Behavior Discovery As Database Scheme Design

Takao MIURA
Dept.of Elect.& Elect. Engr., Hosei University
Kajinocho, Koganei, Tokyo, Japan

Isamu SHIOYA
SANNO College, Kamikasuya, Isehara, Kanagawa, Japan

Kohei WATANABE
Dept.of Elect.& Elect. Engr., Hosei University
Kajinocho, Koganei, Tokyo, Japan

Abstract

In this investigation, we propose an inductive classification method to a collection of temporal information which mean behavior. We assume Markov property and discuss Ergodic analysis by which we can extract stationary parts of the behavior. And we discuss a design methodology for database schemes based on the theory.
Keywords – *Temporal Learning and Discovery, Database Design*

1 Motivation

In this work, we consider *changes to databases* as temporal information and discuss how to find behavior *rules* by inductive classification. Also we show a new design methodology to databases as one of the direct applications. Essentially *temporal* data are different from *time series* data since the latter are considered independent with each other such as human heights and earthquake measurements while the former must be influenced from its history. Typical examples are odds of horse races and charts of stock market. When we observe such feed-back from historical data, we have special rules as *fashion, trends, learning* and *lesson*, all these mean typical transition patterns (the way of changes) of information. Formally stated, this is called *Markov property* of transition which is an assumption that there exist common and time-independent patterns of transitions. If we find Markov property of our changes, we extract characteristic aspects on which changes cause interesting (predicative) transitions. Once we get such rules, we can utilize them as new knowledge, for instance we can learn how to design database schemes.

When there exist predicative transitions, it is possible to show that Markov property of behavior may cause *stationary* states. In fact, we can predict behavior patterns that change occurs in a stable manner independent of current status and time. Such Ergodic property improves inductive inference as a stochastic process.

Traditionally design of databases has been considered as one of heuristic arts, in other words, we don't have any evaluation theory based on qualitative or quantitative aspects. Also there exists no way to revisit and follow the design during the growth of database contents, thus there has been no other ways except design from scratch in a top-down manner[1]. However, to our surprise, these stationary patterns of behaviors hint us a new scheme as the form of (convergence)*limit* of transition called *stationary schemes.*

In the following, we will develop our theory for inductive database design that are completely different from traditional design methodology.

Section 2 contains some definitions. In section 3, we introduce the relationship between database design and machine learning. Section 4 contains temporal information from the viewpoint of inductive classification. Then, in section 5, we define how to classify transitions by changes of databases, and we show some experiments in section 6.

0-7695-0756-5/00 $10.00 © 2000 IEEE

2 Inductive Scheme Design

Database scheme is knowledge by which we classify information. According to this knowledge, we identify *concepts* and manage them consistently and uniformly. Formally we say an object d is of a class c, or d belongs to c if d has the property that c carries. In this investigation, we assume every object belongs to just one class (or we assume *single hierarchy* among classes). We call such a model *instance-based* since we assume every object carries its own membership information. In this investigation, we define $\mathcal{C} = \{c_1, .., c_K\}$ as all the classes in this database.

By *database design*, we give description of database scheme. As well-known, it is hard to automate and evaluate database design process. For example, class hierarchy is useful to obtain concise knowledge representation, but there is no systematic way to examine[5, 18].

To manage a collection of a class c efficiently and uniformly, we discuss *appropriateness*[9] in this investigation. By *appropriateness* we mean a condition that there exists some constant n_0 such that the number of objects of a class c is greater than n_0 for every c. Since we assume single class hierarchy, this means all the *lowest* classes are not fine. Readers may see that large sets of objects can be managed efficiently by means of B-trees and hash techniques but fine sets can't be and we have to obtain solutions from design steps. If all the objects are appropriate, we can say granularity of classes in the database is not fine, thus we must have uniform, knowledgeable and efficient classification for query execution/optimization, type-validation and database maintenance[1].

Machine learning from databases is a technique targeted for automatic/semi-automatic modeling to the world of interests. This technique takes input from databases (schemes and a collection of objects) and generate possibly improved schemes. Generally database schemes fail to capture *current* contents very well because it had been designed before, and very often we obtain mismatch between them. The technique might generate alternative ideas of schemes suitable for the contents.

