which satisfy both of minimum support and minimum confidence.

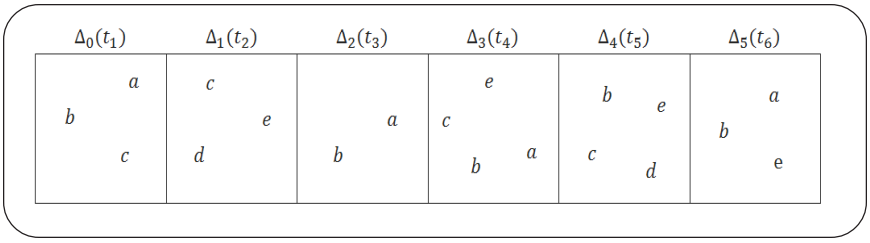

Figure 6. Multidimensional Association Rules

Note that we don't obtain exact values of both support and confidence, because we assume maxspan (*ms*) size on distance and ignore any transactions outsize the windows. For instance, in the figure 6, we have an association rule

$$\Delta_0(a), \Delta_1(c) \Rightarrow \Delta_3(e)$$

Here the rule has *support* = $\frac{2}{5}$ and *confidence* = $\frac{2}{3}$. On the other hand, we must have *support* = $\frac{2}{5-3}$ and *confidence* = $\frac{2}{2}$ which depend on the span value 3 = (Δ_3 - Δ_0). In fact, we examine 3 windows $\{\Delta_0(T_1), \Delta_1(T_2), \Delta_2(T_3), \Delta_3(T_4)\}$, $\{\Delta_1(T_2), \Delta_2(T_3), \Delta_3(T_4), \Delta_4(T_5)\}$ and $\{\Delta_3(T_4), \Delta_4(T_5), \Delta_5(T_6)\}$.

Interestingly Lu [7] has shown that the true values could be approximated by the above *support*′ and *confidence*′, because there should happen huge number of transactions so that we must have few difference and anti-monotonicity.

To obtain 2-itemsets, we generate all the pairs of frequent 1- items and all the items starting at 0″ 0′ 0 as in 1-items. We take counts on an item with relative distances and obtain frequent ones using the minimum support/confidence. As for larger itemsets, we apply the above procedure to obtain frequent itemsets.

4. Multidimensional Association Rule on Tensors

In this section we discuss rule mining on (multidimensional) tensor data model, which can be seen as a natural generalization of Lu[7].

To apply rule mining on tensor, we need *mode-n matricization*, *vectorize*, *shift* and *count* operators/operations described previously.

We summarize the whole algorithm in an Algoritm 2. Let us who how our algorithm goes in the table IV. First we transform the table data by matricization in order to clarify relationship among items. We assume tensors of rank 4 for simplicity. To obtain all the 1-item sets $\Delta_{0''0'0}\{i_n\}$, $n = 1, ..$, we take counts by inner-product of vectorized data and *a* vector 1 and choose the vectors more than minimum support. That is, for the 1-itemset at 0″ 0′ 0, we take any other 1-itemset $\Delta_{s''s's}\{i_n\}$ ($s''s's \neq 0''\ 0'\ 0$), and apply (right) "shift" operator in such a way that $x'' = s''$, $x' = s'$, $x = s$ and take count on the result as illustrated in a figure 7. After taking counts with all other itemsets and examining support and confidence values, we obtain frequent 1-item sets.

