# Inductive Classification Using Taxonomy

Isamu SHIOYA
SANNO College, Kamikasuya 1563, Isehara, Kanagawa, Japan
Email: shioya@sanno.ac.jp

Takao MIURA
Dept.of Elect. and Elect. Engineering,
Hosei University, Kajinocho 3-7-2, Koganei, Tokyo, Japan
Email: miurat@k.hosei.ac.jp

**Abstract**

We discuss how to classify input inductively where we assume taxonomy on attributes. Although we learn classification rules inductively by means of decision tree generation, we can't utilize the generation and testing methodologies for this purpose since each input data carries multiple roles based on taxonomy. In this investigation we introduce Decision Tree with Hierarchy (DTH) to class and attributes; to each attribute we assume taxonomy on the domain in addition to class hierarchy. Then, we extend information-theoretic definition that causes non-exclusive decision trees to test inputs, and we discuss a new algorithm to examine which paths should be examined.

*Content Areas* – Machine Learning, AI in Databases, AI in Data Mining

## 1 Motivation

*Knowledge acquisition* from databases is a technique by which we can obtain useful information that we didn't have found before. If the intentional information of databases (i.e., schemes) are utilized, the process is called *classification* because the intents of the inputs are defined in advance. In this case we can assume all the information are conformed to the scheme.

A *decision tree* (DT) is one of the classification techniques[10, 11]. To classify input, we will specify a collection of classes, attributes to be examined and training data. We output class membership criteria inductively in a form of trees. The tree is a set of conditions to classify input into classes. The technique has many advantages. First we don't need no domain knowledge but probabilistic prediction, thus we have the wider applicability. This also leads to efficient generation of the trees as well as efficient examination of production data. In fact, no back tracking is needed here and it takes polynomial time with respect to the number of attributes and inputs. The result is best optimized in a sense of entropy of class membership. It is worth noting that the issue to obtain the *smallest* decision trees is NP-hard.

On the other hand, no background knowledge such as concept hierarchy (class hierarchy and taxonomy) is utilized and sometimes it may cause *overfitting trees*[2]. That is, we would generate *wide-spread* trees to reason all the training data but we might make wrong classification to the total data, because such classification process would become slow, incorrect, less useful, less precise and hard to maintain. To avoid such a tree, a notion of *noises* is introduced by which we would guess erroneous input in addition to property of *prediction*. More serious is a fact that we can't examine any input over taxonomy because decision trees have no knowledge about it.

Decision trees with taxonomies play new roles on classification. When allowing taxonomies and class hierarchy to decision trees, we can examine classification by decision trees where elementary condition $A_i = "a_i"$ appears in non-exclusive way. For example, a tuple $< chorus, piano >$ satisfies two conditions $\{PLAYER = "chorus", ACCOMPANIMENT = "keyboard"\} : TAPE$ and $\{PLAYER = "vocal", ACCOMPANIMENT = "piano"\} : CD$ where $TAPE, CD$ represent classification at some music shop. Also we assume two taxonomies: *piano* `ISA` *keyboard* and *chorus* `ISA` *vocal*. Remember such situation can't happen in conventional decision trees. This ambiguity can be resolved by new decision trees. Informally, the trees describe elementary conditions $A_i = "a_i"$ and their test sequences (paths). When examining a tuple, we decide the least general one along the paths.

**EXAMPLE 1** In the above case, we assume the decision tree in figure 1. The tuple $< chorus, piano >$ satisfies two conditions $\{PLAYER = "chorus", ACCOMPANIMENT = "keyboard"\} : TAPE$ and $\{PLAYER = "vocal", ACCOMPANIMENT = "piano"\} : CD$, but becomes $TAPE$ since *chorus* is *closer* to the first condition.