One of the typical examples is an inductive classification for objects E in database. Traditionally a collection $\mathcal{C}$ of classes had been given at the initial stage of database design. We assume a common set of characteristics (attributes) $\mathcal{A} = \{A_1, .., A_m\}$ to $\mathcal{C}$. If we obtain conditions Δ over $\mathcal{A}$ to classify objects E correctly into one of $\mathcal{C}$, we could examine and justify current classification, because we can restate classes by means of Δ. Such method is called *class learning from examples* or inductive classification.

A *decision tree* has been widely applied to a variety of applications. This tree provides us with classification conditions. Given an object d at an intermediate node labeled with an attribute A_i, we examine an attribute value of d (say, $A_i(d)$) and, according to the value, we go to an appropriate branch. We repeat this process until we get to a leaf. Each path from the root corresponds to a conjunctive condition $A_1 = "\alpha_1" \wedge ... \wedge A_l = "\alpha_l"$ to identify answer classes.

To generate trees, if there is only one class c to which the relevant objects belong, we have a leaf of c. Otherwise, we make grouping at one of the attributes that are not yet selected, and we generate an intermediate node of this attribute. Edges are generated according to the attribute values for branching.

The most important problem here is what attribute should be selected to go into lower levels. Well known algorithms such as `ID3` and `C4.5` utilize *entropy*. Entropy is defined by using probability of *events* (class membership in this case), and we regard the ratio of the number of objects of a class c to the total number as the probability. To select attributes, they try to maximize *information gain* that is defined by means of relative entropy[14, 15]. This is the reason why decision trees have been widely utilized because we don't assume any domain knowledge but also sometimes we get uneasy and strange results.

Let us outline some relationship between decision trees and database design. As we said, large sets of objects can be managed efficiently by means of data structure techniques and fine sets can't be. We can relieve this issue only by *merging* or *restructuring* classes. But the solutions come from inspecting semantics of classes and objects, that is, from design steps.

During the growth of databases, the scheme should be evolved because of changes of the object' roles. Unfortunately designated scheme can't always capture the current contents since the scheme should be long-lived. In fact, databases are not flexible while very often information arrives with new intension inside, then it doesn't bring any change of common knowledge into scheme change.

Here *changes* of database mean updates of class

[1]See [9] for multiple class hierarchy.

membership as well as attribute values (characteristic values) to objects. In this investigation we don't discuss any update operation of birth/death of objects nor of scheme evolution. Each change is called a *transition*, and behavior to make changes is called a *transaction*. A few design methodology has been proposed so far[16] mainly from the viewpoint of top-down approach but not of *growth* of database as proposed later. Such situation could be relieved by clustering objects and concept formation[6]. In this work, we will analyze transition patterns *inductively* looking at changes of class membership and attribute values over objects considered as temporal information and put back into database schemes[11].

Characterizing classes by inductive approach should vary according to changes of databases. Then let us define a collection E of objects at time t by E_t, which can be seen as a set of temporal objects or a state at t. Let $E_t \rightarrow E_{t+1}$ be all the changes of temporal objects at time t which also can be seen as *transition* of states. Then inductive analysis of behavior depends on how we classify the set of transactions.

In the following, we discuss only collections of finite, discrete and non-numeric values and their (discrete) changes. That is, we assume every object d can be changed as $A(d)_t \rightarrow A(d)_{t+1}$ where $A(d)_t$ means an attribute value of d at t. One of temporal aspects can be described by *depth of states n*. Given objects E, attributes A_i (i=1,..,n) and $\mathcal{E}$ over $E \times A_1 \times ... \times A_m$, let $\mathcal{E}_t$ be a snapshot of $\mathcal{E}$ at t. Then an inductive classification over $\mathcal{E}_1 \times ... \times \mathcal{E}_n$ can describe transitions to recent n snapshots. Similarly we can think about transition of class membership by inspecting $\mathcal{C}_1 \times ... \times \mathcal{C}_n$ instead of $\mathcal{C}$, where each $\mathcal{C}_t$ means a snapshot of class membership at time t.