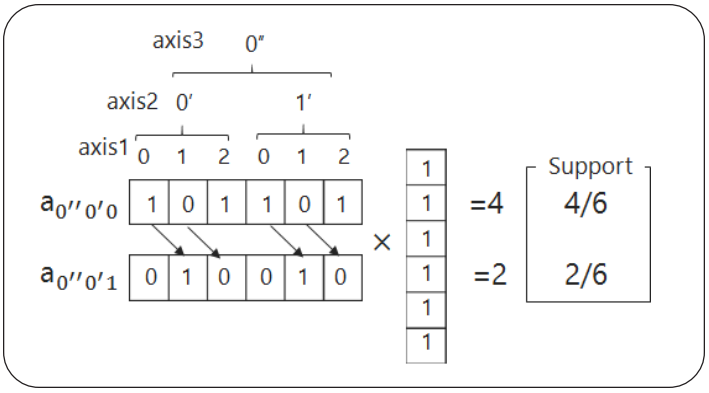

Figure 7. 1- *itemsets*

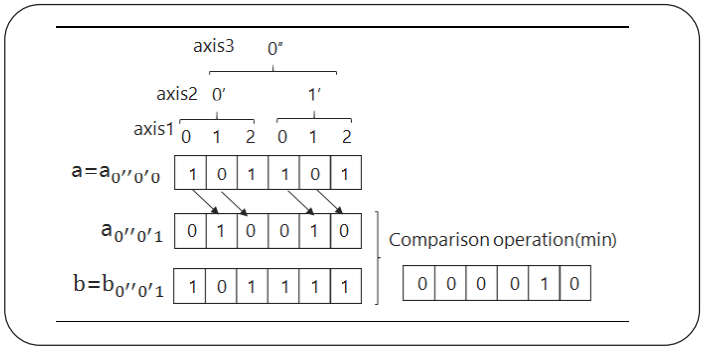

Figure 8. 2- *itemsets*

As for 2 or more itemsets, let us generate k-itemsets C_k from frequent (k - 1) itemsets $F_{k\text{-}1}$. For each k - 1 itemset appeared in vectorized data $\Delta_{0''\,0'\,0}\{a\}$ and $\Delta_{0''\,0'\,1}\{a, b\}$, we apply (right) "shift" operator to the vector a and b to adjust **a** to 0″ 0′ 1 from 0″ 0′ 0, so that we can consider two vectors as the ones of same dimensions even if they have different distance, and extract association rules. See a figure 8. Finally we examine both support and confidence by taking counts to obtain frequent itemsets.

5. Experiments

Here let us discuss some experimental results to show how well our approach works in terms of computing complexity and data description length compared to E-Apriori. We examine weather data in 2000 by meteorological satellite *Himawari* containing altitude, area and temporal information.

5.1 Preliminaries

In our experimental data *Himawari* 2000, all the information have been collected every day in 2000 through Automated Meteorolrgical Data Acquisition System (*AMeDAS*). We take 3 axis, *period, area* and *altitude* with distance and construct a tensor of order 3.

Area consists of 8 regions: Hokkaido, Tohoku, Kanto, Hokuriku, Tokai, Kinki, ChugokuShikoku and Kyushu region. *Altitude* consists of 8 elevation sections where we divide altitude 0*m*- -240*m* every 30*m*. We select observation points randomly from each section considered as transactions. *Period* means a time zone of April 1 to April 10. Given 7 as maxspan value, we examine 4 groups of April 1 to 7, 2 to 8, 3 to 9 and 4 to 10 where each group satisfies maxspan threshold.

We examine attributes of items appeared in transactions: daily precipitation, sunshine hours, maximum temperature, minimum temperature, average wind speed and maximum wind speed. We describe these values by quantification as illustrated in a table 5. Each transaction is a vector of 25 dimension where each dimension contains 1 (YES) or 0 (NO).

E-Apriori assumes single element in every attribute so that transactons have 6 attributes over YES/NO value.

As said previously, it takes heavy time and space complexities. Here we examine both time-complexity and space complexity to show TDM allows us to model several operations specific to multidimensional data mining yet to reduce amount of description. In this experiment, we compare our tensor approach and E-apriori with each other.

daily precipitation (mm)	0, 1∼
sunshine (hours)	0 ∼3, 3∼6, 6∼9, 9∼
maximum temperature (deg.C)	∼0, 0∼10, 10∼20, 20∼30, 30∼
minimum temperature (deg.C)	∼10, -10∼0,0∼10, 10∼20, 20∼
average wind speed (m/s)	0∼3, 3∼6, 6∼9, 9∼12, 12∼
maximum wind speed (m/s)	0∼3, 3∼6, 6∼9, 9∼12, 12∼

Table 5. Quantification

5.2 Results

For association rule mining, it take same time complexity in both approaches. Assuming 0.80 and 0.50 as minimum confidence/ minimum support respectively, let us show how many rules we obtain in table1 6 and 7.

