Incremental Update of Decision Trees for Temporal Objects

Takao MIURA
Hosei University
Dept.of Elect. and Elect. Engr.
Kajino-cho 3-7-2, Koganei
Tokyo, JAPAN

Isamu SHIOYA
SANNO College
Dept.of Management and Informatics
Kamikasuya 1573, Isehara
Kanagawa, JAPAN

Abstract

In this investigation we discuss inductive classifiers to temporal information. Here we dicuss how to manage them in an incremental manner over a sequence of information. Our basic idea comes from calculating entropy in an incremental manner (i.e. for one step transition) to preserve the *precision* of classification.

1 Motivation

In theory of databases, we are expected to design all the information *in advance* into the form of *schemes* and then we can populate them. Note it is hard to change the schemes after construction of databases. This lack of flexibility of database framework causes severe problems to decision support, modelling and many other advanced applications. In fact, databases keep growing accpording to the changes of the world of interests, but there is no way in databases to follow the growth without re-building from scratch. One of the solutions comes from *knowledge acquisition* techniques. This approach provides us with suitable scheme discovery of databases which gives us a solution to scheme mismatch against growing databases.

A *decision tree* is a classifier of information to make implicit knowledge more explicit[Quinlan, 1986; 1993]. Here the generation of decision trees means how machines can learn a way to assign information to one of given classes. The tree is a set of conditions to classify input into classes. To classify input, we will specify a collection of classes, class membership criteria, attributes to be examined and training data. This technique is advantageous since we don't need no domain knowledge and we have the wider applicability. This also leads to efficient generation of the trees. In fact, no back tracking is needed thus it takes only polynomial time with respect to the number of attributes and inputs. The result is best optimized in a sense of entropy of class membership. have been proposed such as *C4.5*[Quinlan, 1993].

The technique seems useful to databases, since current contents of databases are consistent (no noise) and it can be seen as training set of data to the future ones. However, because no background knowledge is utilized during decision tree generation, it sometimes makes wide-spread trees to reason all the training data and wrong classification to the total data. More important is that there is few consideration to temporal aspects. Because *time* can be seen as an attribute over continuous values, no special semantics have been attached[Boronowsky, 1998]. This causes problems in real-time monitoring because decision trees have to be made from scratch. Successive changes of states cause vast amount of computation and spaces.

In this investigation, we extend a theory of decision trees over discrete (non-numeric) information by focusing incremental aspects of decision trees for time-series information because one step transition could be managed easily. We discuss how to obtain the trees with some allowance of *entropy*.

In section 2 we introduce temporal aspects of information, decision trees and the interaction between the two. Also we define *temporal decision trees* and discuss the feasibility. In section 3, we establish the foundation of incremental change of entropy and in section 4 we discuss how to manage them efficiently. Section 6 contains related works and we conclude our investigation in section 6.

2 Temporal Decision Trees

2.1 Objects and Decision Trees

Objects (or *entities*) and *classes* are primitive concepts where an object means a thing in the world of interests and a class its intention. It is assumed that every object belongs to a class and that an *attribute* is introduced to stress a role of a class. In this investigation we discuss *table data* E as our testing vehicles. Implicitly each row means an object and t_j describes a collection of the characteristic values over given set $\mathcal{A}$ of attributes $A_i, i = 1, .., n$, $\mathcal{C} = \{c_1, .., c_q\}(q \leq k)$ means which class an object belongs to. For simplicity we assume $\mathcal{A}$ and $\mathcal{C}$ are disjoint and there is no intersection between c_i and c_j objects. There can be duplicate rows but no duplicate object is allowed, and that no explicit *object* value appears.