3 Markov Property of Temporal Data

Inductive classification by decision tree technique utilizes entropy as a base of induction (prediction). Entropy is defined as $-\Sigma_{i=1}^{k} p_i \log_2 p_i$ where $p_i > 0$ means probability of event i, $i = 1,..,k$ $(k < \infty)$. This value can be considered as average depth of binary decision trees to all the events[11]. To obtain decision trees, we consider the ratio $p_i = n_i/N$ where the number n_i of objects of a class c_i to the total number N of objects. Remember these p_i can be defined to each state $\mathcal{E}_t$. Thus we must have sequence of the probabilities $p_i^{(0)}, p_i^{(1)}, .., p_i^{(t)}, ...$

For temporal information $E_1, .., E_t, ..$, we might have sequences of changes to class membership or changes of attribute values. That is, let $n_{i,j}$ be the number of objects of a_i changed to a_j where a_i is a class or a value, then the transition probability $p(i \rightarrow j)$ can be described as $n_{i,j}/n_i$. Generally this value depends on time t and we can't always expect common patterns. A transition is called *Markovian* (or it has *Markov property*) if the transition probability $p(i \rightarrow j)$ doesn't depend on time but on histories (past states). We say a transition is *Markovian of depth n* if n is the maximum depth of states on which transition depends. Especially the transition of $n = 1$ is called *simple* Markovian. It means that one step transition i to j is determined by i-th and j-th steps, but not dependent on time nor past states before i. If our transition doesn't depend on any past state, it is called *independent* and we must have $p(i_1, .., i_n \rightarrow j) = q(j)$ where $q(j)$ means an initial probability of the event i.

In a case of class membership, a simple Markov transition is described by means of K-square matrix $[p(i \rightarrow j)], i, j = 1..K$, while general Markov transition of depth n by a $K^n \times K$ matrix $[p(i_1, .., i_n \rightarrow j)]$ where $p(i_1, .., i_n \rightarrow j) = n_{i_1,..,i_n,j}/n_{i_1,..,i_n}$, and $n_{i_1,..,i_n}, n_{i_1,..,i_n,j}$ mean the number of objects of a class c_{i_1} changed to c_{i_2} changed to .. to c_{i_n} and the one finally changed to a class c_j respectively where $\mathcal{C} = \{c_1, .., c_K\}$.

Initially an object of c_i is changed to a class c_j with a probability $p(i,j) = q(i) \times p(i \rightarrow j)$. Also $p^{(t)}(i,j)$ means a probability with which that an object of an initial class c_i is changed to c_j after t steps, and obtained by $\Sigma_{k=1}^{K} p^{(t-1)}(i,k) \times p(k \rightarrow j)$. Finally $q^{(t)}(j)$ is a probability that an object belongs to a class c_j after t steps, obtained by $\Sigma_{k=1}^{K} q(k) \times p^{(t)}(k,j)$. In a case of simple Markov transition with the transition matrix $P = [p(i \rightarrow j)]$, $\overline{q}^{(t)} = (q^{(t)}(1), .., q^{(t)}(K))$ can be calculated simply by $\overline{q}^{(0)} \cdot P^t$ where $\overline{q}^{(0)}$ is a vector of initial probabilities of class membership.

Similar discussion holds in the case of attribute values.

Of course we can think about decision trees on each state $\mathcal{E}_t$ and its incremental property[11, 12]. Here let us consider a theory for inductive classification of transition patterns.

When we assume all the transitions are simple Markovian, the transition can be managed and manipulated easily once we get a transition matrix on each attribute and class membership. Then, on a table over $A_1 \times ... \times A_n$, each object is described as a row and $a_{i,j}$ means the probability of changes of the attribute values $a_i \rightarrow a_j$, and the changes $c_{i,j}$ of class membership by $c_i \rightarrow c_j$.

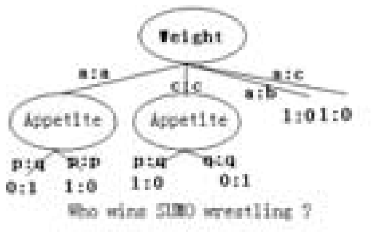
Since we watch the transitions within each attribute and class membership, it seems nice for us to classify the changes over $\mathcal{E}_t \times \mathcal{E}_{t+1}$ inductively. Then we can apply our decision tree technique to this set in an obvious way. Thus we get classification for transition patterns.