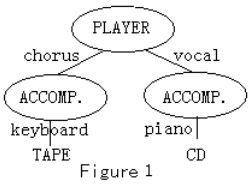

Figure 1

An above example says we have to discuss how to examine the trees as well as generation of the trees. In this work, we introduce *decision trees with hierarchy* (DTH) to utilize class hierarchy and taxonomy over attribute values. DTH is different from DT since DTH *interpolates* input information at two points. The first is that we build up DTHs by using taxonomy. To do that, we need new framework to capture the new environment. The second point is that we examine any inputs which don't appear directly in the trees but can be deduced by using taxonomy.

In this work, section 2 contains definitions of decision trees. In section 3, we formalize a notion of DTH and discuss how we generate the trees. In section 4 we show how to examine inputs to DTHs. Section 5 contains some experimental results. After discussing related works in section 6, we conclude our work in section 7.

# 2 Definitions

## 2.1 Data Descriptions

Here we assume *objects* (or *entities*) and *classes* as primitive concepts where an object means a thing in the world of interests and a class its intention. Every object belongs to some classes. Also we assume a set of *attributes* to characterize objects. Given a class $c$, we denote a collection of objects of a class $c$ by $\Gamma(c)$. In this investigation we discuss *table data* $T$ as our testing vehicles :

$$T = \left(\begin{array}{ccc} A_1 & ... & A_n{:}\mathcal{C} \\ \hline a_1^1 & ... & a_n^1{:}c_1 \\ .. & ... & ..:.. \\ a_1^k & ... & a_n^k{:}c_k \end{array}\right) = \left(\begin{array}{c} \mathcal{A}:\mathcal{C} \\ \hline t_1 : c_1 \\ .. \\ t_k : c_k \end{array}\right)$$

Each row (tuple) means an object and $t_j$ is described by a collection of the characteristic values over a 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[1]. Since every object correspond to one tuple, duplicate tuples are allowed but duplicate objects are not allowed.

We say $c_1$ `ISA` $c_2$ for two classes $c_1, c_2$, if we have a constraint $\Gamma(c_1) \subseteq \Gamma(c_2)$, and a *class hierarchy* is a set of `ISA` where every class has just one *parent* with one exception `@E` that means an *object class* as a very top. Here we assume every class appears in the hierarchy. The hierarchy is called *single* if $\Gamma(c) \cap \Gamma(c') \neq \phi$ means $c$ `ISA` $c'$ or $c'$ `ISA` $c$, i.e., $\Gamma(c) \subseteq \Gamma(c')$ or $\Gamma(c') \subseteq \Gamma(c)$. In the following, we assume a single class hierarchy.

We assume a notion of *taxonomy* over every attribute. A taxonomy over an attribute $A$ shows how values are organized into hierarchy, where ancestor values mean *more abstract*. Given two attribute values $a_1, a_2$ in $A$, we say $a_1$ `ISA` $a_2$ if $a_2$ is an ancestor of $a_1$ (i.e., $a_2$ is more abstract than $a_1$). We assume a *single* hierarchy too and the very top `@`. Conceptually all children *cover* their parent; a parent value $a$ could be described as one of its children $a_1, .., a_m$. A table $T$ is called *consistent* if all the attribute values appear in the taxonomies. In this work, we assume all the tables are consistent. Note a tuple which contains *intermediate* values $v$ on taxonomy describes that we know $v$ but we can't describe it in more specific concepts. Thus we can say the object is described as the tuple in the least general way. Trivially we can replace $v$ value by upper one but still keep consistency.

**EXAMPLE 2** Let us explain our running example. We assume 2 attributes *growth*(`fat, stout` and `lean`), *food* (`vegetable, meat`) and classes *healthy, weak* and *sickly*.

There are two training sets of data $T_1$ and $T_2$ respectively:

| *growth* | *food* | *health* |
|---|---|---|
| *stout* | *meat* | *weak* |
| *fat* | *meat* | *sickly* |
| *lean* | *vegetable* | *healthy* |

| *growth* | *food* | *health* |
|---|---|---|
| *stout* | *meat* | *weak* |
| *fat* | *meat* | *sickly* |
| *lean* | *vegetable* | *healthy* |
| *lean* | *meat* | *weak* |

[1] We assume $\mathcal{A}$ and $\mathcal{C}$ are disjoint.