EXAMPLE 1 As our running example, we assume 3 attributes and the class belonged to: *WindForce* (Windy, Breeze, NoWind), *Weather* (Fine, Cloudy, Rainy) and

Temperature (High, Medium, Low) are the characteristic values by which we will decide whether some horse races are held (*Yes*) or not (*No*) on *Condition*. We have a sequence E_1, E_2 and E_3 as below:

WF	$Weather$	$Temp$	$Condition$
$Windy$	$Fine$	$Medium$	No
$NoWind$	$Fine$	$High$	Yes
$NoWind$	$Fine$	Low	No
$Breeze$	$Cloudy$	Low	Yes
$Windy$	$Cloudy$	$High$	No
$NoWind$	$Cloudy$	$Medium$	Yes
$Windy$	$Rainy$	$High$	No
$NoWind$	$Rainy$	$Medium$	No

WF	$Weather$	$Temp$	$Condition$
$Breeze$	$Rainy$	$Medium$	Yes

WF	$Weather$	$Temp$	$Condition$
$NoWind$	$Cloudy$	$High$	Yes

□

A *classifier* is targeted not only for classifying a collection of objects but also for describing *how* to classify them and *what the objects mean*. In our story, a classifier consists of a set of conditions $\alpha_1, .., \alpha_k$ where α_i determines a class in such a way that rows in E satisfying α_i belong to the class, the set of rows is denoted by $E_i = \alpha_i(E)$. Each α_i is described as the conjunction of elementary form of $A_i =$ "a_i" and all the E is covered exclusively by $\alpha_1 \vee ... \vee \alpha_k$.

In a classifier called a *decision tree*, each node corresponds to a branching process, each branch to a condition $A_i =$ "a_i" where every path from the root to the leaf makes α_j. Branches are generated on an attribute according to the set of attribute values until all the rows belong to an identical class.

Induction of decision trees can be made by an algorithm $\mathtt{DT}(S, AT, AL)$ where S means a set of rows, AT a set of attributes and AL a set of conditions. A table E over is given as an input over a set of attributes $A_1, .., A_n$. We start the algorithm with $\mathtt{DT}(E, \{A_1, .., A_n\}, \phi)$. If it terminates successfully, we get the tree conditions and the corresponding classes of the paths $\alpha_1(d_1), ..., \alpha_k(d_k)$.
(1) If all the rows in S belong to an identical class d, we return $AL(d)$ successfully.
(2) Assume otherwise. Select an attribute $A \in AT$, and
(2.1) Make grouping of S on A into $S_1, .., S_w$. Each S_j is generated by the condition $A =$ "a_j" to S.
(2.2) For each S_j, do $\mathtt{DT}(S_j, AT - \{A\}, AL \cup \{A =$ "a_j"$\})$ in a recursive manner.
(2.3) Collect all the results as $AL_1(d_1) \vee .. \vee AL_w(d_w)$. If some S_j fails, we regard the result as distinct *undefined* class.
(3) If AT is empty, return fail.

We sometime replace the condition (1) by *most* of the rows belong to a class which means noises are allowed. Similarly, the step (3) is sometimes considered as *gray* zone and the classification might be inherited from the parent.

In step (2), to select attributes, the original approach has been proposed based on theory of *entropy*, i.e., probabilistic prediction. This is the point of decision trees by which we can obtain effective and much efficient results without inspecting the input many times. Intuitively an entropy of S (about class membership), $Ent(S)$, is an amount of uncertainty information; bigger entropy means less definite membership. To select an attribute, we calculate all the entropy gains by *instantiating* attributes and choose the one with the biggest.

Entropy can't be well-defined by means of probability density functions, since the value depends on how we describe these functions. Therefore we can discuss only discrete distribution generally.

EXAMPLE 2 In our running example, we begin the fact that the original entropy values for E_1, $E_1 \cup E_2$, $E_1 \cup E_2 \cup E_3$ (without *version*) are 1.860, 2.077 and 1.998 respectively. As a pruning ratio we assume 50 percent to small nodes of elements 3 or less; that is, we say *one class* if there are more than half objects of the class in such a node. For these information, we have three decision trees as below:

(WF)	
Windy	No
Breeze	Yes
NoWind	Yes

(WF)		
Windy		No
Breeze		Yes
NoWind	(Weather)	
	Fine	Yes
	Cloudy	Yes
	Rainy	No

(WF)	
Windy	No
Breeze	Yes
NoWind	Yes

□

2.2 Temporal Information

In *temporal* information there are 2 different categories of *valid time* and *transaction time*[Tansel *et al.*, 1993]. By the former users specify actual event-time, while by the latter users know when data has been registered into systems. Here we discuss transaction time. Such information $E_1, E_2, .., E_m$ may appear in a *real-time* mode. In fact, E_m appears just after $E_1, .., E_{m-1}$ and we may examine some information only from E_m by keeping $E_1, .., E_{m-1}$. That is why we stick to *incremental* nature. Each of $E_1, .., E_m$ does not contain all the details, and we call it *event* information. It is assumed here that we have differential (thus relatively small amount of) completed data with respect to base information. Although time concept is essentially described as continuous data, relevant information is recorded as discrete values such as *version number* and such implicit time concept is called *temporal*.

Another important point is to recognize a notion of *object identity*. If it doesn't matter, we get temporal information E_1, E_2 but we don't know *how* they have changed attribute values. The former is called *object-sensitive* and the latter *object-independent*. Since there is no explicit aspect of objects in decision trees, we discuss object-independent temporal event information in a real-time manner.

2.3 Temporal Decision Trees

Given a set $\mathcal{A}$ of attributes $A_i, i = 1, .., n$ and $\mathcal{C} = \{c_1, .., c_k\}$, we discuss *table data* E_t at a time t ($t = 1, .., m$) as our testing vehicles. Each row of E_t means an object at t and it describes a collection of the characteristic values over given set $\mathcal{A}, \mathcal{C}$. A class that an object

belongs to, as well as its characteristic values, can be different at a different time. We assume a sequence $\mathcal{E}_m$ of tables $E_1, ..., E_m$ that may describe *event* information. We note there can be duplicate rows and that no explicit *object* value appears.

Temporal decision trees (TDT) are classifiers of $\mathcal{E}_m$ to $\mathcal{C}$ that utilizes all the $\mathcal{E}_m$. In a naive approach of decision tree generation, whenever we get E_m at a time m, we would calculate the trees completely and immediately from scratch. Thus our true problem arises how we migrate the sequence of entropy for TDTs.

In the input $\mathcal{E}_m$ we assume there exist $n_{j,t}$ rows of the class c_j in E_t at time t where $j = 1, ..., k$ and $t = 1, .., m$ and we have a matrix $((q_{j,t}))$ of the frequency distribution ($q_{j,t} \geq 0$): Then we have a sets of uniformly distributed probabilities $p_{j,t} = q_{j,t}/n$ of the class membership where $q_{j,t} > 0$ means a number of occurrences and n means the all the sum of $q_{j,t}$. Let us define p_j as $p_{j,1} + ... + p_{j,m}$. The value $\log_2(1/p_j)$ is called *amount of temporal information* and an *temporal entropy* of the class membership, denoted by $Ent(\mathcal{E}_m)$, is defined as the expected amount of temporal information : $\Sigma_j p_j \log_2(1/p_j) = -\Sigma_j p_j \log_2 p_j$. Similarly we can define $Ent_A(\mathcal{E}_m)$. To generate branch on A, we define *gain* function $G(\mathcal{E}_m)$ as $Ent(\mathcal{E}_m) - Ent_A(\mathcal{E}_m)$. We will select the attribute A by which we obtain the maximum gain as the original one.

3 Incremental Change of Entropy

Given a TDT over a set of objects S, let us add a tuple $t = \langle a_1 ... a_n \rangle$ into a class c_1. In the input S of n rows, we assume there exist n_j rows of the class c_j, uniformly distributed probability $p_j = n_j/n$ of the class membership of c_j, and $n = n_1 + .. + n_q, n_i > 0$. The value $\log_2(1/p_j)$ is called *amount of information* of the class membership, and we define $Ent(S)$, called *entropy* of class membership of S, as the expected amount of information : $Ent(S) = \Sigma_j p_j \log_2(1/p_j) = -\Sigma_j p_j \log_2 p_j$.