EXAMPLE 1 As our running example, let us consider some information about *SUMO* or Japanese wrestling. We assume 3 attributes and 1 class, *Weight*, *Appetite*, *Spirit* and a class *Win*. Weight consists of a (heavy), b (middle) and c (light), Appetite contains p (good) and q (usual) and Spirit contains x (tough) and y (mild). Win consists of 1 (yes) and 0 (no).

Observing changes of many strong SUMO wrestlers, we want to decide which types of wrestlers win continuously. And we examine two states between the wrestlers. Let us describe the change as $b : a$ which means a wrestler of a middle weight becomes heavy. Here are our sample data:

Weight	Appetite	Spirit	Win
a:a	p:p	x:x	1:0
a:a	p:q	x:x	0:1
a:b	p:p	x:x	1:0
a:c	p:p	x:x	1:0
b:a	q:q	x:x	1:0
b:c	p:p	x:x	0:1
c:a	p:p	x:x	0:1
c:b	p:p	x:x	0:1
c:c	p:q	x:x	1:0
c:c	q:q	x:x	0:1

Considering all the values as units, we can analyze them as usual. Te maximum gain can be obtained by looking at Weight first. Then result tree talks about "Who wins SUMO wrestling ?".



For example, a transition of $a : a$ on Weight and Appetite $p : q$ happens, then he could win at next round if he won this round.

Here we have two problems. The first problem is *how* can we design database scheme from the classification of transition patterns ? To do that, we have to analyze behavior in a statistical manner and generate some scheme suitable for updates. The second one is more serious. Although result schemes are nice for the *current* contents, but how about *future* ? We believe hat we should examine the contents and extract the *stationary* parts into the form of schemes. That's why we don't discuss probability of transition but Markov property.

4 Limit Schemes of Databases

In this section we assume simple Markov transitions and discuss how to design schemes based on Ergodic theory.

Let us discuss transition of class membership for a while. According to Ergodic theory of stochastic processes, if we have $p^{(t)}(i \rightarrow j) > 0$ for $t >> 0$ and every i, j, there exist both $\lim_{t\rightarrow\infty} q^{(t)}(i)$ and $\lim_{t\rightarrow\infty} p^{(t)}(j \rightarrow i)$ that are identical and not 0[3]. The solutions can be obtained by $K + 1$ equations (called *limit equations*):

$$X_i = \Sigma_j X_j \times p(i \rightarrow j), i = 1, .., K$$
$$\Sigma_i X_i = 1$$

The solution vector called *limit distribution*, $\overline{q}^{(\infty)} = (.., q^{(\infty)}(i), ..)$, means limit of distribution of class membership of $c_1, .., c_K$ after sufficiently large steps of transitions. Since this vector also means limit distribution of transition, that is, $\lim_{t\rightarrow\infty} q^{(t)}(i) = \lim_{t\rightarrow\infty} p^{(t)}(j \rightarrow i)$, the solution becomes independent of time.

From the viewpoint of database design, limit distribution implies a fact that we can calculate time-independent part of schemes (called *stationary* part of *future* schemes). Of course, we can think only about information of finite time and $q^{(t)}(i) = q^{(\infty)}(i) + v^{(t)}(i)$ must contain a *transient* part $v^{(t)}(i)$. More specifically, the difference includes:

(1) the transient part (i.e., errors which depend on time).
(2) observational errors of $q^{(t)}(i \rightarrow j) > 0$ (i.e., no transition arises).
(3) no simple Markov transition.
(4) probability errors.

We can't avoid the case of (4) but other should be examined. Let us discuss others in the following sections.

EXAMPLE 2 Let us explain limit distribution to our SUMO example. Our class membership transition is described by means of a matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. The limit equation $x_1 + x_2 = 1, x_1 = x_2$ has solution $x_1 = x_2 = 1/2$. W eight attribute has 3× 3 transition matrix :

$$\begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/4 & 1/2 & 1/2 \end{pmatrix}$$

The limit equation has the solution $x_1 = 2/5, x_2 = 1/5, x_3 = 2/5$.

As for Appetite attribute the transition matrix is:

$$\begin{pmatrix} 3/4 & 1/4 \\ 0 & 1 \end{pmatrix}$$

The limit equation has the solution $x_1 = 0, x_2 = 1$.

Finally Spirit attribute has a transition matrix:

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

And the limit equation has no solution.