There exist taxonomies on *growth* and *food* and a hierarchy on classes. See figure 2, 3 and 4.

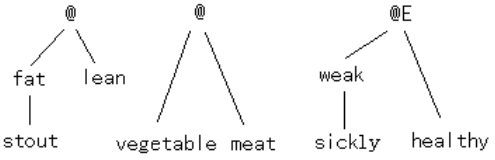


Figure2 Figure3 Figure4

## 2.2 Decision Trees

*Classification* has been targeted not only for classifying objects correctly into one of the classes but also for developing *how* to do that, because this could make clear *what the objects mean.* In our story, the classification consists of a set of conditions $\alpha_1, .., \alpha_k$ where each $\alpha_i$ determines a class in such a way that all the rows in $T$ satisfying $\alpha_i$ belong to the class, the set of rows is denoted by $T_i = \alpha_i(T)$. Each $\alpha_i$ is described as the conjunction of elementary form of $A_i =$ "$a_i$" and all the $T$ is covered exclusively by $\alpha_1 \vee ... \vee \alpha_k$.

A special attention has been paid on the tree form of the classification, called *decision trees* by which we can obtain compact representation and efficient navigation of the rules. Each node corresponds to a branching process, each branch to a condition $A_i =$ "$a_i$" where $a_i$ is an attribute value on $A_i$ and every path from the root to the leaf makes $\alpha_j$. Branches are generated on an attribute $A$ according to the set of attribute values over $A$ until all the rows satisfying these conditions belong to an identical class. In fact, tree representation means we consider $\alpha_i$ as a list of elementary forms.

Induction of decision trees can be made by the following 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 $T$ is given as an input over a set of attributes $A_1, .., A_n$, but in this case, all the classes $c_1, .., c_q$ in $T$ are assumed disjoint, i.e., there is no $c, c'$ such that $\Gamma(c) \cap \Gamma(c') \neq \phi$. We start the algorithm with $\mathtt{DT}(T, \{A_1, .., A_n\}, \phi)$. 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 where $d$ means the class in $S$.
(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.

Conventional story says that, in the step (1), '*all* the rows' could be replaced by '*most* of the rows', in other words, noises are allowed though we don't discuss any more. The step (2.3) also might be considered as *gray* zone and the classification might be inherited from the parent.

However which attributes should be selected at the step (2) above ? To select attributes, this approach utilizes 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.

The scenario of the selection is as below : 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 $E(S)$, called *entropy*, as the expected amount of information :

$$E(S) = \Sigma_j p_j \log_2(1/p_j) = -\Sigma_j p_j \log_2 p_j$$

. Let $a_1, .., a_w$ be all the attribute values appeared on $A$ in $S$, and $S_1, .., S_w$ be all the groups of $S$ according to the conditions $A =$ "$a_j$", $j = 1, .., w$. Then we define $n_i^j$ as the number of the rows in $S_j$ that belong to a class $c_i$. Note $n_i = \Sigma_{j=1}^{w} n_i^j$ and $m_j = \Sigma_{i=1}^{q} n_i^j$ is the number of rows of $S_j$. The entropy $V_j$ of the class membership of $S_j$ is, by the definition :

$$-\Sigma_{i=1}^{q}(n_i^j/m_j)\log_2(n_i^j/m_j)$$

and the entropy $E_A(S)$ of the class membership of $S$ over $A$ is defined as :

$$\Sigma_{j=1}^{w}(m_j/n) \times V_j$$

By the above consideration, the *gain* $G_A(S)$ to do branching on $A$ is defined as $E(S) - E_A(S)$. We will select the attribute $A$ by which we obtain the maximum gain.