Since $Ent(S) = -\Sigma_{i=1}^{k}(n_i/N)\log(n_i/N) = \log N - (1/N)\Sigma_{i=1}^{k} n_i \log n_i$ where n_i means the number of objects in S of a class c_i and $N = n_1 + .. + n_k$, $Ent(S \cup \{t\})$ becomes $\log(N+1)$ - $(1/(N+1))((n_1+1)\log(n_1+1) + \Sigma_{i>1}^{k} n_i \log n_i)$. Let $\delta(S, t^+)$ be the difference of the entropy $Ent(S \cup \{t\}) - Ent(S)$. Then we have

$\delta(S, t^+) = (1/(N+1))\{step(N)$ - $step(n_1)$ - $Ent(S)\}$

where $step(x) = (x+1)\log(x+1) - x \log x$.

In a case of deletion of a tuple $t = \langle a_1 ... a_n \rangle$ of a class c_1, $\delta(S, t^-) = Ent(S - \{t\}) - Ent(S)$ becomes $(-1/(N-1))\{step(N-1)$ - $step(n_1 - 1)$ - $Ent(S)\}$ where $N > 1$. In the case of $N = 1$, we have an empty set S and there is no value to be discussed.

The above discussion leads us to the case of $S_{A_1="a_1"}$ which consists of the elements in S whose A_1 values are a_1. In fact, $\delta(S_{A_1="a_1"}, t^+)$ and $\delta(S_{A_1="a_1"}, t^-)$ are $((1/(N_{a_1}+1))\{step(N_{a_1})$ - $step(n_1^{a_1})$ - $Ent(S_{A_1="a_1"})\}$ and $((1/(N_{a_1}-1))\{step(N_{a_1}-1)$ - $step(n_1^{a_1}-1)$ - $Ent(S_{A_1="a_1"})\}$ respectively where $n_1^{a_1}$ means the number of the elements of a class c_1 in $S_{A_1="a_1"}$ and $N_{a_1} = n_1^{a_1} + .. + n_k^{a_1}$. Note $Ent((S \cup \{t\})_{A_1="a"}) = Ent((S)_{A_1="a"})$ if $a \neq a_1$.

We define $Ent_A(S)$ as $\Sigma_{a \in A}(N_a/N)Ent(S_{A="a"})$ where N_a means the number of the elements of $S_{A="a"}$. When t is added to S and a_1 over A_1 has already appeared in other tuples, we have $Ent_{A_1}(S \cup \{t\}) = \Sigma_{a in A_1}(N'_a/(N+1))Ent((S \cup \{t\})_{A_1="a"})$ where N'_a means $N_a + 1$ if $a = a_1$ and N_a otherwise. Let $\Delta(S_{A_1}, t^+)$ be the difference of the entropy $Ent_{A_1}(S \cup \{t\})$ - $Ent_{A_1}(S)$. We have

$\Delta(S_{A_1}, t^+) = ((N_{a_1}+1)/(N+1)) \times \delta(S_{A_1="a_1"}, t^+) + (1/(N+1)) \times \{Ent(S_{A_1="a_1"}$ - $Ent_{A_1}(S)\}$

If a_1 over A_1 does not appear in other tuples, $S_{A_1="a_1"}$ is empty and we can show $Ent_{A_1}(S \cup \{t\}) = (N/(N+1)) \times Ent_{A_1}(S)$. That is, $\Delta(S_{A_1}, t^+) = (-1/(N+1)) \times Ent_{A_1}(S)$.

In a case of deletion, since $Ent_{A_1}(S - \{t\}) = \Sigma_{a in A_1}(N'_a/(N-1))Ent((S - \{t\})_{A_1="a"})$ where N'_a means $N_a - 1$ if $a = a_1$ and N_a otherwise, we have $\Delta(S_{A_1}, t^-) = Ent_{A_1}(S - \{t\}) - Ent_{A_1}(S)$ as $((N_{a_1}-1)/(N-1)) \times \delta(S_{A_1="a_1"}, t^-)$ - $(1/(N-1)) \times \{Ent(S_{A_1="a_1"}$ - $Ent_{A_1}(S)\}$

If a_1 over A_1 does not appear in any other tuples, then $S_{A_1="a_1"}$ is a singleton and we can show $Ent_{A_1}(S - \{t\}) = (N/(N-1)) \times Ent_{A_1}(S)$. That is, $\Delta(S_{A_1}, t^-) = (1/(N-1)) \times Ent_{A_1}(S)$.