Unfortunately, in the above observation, we can't see what happens towards transition to ∞ state and we may have no solution of limit equations when some transitions are impossible among classes[2]. However, by using $\mathcal{Z}$ transformation, it is possible to obtain transient parts as well as stationary parts[4] even if $\Sigma_j p(i \to j) = 0$ for some i. In the following let us summarize the calculation issue.

Given $f(t)$ for time t as values $f(0), f(1), .., F(z) = \Sigma_{t=0}^{\infty} f(t) z^t$ is called *$\mathcal{Z}$ transformation* of f. This function has a unique inverse transformation if $|z| < 1$. Let us restate our limit equations to $F(z)$ by $\mathcal{Z}$ transformation with v ector coefficients. Then the function satisfies $z^{-1}(F(z) - \overline{q}) = F(z) \cdot P$ and we must hav e $F(z) = \overline{q} \cdot (\overline{I} - zP)^{-1}$. Since there exists an inverse matrix $(\overline{I} - zP)^{-1}$, t-th coefficient $\overline{H}^{(t)}$ contains stationary and transient parts[3]: $\overline{H}^{(t)} = P^{(\infty)} + P'^{(t)}$.

Remember $\overline{q}^{(t)} = \overline{q}^{(0)} \cdot \overline{H}^{(t)}$, but once we get a stationary part $P^{(\infty)}$ of $\overline{H}^{(t)}$, we have $\overline{q}^{(\infty)} = \overline{q}^{(0)} \cdot P^{(\infty)}$, that is, the num ber of objects of c_i at time t becomes $\overline{q}^{(0)}(i) \cdot P^{(\infty)}$. If it is completely Ergodic, this result doesn't depend on $\overline{q}^{(0)}$.

EXAMPLE 3 Let us analyze some cases in our running example.

[2]This is called *non-complete* Ergodic process where there might be no object in a class and $\Sigma_j p_{i,j} = 0$ for some i.

[3]If P has no zero row, $F(z)$ has $(1/(1-z)$ as an eigen value, but there might be no stationary part otherwise.

(1) First of al, let us see the Appetite attribute. The transition matrix P is:

$$\begin{pmatrix} 3/4 & 1/4 \\ 0 & 1 \end{pmatrix}$$

Then $\overline{I} - z \cdot P$ becomes:

$$\begin{pmatrix} 1 - (3/4)z & -(1/4)z \\ 0 & 1 - z \end{pmatrix}$$

The inverse matrix $(\overline{I} - z \cdot P)^{-1}$ becomes :

$$\frac{1}{1-z} \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} + \frac{4}{4-3z} \begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix}$$

And t-th term of the po wer series at time$t$, $\overline{H}^{(t)}$ becomes :

$$\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} + (3/4)^t \begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix}$$

If $\overline{q}^{(0)} = (1, 0)$, then we have $\overline{q}^{(t)} = ((3/4)^t, 1 - (3/4)^t)$, if $\overline{q}^{(0)} = (0, 1)$, then we have $\overline{q}^{(t)} = (0, 1)$, and if $\overline{q}^{(0)} = (1/2, 1/2)$, then we have $\overline{q}^{(t)} = ((1/2) \cdot (3/4)^t, 1 - (1/2) \cdot (3/4)^t)$. In any cases, we have $\overline{q}^{(\infty)} = (0, 1)$.

(2) Winner class can be described by $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Then $\overline{I} - z \cdot P$ becomes $\begin{pmatrix} 1 & -z \\ -z & 1 \end{pmatrix}$. We have the inverse matrix $(\overline{I} - z \cdot P)^{-1}$:

$$\frac{1}{1-z} \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix} + \frac{1}{1+z} \begin{pmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{pmatrix}$$

And t-th term of the po wer series at time$t$, $\overline{H}^{(t)}$:

$$\begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix} + (-1)^t \begin{pmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{pmatrix}$$

This result says that the transition has periodic property and is not convergent although we have a stationary part (the solution of limit equation).

(3) Spirit attribute has no stationary part since the inverse matrix $(\overline{I} - z \cdot P)^{-1}$ becomes: $\frac{1}{1-z} \begin{pmatrix} (1-z) & (-z) \\ 0 & 1 \end{pmatrix}$ and has no stationary part in t-th coefficient.