**EXAMPLE 3** Let us generate decision trees to $T_1$ and $T_2$. See figure 5 and 6.

In $T_1$ where $\alpha = 0.1$, the entropy is 1.58496 and we get the best gain at *growth* where the relative entropy is 0.0. When *growth="fat"* we have a class *sickly*, when *growth="lean"*, we have a class *healthy*. and when *growth="stout"*, we have a class *weak*.

In $T_2$, the entropy is 1.50 and we get the best gain at *growth* where the relative entropy is 0.5. When When *growth="fat"* we have a class *sickly*, when *growth="lean"* and *food="meat"*, we have a class *weak*, when *growth="lean"* and *food="vegetable"*, we have a class *healthy* and when *growth="stout"*, we have a class *weak*.

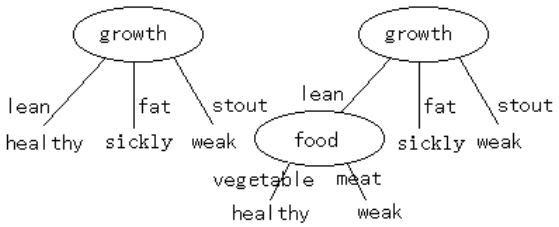

Figure5



Figure 6

# 3 Decision Trees with Hierarchy(DTH)

In the original theory, as we pointed out, the algorithm ignores domain-specific knowledge such as taxonomies and class hierarchy to reason all the input. This is one of the reasons why we generate *wide-spread* trees to reason all the training data. DTH differs from the ones proposed before since we don't have suitable theory of entropy any more. Remember the classes $c_1, .., c_q$ have been assumed to be disjoint in the original theory and the entropy $-\Sigma p_i \log_2 p_i$ describes the expected amount of information with respect to *exclusive* class membership. Let us revisit the definition under the new environment.

In a table $T = \begin{pmatrix} t_1 : c_1 \\ .. \\ t_k : c_k \end{pmatrix}$ of $k$ tuples, it seems reasonable for us to consider $t_j : c_j$ as an object carrying the values $t_j$ of the *lowest* (or least general) class $c_j$ that the object has. In this case the object is in all the ancestor classes. This is true even if there are duplicate tuples. Similarly $t_j$ holds *lowest* value on each attribute taxonomy.

Let us state our situation formally. Assume $e$ of a class $c_1$ and $c_1$ `ISA` $c_2$. Then $e$ should be of a class $c_2$. Given $p_1 = n_1/n_2$ and $p_2 = n_2/k$ where $n_1, n_2$ are the cardinalities of $\Gamma(c_1)$,$\Gamma(c_2)$ respectively, and $k$ means the number of rows, the probability of the membership is obtained by $p_1 \times p_2 = n_1/k$, thus we consider the probability of $c_1$ membership of $e$ as the one without looking at the hierarchy. We have $\Sigma_{i=1}^{q} n_i \geq k$ because a class has objects of the class and the ones of all the descendants. We change the definition of entropy though we define it as the expected amount of information:

$$E(T) = \Sigma_j p_j \log_2(1/p_j) = -\Sigma_j p_j \log_2 p_j$$

. Note summation is made according only to the classes appeared in the table same as previously defined, but the (new) class membership can be obtained easily by looking at `ISA` hierarchy and the input table $T$. This is because we look at only least general tuples. But, for two distinct tuples $t_1 : c_1$ and $t_2 : c_2$ where $c_1$ `ISA` $c_2$, $t_1$ should be considered as an element of a class $c_2$.

As for relative entropy, first we divide the table into (non-disjoint) subsets. For instance, we have $a_1, .., a_w$ on an attribute $A$ in $T$, then we make a set $T|a_j$ of tuples in $T$ whose $A$ value `ISA` $a_j$. Then we obtain the entropy $Ent(T|a_j)$. The relative entropy is defined as expected entropy among $T|a_1, .., T|a_w$.

$$Ent_A(T) = \Sigma_{j=1}^{w} (m_j/k) Ent(T|a_j)$$

**EXAMPLE 4** Let us continue our running example of *Health* decision. According to $T_1$ and taxonomies, we apply our algorithm to these data. The entropy $E(T_1)$ is 1.44662, the branch attribute is *growth* again with the relative entropy 0.25. Cmpared to the conventional tree, we have some difference on the branch on `fat`, the result class is now *weak* (see figure 7).