4 Updating Temporal Decision Trees

4.1 Data Structure and Operators

Given $\mathcal{E}_m = E_1, ..., E_m$, to describe TDT, we introduce a *multi-dimensional associative array* where $A_1, .., A_n, N_1, .., N_k$ each element of $\langle a_1, .., a_n, d_1, .., d_k \rangle$, describes a number of rows of the values $a_1, .., a_n$ belonging to a class c_j in E is d_j ($j = 1, .., k$). Let E be $\cup \mathcal{E}_m$ transformed into this array form. Here we assume every domain of A_j is finite and such array can be implemented easily by hash structure with almost all the operations take time of $O(1)$.

To obtain entropy $Ent(E)$ or $Ent_A(E)$, there must be some *operators*. We assume *selection* operator σ_Γ to obtain all the rows satisfying conditions Γ: $\sigma_\Gamma(E) = \{t \in E, \Gamma(t) = True\}$. Γ is a null (means *true*) or the conjunction of the form $A_i =$ "a_i". For each tuple $t \in E$, $t[A_i]$ means the vale of t over A_i. And $cnt\langle a_1, .., a_n \rangle$ is a tuple $\langle d_1, .., d_k \rangle$. A function $\gamma(E)$ returns $\langle d'_1, .., d'_k \rangle$ where d'_j means all the sum of N_j values in $cnt(E)$. Let $\Sigma\langle n_1, .., n_k \rangle$ be $n_1 + .. + n_k \rangle$. Then $\rho(E)$ returns the entropy value by $-\Sigma_j(q_j/N)\log(q_j/N)$ where $q_j > 0$ is the sum of N_j values in $cnt(E)$ and N the sum of q_j. Thus $Ent(E)$ is obtained by $\rho(E)$, $Ent_{A="a"}(E)$ by $\rho(\sigma_{A="a"}(E))$ and $Ent_A(E)$ by $\Sigma_{a \in A}(\gamma(E)[N_A]/\Sigma\gamma(E)) \times Ent_{A="a"}(E)$.

We can restate many expression by means of these operators, like $Ent(S_{A="a"})$ is considered as $Ent(\sigma_{A="a"}(E))$. Another examples is, when generating branches on a node of $A_1 =$ "a_1" $\wedge$ $A_2 =$ "a_2" by $A_3 =$ "a_3", we obtain the new entropy value on the branch by $Ent_{A_3="a_3"}(\sigma_{A_1="a_1" \wedge A_2="a_2"}(E))$.

4.2 Maintaining Entropy

Given $\mathcal{E}_{m-1}$ and $E = \cup\mathcal{E}_{m-1}$ transformed into an multidimensional associative array, we assume $E_m = \{t\}$ to be added where $t = \langle b_1 ... b_n \rangle$.

We maintain our TDT by modifying special data structures E and $[E]$ as described below. That is, we make up E and $[E]$ that describes a TDT where each node corresponds to a set of tuples S of interests, each child node corresponds to one of grouping of S over some attribute A as $S_{A="a"}$.

To obtain the incremental change at a node[1] of the TDT which corresponds to $S = E_{A_1="a_1",..,A_g="a_g"}, g \geq 0$, we need 3 kinds of information: the entropy $Ent(S)$ of class membership of S, the numbers of objects $n_1, .., n_k$ of classes $c_1, .., c_k$ in S, and the entropy $Ent_{A_j}(S)$ of class membership of S over $A_j, j = g+1, .., m$ (A_j means not-processed-yet attribute). To obtain $Ent_{A_j}(S)$, we must have all of $Ent(S_{A_j="a_j"})$ where $a_j \in S[A_j]$. Note we have to keep $Ent_{A_j}(S)$ other than the maximum gain. For a leaf node, we need the first two information.