Finally let us discuss how to mak e decision trees from stationary parts of (limit) schemes. First of all, we assume we have $P^{(\infty)}$ matrix, a stationary part of class membership transition of a scheme.

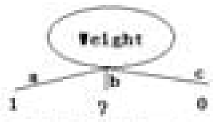
Although we don't assume birth and death of objects, the total number $n^{(\infty)}$ of objects may decrease because only *stationary classes* $\mathcal{S} = \{c_1, .., c_k\}$ are considered. Now our goal is how to generate decision trees for stationary classes. Remember there is no change to objects of stationary classes since we don't assume birth and death of objects. Let $n_1^{(\infty)}, .., n_k^{(\infty)}$ be the numbers of objects of the stationary classes. Let us extract all the rows E' of the stationary classes from the initial step E_0.

By examining all the attributes of the objects of the stationary classes, we can generate decision trees at E_0 that is *not* our goal. Again we apply our theory to transitions of all the values $dom(A_i)$ to every attribute A_i, and we will obtain *stationary values* of each attribute domain. Let $R = [a_{i,j}]$ a transition matrix defined over $dom(A_i) \times dom(A_i)$ where $a_{i,j}$ means a_i is changed to a_j with the probability $a_{i,j}$. It may contain zero rows since $\Sigma_j a_{i,j} = 0$ when no value is changed to a_i. By obtaining $E'[A_i]^{(0)} \cdot R^{(\infty)}$, we can get the total number of stationary values. When there exists no stationary value, we ignore the attribute because we don't have any *hook* at time ∞. There remain *stationary attributes* and we project E' over the stationary attributes. We call it E' again.

Now let us describe how we generate *limit decision trees.* First of all, we have the entropy of class membership in limit: $\log_2 n^{(\infty)}$ - $(1/n^{(\infty)})\Sigma_{i=1,..,k} n_i^{(\infty)} \log_2 n_i^{(\infty)}$

Although we obtain all the numbers $n^{(\infty)}$ and $n_i^{(\infty)}$, we don't always obtain limit states of individual objects at every branch of our limit decision tree. In this investigation, we adjust each number of the occurrences appeared in several parts of the tree based on $P^{(\infty)}$ of the total class membership. For instance, let W be a collection of objects ($W \subseteq E'$) where A value are equal to a (A is a stationary attribute), m be the number of objects in W, and m_i be the number of objects of c_i in W. Then we define $m_i^{(\infty)}/m^{(\infty)}$ as $(m_i/m) \times (n_i^{(\infty)}/n^{(\infty)})$. Eventually we will obtain limit schemes of classes, their sizes and classification conditions Δ^{∞}.

EXAMPLE 4 Let us continue to discuss our SUMO example. As stated previously , all the classes 1 (yes) and 0 (no) classes are stationary. However, as stated previously too, Spirit attribute is not stationary any more, and Appetite becomes singleton (only p). Thus there remains only one attribute Weight to our limit decision tree. The values a, b and c are stationary. Since we have $E = \{(a,1),(a,1),(a,1),(a,0),(b,1),(b,0),(c,0),(c,0),(c,0),(c,1)\}$ as an initial state, we obtain the tree.



Limit Decision Tree

5 Experimental Results

We examine a set of score data of students in a class room. We adopted the information since the students get used to *atmosphere* and *communication* during the class management, the data must be historically sensitive. Assuming simple Markov transitions and let us calculate limit decision trees.

In our class room there are 47 students, each record consists of student ID and 5 histories of evaluation. Each evaluation contains roll, report, examination and result. We assume *Roll* information contains 1 and 0, *Report* contains 1 and 0, and *Examination* includes 0,1,.., and 10 which we will divide by 2, and *Result* contains classes `a,b,c,d,e,f`. Considering Result as object classes, we analyze other 3 attributes: For example, the first record describe a student 63142, he attended 4 times, submitted 5 reports and he got scored "1,3,2,3,0".

```
63142 1:1:1:a 1:1:3:a 1:1:2:b 1:1:3:b 0:0:0:b
64159 1:1:1:a 1:0:4:c 1:0:3:c 0:0:0:f 0:0:0:f
65265 0:1:1:d 1:1:4:a 1:1:4:a 1:1:8:a 1:1:3:b
67269 1:0:0:f 1:0:1:c 0:0:2:e 0:0:0:f 0:0:0:f
68557 1:1:1:a 1:1:4:a 1:1:3:a 1:1:0:f 1:1:1:f
72015 1:1:1:a 1:1:2:a 1:1:2:b 1:1:8:a 1:1:1:f
72022 1:1:1:a 1:1:4:a 1:1:1:f 1:1:6:a 1:1:1:f
72057 1:1:1:a 1:1:4:a 1:1:5:a 1:1:10:a 1:1:5:a
72098 1:1:1:a 1:1:3:a 1:1:4:a 1:1:5:a 1:1:1:f
72099 0:0:1:e 1:0:3:c 1:0:2:c 0:0:0:f 0:0:0:f
........
```