On the other hand, in the case of $T_2$, the entropy is 1.31128 and we get a simple tree on *food* with the relative entropy 0.396241. See figure 8.

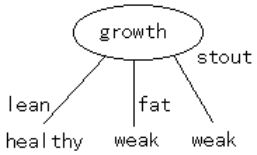

Figure 7

Figure 8

Our DTH generation algorithm `DTH`$(S, AT, AL)$ is now described:

(1) If the entropy $E(S)$ is small ($E(S) \leq \alpha$), no useful information can be obtained here for class membership, and we return $AL(d)$ successfully where $d$ means the largest class in $S$.
(2) Assume otherwise. Select an attribute $A \in AT$ of the best gain, and
(2.1) Make grouping of $S$ on $A$ into $S_1, .., S_w$ non-exclusively. Each $S_j$ is generated by the condition $A =$ "$a_j$" to $S$.
(2.2) For each $S_j$, do `DTH`$(S_j, AT - \{A\}, AL \cup \{A =$ "$a_j$"$\})$ 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.

Such approach has several advantages. First, we can keep all the advantages of decision tree generation. (That is, we have efficient generation/evaluation of trees, the process goes in a bottom-up manner, noises are allowed. But no domain knowledge is required but entropy is utilized.). Also we introduced entropy over taxonomy into DTH generation.

# 4 Testing Decision Trees with Hierarchy

When allowing taxonomies and class hierarchy to decision trees, we can examine classification in more powerful way. DTH is nothing but a decision tree where elementary condition $A_i =$ "$a_i$" appears in non-exclusive way. As we pointed out in the first section, a tuple $< a_1, b_1 >$ satisfies two paths $\{A_1 =$ "$a_1$", $A_2 =$ "$b_2$"$\}$ and $\{A_1 =$ "$a_2$", $A_2 =$ "$b_1$"$\}$ where $a_1$ `ISA` $a_2$ and $b_1$ `ISA` $b_2$. Such ambiguity can be resolved by introducing *semantic metric* among conditions. Informally, when selecting

one of possible paths, we calculate the number of arcs between the two values considered as metric, then we compare the values in lexicographical order of attributes that is given by the decision tree.

**EXAMPLE 5** In the above case, we assume the decision tree in figure 9. The tuple $< a_1, b_1 >$ satisfies two conditions $\{A_1 = "a_1", A_2 = "b_2"\} : c_1$ and $\{A_1 = "a_2", A_2 = "b_1"\} : c_2$, but becomes $c_1$ since the former metric is $< 0, 1 >$ while the latter $< 1, 0 >$. The path has the smaller metric 0.

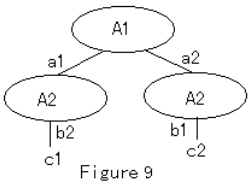

Figure 9

Such process is called DTH-test and described as `DTHtest( DTH, t)` that means how to examine input data (tuple `t`) by using `DTH`.

(1) take a root of `DTH` (an attribute) to be examined.
(2) select a branch that has the minimum metric.
(3) then go down along the path to the child node.
(4) if the node is intermediate, go back to (2).
(5) when we get to the leaf, the node tells us class name ; when we see no path to be selected, we conclude *undefined* (no definite class).

Important is that we can examine data by inferring *least general* ancestors when testing DTH, even if the ancestor doesn't appear in $E$. This means some sort of *interpolation* of attribute values/classes works well.

On the other hand, sometimes we can't examine data by inferring *least general* ancestors when testing DTH for values *not* appeared in E. This can happen when we need deduction of attribute values/classes in a upper direction.