We have extracted 1211 (multidimensional association) rules in total. Here are some example rules extracted:

maximum temperature 10 ~ 20 in Kanazawa ⇒ minimum temperature ~ -10 next day

maximum temperature 0 ~ 20 in Wakkanai "and"

minimum temperature ~ -10 in Rokkasho ⇒ minimum

temperature ~ - 10 in Wakkanai next day

There happens no rule over more than 3 days and the attributes could be affected within 2 days in this case.

No.Items	1	2	3	4	5	6	7	8	9	10
1∼7	93	422	919	926	463	153	44	10	0	0
2∼8	84	357	701	660	315	93	21	3	0	0
3∼9	83	393	821	808	409	148	58	24	7	1
4∼10	75	311	601	532	196	29	0	0	0	0
CommonItems	73	267	442	335	109	15	0	0	0	0

Table 6. Number Of Association Rules Per Itemset

City	**Number of Rules**
Sapporo	671
Hakodate	633
Akita	200
Sendai	904
Rokkasho	671
Tokyo	113
Yokohama	240
Kanazawa	20

Table 7. Number Of Association Rules Per City

5.3 Discussion

First of all let us show the data length of *Himawari* in a table 8. To compare description contents, we transform Eapriori algorithm into an algorithm 3 to generate efficiently more than 3 item sets. In our case, E-apriori requires space 549.8 times more.

	Itemsets (all elements)	Relative Address	Total
E-apriori	$6 \times 8 \times 8 \times 7 = 2,688$	$6 \times \frac{(8\times8\times7)^2-1}{2} = 602,109$	604,792
Tensor	$25 \times 8 \times 8 \times 7 = 11,200$	0	11,200

Table 8. Data Description Length (Tensor and E-apriori)

Let us point out that Algorithm 2 has the part *foreach* (*element* $\Delta_{s''s's}$ (*N*)) which requires time equal to the part *foreach* (*extended transaction* $\Delta_{s''s's}$ (*t*)) in Algorithm 3, because both parts examine all the items in transactions. In other words, they take time of $O(n^3)$.

To describe all the required data, we need the one for all the itemsets and for relative distancees so that our tensor model requires space 11, 200/602, 109 = 0.0186 times (less than 2%).

Important is a fact that, the more the data grows, the less ratio we have. The main reason comes from a fact that we need all the relative distances in E-apriori algorithm within maxspan range. More precisely, each order keeps (*maxspan*) of N'', N', N relative distances, we should have the relative information $(N'' \times N' \times N) + (N'' \times N' \times N\text{-}1) + \ldots + 1$. When there exist M elements, we need space $M \times \frac{\left(N''\times N'\times N\right)^{n-1}-1}{2}$ more for this purpose in E-apriori. Our tensor approach outperfoms whenever the amount of data grows.

Generally, the more orders the tensor holds, the more space we need dramatically for the relative distance in E-apriori. However, tensor description also requires more space in this case. For example, in E-apriori, each "sunshine hour" are described as one item

Algorithm 1 E-Apriori

$\bullet k = 1$
$C_1 = \{\{\Delta_x (i_n)\}\} \mid (i_n \in D) \wedge (0 \leq x \leq maxspan)$
for extended transaction $\Delta_s (t)$ **do**
  for candidate c: $\Delta_x (i_n) \in C_1$ **do**
    c.count++;
  end for
end for
$L_1 = \{c : \{\Delta_x (i_n)\}\} \mid (c \in C_1) \wedge (c.count \geq support)$