First of all, we examine the transition from the first state to the second.

As a first step, we examined *Result* classes. The initial distribution is $(37/47, 2/47, 7/47, 0, 0, 8/47)$. The transition matrix becomes:

$$\begin{pmatrix} 35/42 & 2/42 & 4/42 & 0 & 0 & 1/42 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 2/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/1 & 0 & 0 & 0 \\ 0 & 0 & 2/2 & 0 & 0 & 0 \end{pmatrix}$$

Unfortunately we couldn't see the stationary part of this transition, i.e, we got $\overline{0}$. This means w e can't obtain stationary classes and all the objects are transient.

W e examined the final transition from the state 4 to 5. Again we checked stationary classes: In this state *Result* has distribution as the probability $(23/47, 6/47, 3/47, 0, 0, 15/47)$ and it contains the transition :

$$\begin{pmatrix} 5/23 & 11/23 & 0 & 0 & 0 & 7/23 \\ 0 & 6/6 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 0 & 2/3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15/15 \end{pmatrix}$$

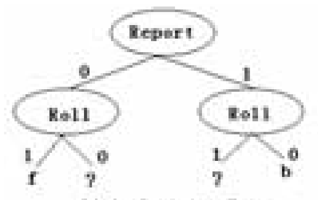
And its stationary part:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

That means there are 2 stationary classes of objects, b and f in a limit decision tree where each class contains 6 and 15 students respectively.

And according the classes, w e analyzed the relevant rows. *Roll* has the limit distribution of $(4/21, 17/21)$, and the transition matrix is $\begin{pmatrix} 4/4 & 0 \\ 2/17 & 15/17 \end{pmatrix}$. And we have the stationary parts of *Roll* as $\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$. Thus the limit distribution is $(4/21, 0)$. We have 1 as only one stationary value.
Similarly, *Report* attribute has the distribution $(5/21, 16/21)$ and the transition matrix is $\begin{pmatrix} 5/5 & 0 \\ 2/16 & 14/16 \end{pmatrix}$. We have the stationary parts of *Report* as $\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$. The final distribution becomes $(5/21, 0)$. As before, we got 1 as only one stationary value. In *Examination* we couldn't obtain a stationary part.

Eventually we have the limit decision tree:



Limit Decision Tree

We have assumed Mark ov transitions in this experiment. Initially it seems for the students to feel uneasy to a teacher and his class management. This might be true for the teacher. That is why we couldn't obtain the stationary part of our limit scheme.

However, after a while, half of the students became *stable*, i.e., they were expected to get score of b or f constantly. This w as the same feeling of the teacher. The classification became trivial but was obtained inductively by inspecting *Report* and *Roll* as indicated by our limit decision tree.

W e got the stationary part of scheme, and the tree says only *sure* part. This is suitable for scheme design. However one of the problems is that, the deeper w e examine states, the better we go to schemes. In this experimen t, we assumed all the transitions were *simple Markovian*, and the difference among the matrices were assumed as errors. Clearly w e ha ve to validate this assumption although this is not easy task. For example, we ha ve to watch all the transition and validate the depth if necessary to see *when* and why we have to revisit depth of Markov memory.

6 Related Works

Knowledge discov ery from databases (KDD) is now centered as one of the importan t and attractive research issues, for sophisticated applications lik e design process and data warehouses[17, 2].