**EXAMPLE 6** Assume we have another kind of taxonomies (see fig 10) and two other training data $T_3$ and $T_4$.

| *growth* | *food* | *health* |
|---|---|---|
| *stout* | *meat* | *weak* |
| *normal* | *vegetable* | *weak* |
| *lean* | *meat* | *healthy* |

| *growth* | *food* | *health* |
|---|---|---|
| *stout* | *meat* | *weak* |
| *fleshy* | *vegetable* | *weak* |
| *lean* | *meat* | *healthy* |

Then we have the DTH for $T_3$ with the entropy is 0.918296, relative entropy on *growth* is 0.550978. Also we have another simple DTH for $T_4$ with the entropy 0.918296 (see fig 11).

By using `DTHtest`, a tuple $< fat, vegetable >$ can be tested to $T_3$. In this DTH, we get the result class *weak*. However in the case of $T_4$, we can't because `fat` doesn't appear in $T_4$ and thus in the DTH.

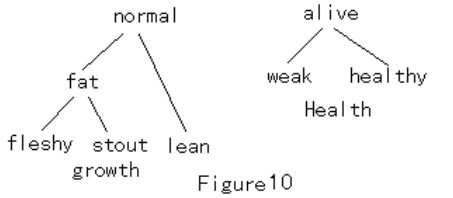

Figure10

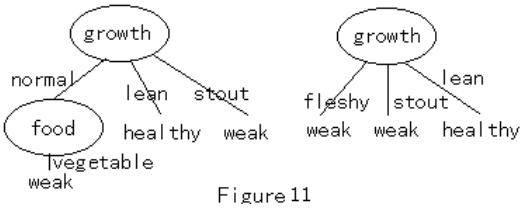

Figure11

# 5 Experimental Results

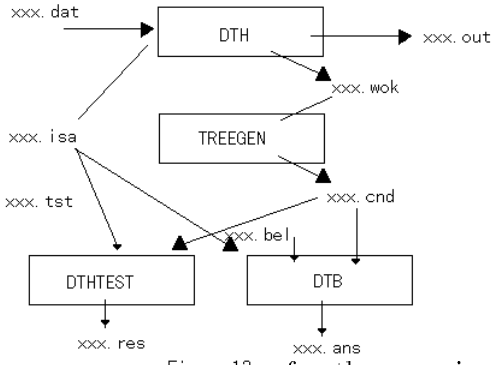

Figure12

Here are our programs for the experiment (fig. 12). `DTH` is for generation of a DTH tree from training data (`xxx.dat`) and taxonomy (`xxx.isa`). `TREEGEN` makes a sequence of tree conditions from the DTH. By `DTHTEST` we can examine test data (`xxx.tst`) against DTH.

For our experiments, let us explain some description of data and our software[2].

We examined *evaluation* information to status of reports submitted by students, by which we rank the acceptability according to check-roll, registration and exam-score. The information contain 282 tuples to 47 students over 3 attributes without any noise nor missing value. Some teacher's decision depends on check-roll (yes/no), report-registration (yes/no) and examination (0=worst,1,..,8,9,10=best). Decisions consist of score (1=best,2,3,4,5,6=worst). Here are some examples:

```
1 1 1 1
1 1 3 1
1 1 2 2
1 1 3 2
0 0 0 2
```

[2]We release these software through internet. Visit `http://www.dbl.k.hosei.ac.jp/`

```
1 1 1 1
1 0 4 3
.......
```

We added class hierarchy and attribute taxonomies (19 relationships). On *exam* we have 11 relationship and on classes we have 8 relationship. See fig 13.