$\bullet k = 2$
$C_2 = \{\Delta_0 (i_m), \Delta_x (i_n)\} \mid (\Delta_0 (i_m) \in L_1) \wedge$
$(\Delta_x (i_n) \in L_1) \wedge ((x \neq 0) \vee (x = 0 \wedge i_m < i_n))$
for extended transaction $\Delta_s (t)$ **do**
  for candidate c: $\Delta_0 (i_m), \Delta_x (i_n) \in C_2$ **do**
    c.count++;
  end for
end for
$L_2 = \{c : \{\Delta_0 (i_m), \Delta_x (i_n)\}\} \mid (c \in C_2) \wedge (c.count \geq support)$

$\bullet k > 2$
for $(k = 3; L_{k-1} \neq \emptyset; k++)$ **do**
  $C_k = E - Apriori - Gen (L_{k-1})$;
  for $(o = 1; o \neq k; o++)$ **do**
    $G_\emptyset = E - Group (C_k, o)$;
    for extended transaction $\Delta_s (t)$ **do**
      $G_{\Delta_s(t)} = E - Group (C_o, \Delta_s)$;
      for candidate c: $\{\Delta_{x_1} (i_1), \ldots, \Delta_{x_k} (i_k)\} \in G_{\Delta_s(t)}$ **do**
        c.count++;
      end for
    end for
  end for
  $L_k = \{c : \{\Delta_{x_1} (i_1), \ldots, \Delta_{x_k} (i_k)\}\} \mid (c \in C_k) \wedge (c.count \geq support)$
end for
$L = \bigcup_k L_k$

such as 0~3 so that this kind of data becomes dense, while in tensor approach, we describe one sparce vector such as (0~3, 3~6,· · ·...) and we must have more space.

6. Conclusion

In this investigation we have proposed a tensor data model for th epurpose of multidimensional association rule mining. We have introduced several operators such as *vectorize* and *shift* specific to the data mining operations. Then we have described a new algorithm within this framework and shown that simple algorithm works well based on the model. In our experiment, we examined weather data and extracted 1211 rules. Compared to a conventional E-apriori algorithm, we have shown that much less amount of space is enough for the data description.

References

[1] Agrawal, R., Imielinski, T., Swami, A. (1993). Database mining: A performance perspective, *IEEE Transactions on Knowledge and Data Engineering* 5 (6) 914-925.

[2] Gao, C., Wang, J. (2010). Direct mining of discriminative patterns for classifying uncertain data, *In*: Proceedings16th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, *ACM*, 2010.

Algorithm 2 Multidimensional Rule Mining on Tensor

$\bullet k = 1$
$C_1 = \{\{\Delta_x (i_n)\}\} \mid (i_n \in D) \wedge (0 \leq x \leq maxspan)$
Matricization (*with respect to item axis*)
for element $\Delta_{s''s's} (N)$ **do**
  for candidate c: $\Delta_{x''x'x} (i_n) \in C_1$ **do**
    c.dotproduct;
  end for
end for
$L_1 = \{c : \{\Delta_{x''x'x} (i_n)\}\} \mid (c \in C_1) \wedge (c.dotproduct \geq support)$

$\bullet k \geq 2$
$C_2 = \{\Delta_{0''0'0} (i_m), \Delta_{x''x'x} (i_n)\} \mid (\Delta_{0''0'0} (i_m) \in L_1) \wedge$
  $(\Delta_{x''x'x} (i_n) \in L_1) \wedge ((x''x'x \neq 0''0'0)) \vee (x''x'x = 0''0'0 \wedge (i_m < i_n))$
for $(k = 2; L_{k-1} \neq \emptyset; k++)$ **do**
  for element $\Delta_{s''s's} (N) \mid (0 < s'' < x''_{maxspan}, 0 < s' < x'_{maxspan}, 0 < s < x_{maxspan})$ **do**
    for candidate c: $\Delta_{0''0'0} (i_m), \Delta_{x''x'x} (i_n) \in C_2$ **do**
      right shift$\{\Delta_{0''0'0} (i_m), \Delta_{x''x'x} (i_n)\} \Rightarrow$
        $\{\Delta_{max(0,x'')max(0,x')max(0,x)} (i_m), \Delta_{max(0,x'')max(0,x')max(0,x)} (i_n)\}$;
      c.min$\{\Delta_{max(0,x'')max(0,x')max(0,x)} (i_m), \Delta_{max(0,x'')max(0,x')max(0,x)} (i_n)\}$;
      c.vectorize;
      c.dotproduct;
    end for
  end for
  $L_k = \{c : \{\Delta_{x_1} (i_1), \ldots, \Delta_{x_k} (i_k)\}\} \mid (c \in C_k) \wedge (c.dotproduct \geq support)$
end for
$L = \bigcup_k L_k$