Among others, concept learning from databases is one of the ma jor research topics over the intersection of both databases and knowledge processing. Traditionally the research topic as *scheme evolution* in databases means ho w to obtain *flexible* mechanism of databases at re-design in a top-down manner[1] but lack of feed-bac k from database instances to adjust schemes suitable for *current* databases. *Scheme disc overy* is a technique to obtain schemes incrementally from current schemes and the instances, while *scheme learning* means a technique to generate schemes from collections (clusters) of instances directly. The latter approach might cause accuracy and efficiency since the results depend on formalization of the problems.

When analyzing the description extensively as well as scheme, we might generate new t ypes. Such knowledge discovery process considered databases as description of the meaning is called *type scheme disc overy in databases*. W e could utilize multiple attributes, then we refine the classification. *De cision tree* is one of the ways of type generation[14, 15, 18]. For these years, the authors have discussed scheme disco very of database, and proposed class design[8], scheme generation by means of class approximation and inductive classification[10]. We have already proposed design methodologies based on decision trees called *CSOP* (Conditional Sum Of Products) under the assumptions of class hierarchy[10].

There exists one approach using Markov chains to the mining of time series[7], and they hav e proposed the technique for modeling complex and multi-v ariant applications. But neither Ergodic analysis nor its design feed-back appear.

7 Conclusion

In this investigation we have proposed a way to behavior discovery based on inductive classification. W e have regarded objects' behavior as a collection of temporal information, and under the assumption of Mark ov transition, and we discussed how to extract stationary and transient schemes based on Ergodic theory. Users can separate database schemes into two parts, *stationary* and *variable* ones. The former becomes independent of temporal aspects that we could validate as classical *scheme*, while the latter described *transient* properties of databases which had been embedded into objects' own information so far.

There remain many issues to extend our theory in future. The most importan t one is how we can obtain suitable depth of Mark ov memory. We assumed *simple* Mark ovian in this work, and it is straightforward to extend for general Mark ovian. But, the problem arises how we can obtain the depth efficien tly or how we can examine it is the best during the change of databases. We have proposed efficient methodology for incremental update of decision trees[12] and we guess the technique can be applied to this theory.

References

[1] Elmasri, R. and Navathe, S.: Fundamentals of Database Systems (2nd), Benjamin (1994)

[2] Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. and Uthurusam y, R. (Eds).: Advances in Knowledge Discovery and Data Mining, *MIT Press* (1996)

[3] Feller, W.: An Introduction to Probability Theory and Its Application, (Vol. I, II), John Wiley (1950)

[4] Howard, R.A.: Dynamic Programming and Mark ov Processes, MIT Press (1960)

[5] Ishizuka, M.: Knowledge Representation and Efficient Inference, Maruzen (1995) in Japanese

[6] Langley, P.: Machine Learning and Concept Formation, *Machine L earning* 3 (1989)

[7] Massa, S. et al: A New Modeling T echnique Based on Markov Chains in Some Behavioral Patterns in Event Based Time Series, *proc.DaWaK* (1999)

[8] Miura, T. and Shioy a, I.: Knowledge Acquisition for Classification Systems, proc. *ICTAI* (1996).

[9] Miura, T. and Shio ya, I.: Paradigm for Sc heme Discovery, Intn'l Symposium on Cooperativ e Database Systems for Advanced Applications *(CODAS)* (1996)

[10] Miura, T. and Shio ya, I. Learning Concepts from Databases, *Conference and Workshop of DEXA* (1998)

[11] Miura,T. and Shioya, I.: Inductive Classification of Temporal Objects, proc.*PACRIM* (1999)

[12] Miura,T. and Shio ya, I.: Incremen tal Update of Decision Trees for Temporal Objects, proc.*KRDB* (1999)

[13] Miura, T., Shioy a, I. and Mori, M.: Making Database Scheme More Accurate by Losing Information, proc. of *FoIKS* (2000)

[14] Quinlan, J.R: Induction of Decision Trees, *Machine Learning* 1-1 (1986)

[15] Quinlan, J.R.: C4.5 - Programs for Machine Learning, Morgan Kauffman (1993)

[16] Sakai, H. et al.: A Method for Behavior Modeling in Data Oriented Approach to Systems Design, *ICDE* (1984)

[17] Piatetsky-Shapiro, G. and Frawley, W.J. (ed.): Knowledge Discovery in Databases, *MIT Press* (1991)

[18] Wu, X.: Knowledge Acquisition from Databases, Ablex Publishing (1995)