For incremental update, we introduce an auxiliary array $[E]$ over $A_1, .., A_n, Ent_{A_1}, .., Ent_{A_n}, Ent, N^k$ where each tuple $\langle x_1, .., x_n; V_1, .., V_n; V; n_1, .., n_k \rangle$ means, for $F = E_{A_1="x_1" \wedge .. \wedge A_n="x_n"}$, $Ent_{A_1}(F) = V_1$, .., $Ent_{A_n}(F) = V_n$, $Ent(F) = V$, and n_i means the number of elements of class c_i in E. Some of x_j, V_i may be $*$ that means *don't care*. For a TDT of E, we keep the following information in $[E]$:

(1) For a root, there are 3 kinds of rows in $[E]$:

(a) a root: $\langle *, .., *;\ Ent_{A_1}(E), .., Ent_{A_n}(E);\ Ent(E);\ n_1, .., n_k \rangle$

(b) direct children of the TDT where $A_1 = "a_1"$, $a_1 \in E[A_1]$: $\langle a_1, .., *;\ *, Ent_{A_2}(E_{A_1="a_1"}), .., Ent_{A_n}(E_{A_1="a_1"});\ Ent(E_{A_1="a_1"});\ n_1^{a_1}, .., n_k^{a_1} \rangle$

(c) reference rows : $\langle *, .., *, a_i, *.., *;\ *, .., *;\ Ent(E_{A_i="a_i"});\ n_1^{a_i}, .., n_k^{a_i} \rangle$ where $A_i = "a_i"$, $a_i \in E[A_i]$ $(i = 2, .., n)$

(2) For other non-leaf nodes of $S = E_{A_1="a_1",..,A_g="a_g"}$ $(g > 0)$, we have 3 kinds of rows in $[E]$:

(a) the node: $\langle a_1, .a_g, *, .., *;\ *, .., *,\ Ent_{A_{g+1}}(S), .., Ent_{A_n}(S);\ Ent(S);\ n_1^{a_1..a_g}, .., n_k^{a_1..a_g} \rangle$

(b)direct children where $A_{g+1} = "a_{g+1}"$, $a_{g+1} \in E[A_{g+1}]$: $\langle a_1, ..a_g, a_{g+1},\ *, .., *;\ *, .., *,\ Ent_{A_{g+2}}(S_{A_{g+1}="a_{g+1}"}),\ .., Ent_{A_n}(S_{A_{g+1}="a_{g+1}"});\ Ent(S_{A_{g+1}="a_{g+1}"});\ n_1^{a_1..a_{g+1}},\ .., n_k^{a_1..a_{g+1}} \rangle$

(c) reference rows with $A_i = "a_i"$, $a_i \in E[A_i]$ $(i = g+2, .., n)$: $\langle a_1, .., a_g, *, .., *, a_i, *.., *;\ *, .., *;\ Ent(S_{A_i="a_i"});\ n_1^{a_1..a_gb}, .., n_k^{a_1..a_gb} \rangle$

(3) For other leaf nodes of $S = E_{A_1="a_1",..,A_g="a_g"}, g > 0$, there is row in $[E]$: $\langle a_1, .a_g, *, .., *;\ *, .., *;\ Ent(S);\ n_1^{a_1..a_g}, .., n_k^{a_1..a_g} \rangle$

4.3 Incremental Update of Decision Tree

Given $\mathcal{E}_{m-1}$, we apply an algorithm IUDT(α, AT, t^+) for $E_m = \{t\}$ to be added where $t = \langle a_1 ... a_n \rangle$, AT means a set of attributes and a node α in the tree. Initially we create data structure $[E]$ for the decision tree to $E = E_1$ by looking at the algorithm DT($E, \{A_1, .., A_n\}, \phi$).

We start the algorithm with the root node: IUDT(the root, $\{A_1, .., A_n\}$, t^+). We start with the root with $E = \cup\mathcal{E}_{m-1}$ and examine (1) nodes above. In the current classifier TDT, we have selected A_1 as the attribute of the best entropy, thus we re-calculate the entropy of class membership of E over A_i, $Ent_{A_i}(E)$, $i = 1, .., n$. This can be made easily by $\Delta(E_{A_i}, t^+)$ since, in $[E]$, there are the current $Ent_{A_i}(E)$ and all the information of the node of $E_{A_i="b_i"}$, N_{b_i}, N, $\delta(E_{A_i="b_i"}, t^+)$, $Ent(E_{A_i="b_i"})$, $Ent_{A_i}(E)$, and $n_i^{b_i}$.