Algorithm 3 Adjusted E-apriori

$\bullet k = 1$
$C_1 = \{\{\Delta_x (i_n)\}\} \mid (i_n \in D) \wedge (0 \leq x \leq maxspan)$
for extended transaction $\Delta_{s''s's} (t)$ **do**
  for candidate c: $\Delta_{x''x'x} (i_n) \in C_1$ **do**
    c.count++;
  end for
end for
$L_1 = \{c : \{\Delta_{x''x'x} (i_n)\}\} \mid (c \in C_1) \wedge (c.count \geq support)$

$\bullet k \geq 2$
$C_2 = \{\Delta_{0''0'0} (im), \Delta_{x''x'x} (in)\} \mid (\Delta_{0''0'0} (i_m) \in L_1) \wedge$
  $(\Delta_{x''x'x} (i_n) \in L_1) \wedge (x''x'x \neq 0''0'0) \vee (x''x'x = 0''0'0 \wedge (i_m < i_n))$
for $(k = 2; L_{k-1} \neq \emptyset; k++)$ **do**
  for extended transaction $\Delta_s (t)$ **do**
    for candidate c: $\Delta_{0''0'0} (i_m), \Delta_{x''x'x} (i_n) \in C_2$ **do**
      c.count++;
    end for
  end for
  $L_k = \{c : \{\Delta_{x_1} (i_1), \ldots, \Delta_{x_k} (i_k)\}\} \mid (c \in C_k) \wedge (c.count \geq support)$
end for
$L = \bigcup_k L_k$

[3] Han, J., Pei, J., Kamber, M. (2011). Data mining: Concepts and techniques, Elsevier, 2011.

[4] Lieven, De L., Moor, B. De and Vandewalle, J. (2000). A Multilinear singular value decomposition, *SIAM journal on Matrix Analysis and Applications* 21 (4) 1253-1278.

[5] Lee, A. J. T., Wang, C-S., Weng, W-Y., Chen, Y-A., Wu, H-W. (2008). An efficient algorithm for mining closed inter-transaction itemsets, *Data and Knowledge Engineering (DKE)* 66, 68-91.

[6] Lu, H., Han, J., Feng, L. (1998). Stock movement prediction and dimensional inter-transaction association rules, 1998 *ACM SIGMOD* Workshop on Research Issues on Data Mining and Knowledge Discovery, Seattle, WA, USA, ACM, New York, USA.

[7] Lu, H., Feng, L., Han, J. (2000). Beyond intratransaction association analysis: mining multidimensional intertransaction associa tion rules, *ACM Transactions on Information Systems* (TOIS) 18 (4) 423- 454.

[8] Kaski, S. (1998). Dimensionality reduction by random mapping: Fast similarity computation for clustering, Neural Networks Proceedings, 1998. *In:* IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on. 1. IEEE.

[10] Nguyen, K.-N.T. *et al*. (2011). Multidimensional association rules in boolean tensors, *In:* Proceedings 2011 SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics.

[11] Sanguansat, P. (2010) Higher-Order Random Projection for Tensor Object Recognition, *In:* Intn'l Symp. on Communications and Information Technologies (ISCIT). IEEE, 2010.

[12] Sun, J., Tao, D., Faloutsos, C. (2006). Beyond streams and graphs: dynamic tensor analysis, *In:* Proceedings 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM.

[13] Yokobayashi, R. and Miura, T. (2017). Modeling random projection for tensor objects, *In:* Proceedings International Conference on Web Intelligence (WI), ACM.