Comparing $Ent_{A_1}(E \cup \{t\})$ to all other $Ent_{A_j}(E)$, we can examine if A_1 is still selected as the best one or not. If not, we have to select the new attribute A_j, we delete $Ent_{A_1}(E)$ related rows and subtrees, and create the subtree (from the root) completely from E by means of $\sigma, \gamma, \Sigma, \rho$ operators. If *yes*, we examine whether this node should becomes a leaf node and we remove all the subtrees and make the node a leaf. Otherwise we go to the node of $E_{A_1="a_1"}$.

In the case of intermediate nodes ((2) case above), we change $Ent(E_{A_1="a_1",...,A_{g-1}="a_{g-1}"})$ by adding the incremental change of the entropy. In the current classifier, we have selected A_g as the attribute of the best entropy, thus we re-calculate the entropy of class membership of $S = E_{A_1="a_1",...,A_{g-1}="a_{g-1}"}$ over A_i, $Ent_{A_i}(E_{A_1="a_1",...,A_{g-1}="a_{g-1}"})$ for $i = g+1, .., n$. If A_g is selected again, the story goes in a same way as the root. Note that, when a_g is a new comer, there is no node $Ent(E_{A_1="a_1",...,A_g="a_g"})$ thus we create a leaf node and the related information.

In the case of leaf nodes, the number of the objects of a class c_1 increases. It may cause new nodes. If the case, we create a new (intermediate) node and select A_g as the attribute of the best entropy as usual, until we obtain the decision subtree.

We summarize our IUDT algorithm.

(0) Let S be the corresponding set of tuples to α.

(1) If α is a leaf node, we add t to α by adjusting $Ent(S)$ and increasing n_1.

(1.1) If all the objects in α belong to an identical class, we return successfully.

(1.2) Assume otherwise. Now α is changed to the intermediate node. Calculate $Ent_A(S)$ for each S and select an attribute $A \in AT$ of the minimum value.

(1.3) Make children nodes $\beta_1, .., \beta_w$ of α on A.

(1.4) Each S_j of β_j is created by the condition $A = "a_j"$ to S for every $a_j \in A$.

(1.5) For each β_j, do IUDT($\beta_j, AT - \{A\}, t^+$) in a recursive manner.

(1.6) Return to the caller.

(2) If α is an intermediate node, we adjusting $Ent(S)$, $Ent_A(S)$ of the minimum attribute A and increase n_1.

(2.1) If $Ent_A(S)$ is not the minimum, then we remove the current information and build a new subtree rooted at α completely as in (1.2)-(1.5) and return.

(2.2) Assume otherwise. Select the child node β of $A = "a"$

[1] Remember every node has been generated along a sequence of the conditions $A_1 = "a_1" \wedge ... \wedge A_g = "a_g"$ for some attribute sequences.

where $t[A] = a$, and apply IUDT$(\beta, AT - \{A\}, t^+)$ in a recursive manner. Return to the caller.
(2.3) If there is no such node, we make a new node β for t^+ and the relevant information.
(2.4) Return to the caller.

Similarly we can discuss a deletion IUDT(α, AT, t^-).

Clearly our algorithm takes $O(h)$ where h means the height of a TDT, because it goes into the leafs of trees but not others. We note that space complexity of $[E]$ is $O(\log |E|)$. In fact, let d be the average size of $E[A_i], i = 1..,n$. For each $Ent_{A_i}(E)$ we create $d \times n$ rows as direct children *and one* subtree for each child of a TDT. The subtree may have $|E|/d$ elements in average, therefore, in total we have $(c \times n \times d/\log d) \times \log |E|$ rows.

It is straightforward to extend this algorithm to a set of tuples and to a case that allows *missing values*.

EXAMPLE 3 In our example, for E_1 we have the root with entropy 1.860 of the class membership. There is a row in $[E]$ $\langle *, *, *;\ 1.406, 1.561, 1.561;\ 1.860;\ 3, 5\rangle$ which means the entropy over $WF, Weather, Temp$ are 1.406, 1.561 and 1.561 respectively. There are 3 elements of Yes class and 5 of No class.

Next we add E_2. The root entropy is adjusted incrementally to 2.077 and the row in $[E]$ is changed to $\langle *, *, *;\ 1.530, 1.585, 1.531;\ 2.077;\ 4, 5\rangle$. But there is no other change. Finally we add E_3. The root entropy is changed to 1.998, the row in $[E]$ is changed to $\langle *, *, *; 1.485, 1.571, 1.522; 2.077; 5, 5\rangle$. But $WF =$ "$NoWind$" node is not a leaf (3 Yes and 2 No) any more, and we modify this node as an intermediate. The row in $[E]$ is changed to $\langle$"$NoWind$", $*, *; *, 1.522, 1.522; 2.093; 3, 2\rangle$. Three new leaf nodes are created for $Weather$:

$\langle$"$NoWind$","$Fine$", $*; *, *, *; 1.0; 1, 1\rangle$
$\langle$"$NoWind$","$Cloudy$", $*; *, *, *; 0.0; 2, 0\rangle$
$\langle$"$NoWind$","$Rainy$", $*; *, *, *; 0.0; 0, 1\rangle$
□

5 Related Works

From the viewpoint of machine learning, a theory of logic-based inductive inference has the strong relation with KDD[Fayyad *et al.*, 1995; Shapiro *et al.*, 1991]. However utilized is only the scheme information but not characteristics of instances nor domain-specific knowledge except a few[Kietz *et al.*, 1994]. There is few research of temporal objects from this point of view to our knowledge. It is not easy to discover implicit conditions to give property values. Traditionally *time series analysis* is well-known and applied to economics. But there is no research for the interaction between time series analysis and non-numerical domains. Also, as we said, the results are hard to interpret and to understand for non-experts. Some technique in machine learning has been successfully applied in many fields like medical engineering and process control[Boronowsky, 1998].

Incremental update of decision tree has been discussed[Utgoff, 1989]. The result says we can think this issue as *change of trees*. Here all the objects have been attached to nodes in the trees. When we know the change of the *trees* we could re-construct then by using the information attached. No incremental calculation has been discussed here thus it is impossible to deal with a large amount of information.

Knowledge discovery in databases (KDD) is one of the hot research topics (see [Mannila, 1997]). It is nothing but knowledge acquisition activity but differs from using database as background (domain) knowledge. Our works show how to obtain new database schemes that are suitable for current database instances (see [Miura *et al.*, 1996; 1998]).

6 Conclusion

In this work, we have proposed a new classification methodology for temporal objects. Our new decision tree algorithm utilizes temporal aspects of the objects which are considered as a sequence of information. We keep attribute values in a condensed manner to obtain entropy efficiently. The algorithm takes only $O(height)$ time to examine whether the current TDT is valid and the additional space complexity is $O(\log |E|)$.

References

[Boronowsky, 1998] Boronowsky, M. Automatic Measurement Interpretation of a Physical System with Decision Tree Induction *Conference of IDEAL* (1998).

[Fayyad *et al.*, 1995] Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. and Uthurusamy, R. (Eds). Advances in Knowledge Discovery and Data Mining. *MIT Press* (1996)

[Kietz *et al.*, 1994] Kietz,J.U. and Morik, K. A Polynomial Approach to the Constructive Induction of Structural Knowledge. *Machine Learning* 14 (1994).

[Mannila, 1997] Mannila, H. Methods and Problems in Data Mining. *Intn'l Conf. on Database Theory* (ICDT) (1997).

[Miura *et al.*, 1996] Miura, T. and Shioya, I. Knowledge Acquisition for Classification Systems, proc. *ICTAI* (1996).

[Miura *et al.*, 1998] Miura, T. and Shioya, I. Learning Concepts from Databases. *Conference and Workshop of DEXA* (1998)

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

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

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

[Tansel *et al.*, 1993] Tansel, A.U. et al. Temporal Databases. *The Benjamin Cumming Publ.* (1993).

[Utgoff, 1989] Utgoff, P.E. Incremental Induction of Decision Trees. Machine Learning 4-2 (1989).