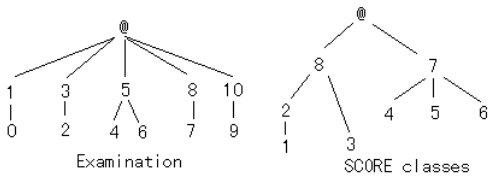

Figure 13

The following figure contains decision tree without any taxonomy (see figure 14). By applying DTH program to our score data, we got DTH in figure 15. The tree

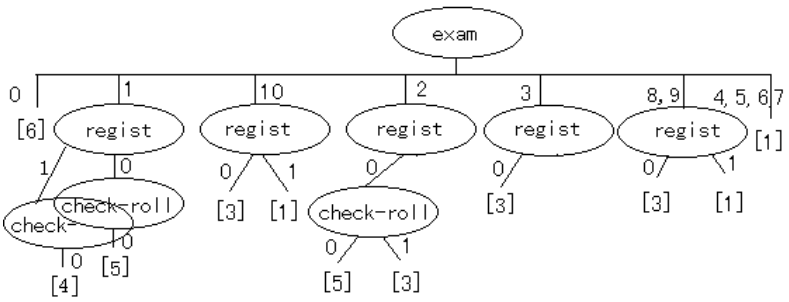

Figure14

contains 3 levels and 25 arcs. The main difference compared to figure 14 comes from attribute selection based on taxonomy. For example, exam="8" is an upper concept of exam="7", and we have a class 1 as a result, while, in figure 14, no consideration is made to this structure and some branch appears. On the other hand, exam="3" covers exam="2" which, in turn, many tuples (actually 98 tuples) are passed to examine classification. Thus we got fine sub trees.

In this experiment, we examined two tuples. The first one is $< @, @, 0 >$: 6 (exam=0 means score=6). This belief can be evaluated and we got success with least general class = 6.

The second tuple is $< no, @, @ >$: 7 (check-roll=no means score=7). The test failed since we got check-roll=3 means score = 3. Since the teacher didn't put stress on check-roll, there was no problem of this rejection.

# 6 Related Works

*Knowledge discovery in databases* (KDD) is one of the hot research topics and now

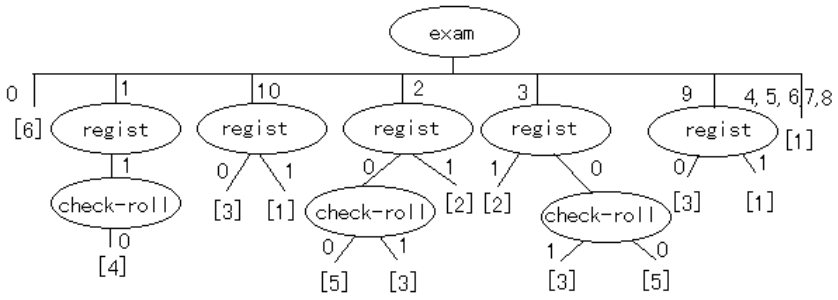


Figure15

focused by many database researchers[1, 3, 12]. It is nothing but knowledge acquisition activity but differs from using database as background (domain) knowledge.

Taxonomy is available many times in application domains and many research have been made. Multi-level association rule mining is one of such kinds of activities[14, 15]. In principle, all the ancestors are added to transactions to discover association rules. We don't traverse the taxonomies every time we add new one but we pre-compute to avoid overhead. Also some part of the sets are pruned for counting itemsets. Another ideas come from level-by-level abstraction[4]. The taxonomies are encoded and prepared for level-wise processes such as regular expressions and we can examine association rules quickly.

Series of the authors works show how to obtain new database schemes that are suitable for current database instances[5, 6, 7, 8, 9, 13].

# 7 Conclusion

In this work, we have proposed a new methodology of decision trees for classification rules based on taxonomy. Our new algorithm utilizes deduction of taxonomies and we have introduced some techniques of decision trees with hierarchy (DTH), that is, multi-level knowledge discovery. We proposed algorithms to generate DTHs based on conventional theory of entropy but the definition is extended carefully to capture taxonomies and class hierarchy. A tuple is evaluated in such way that we interpolate some rules. We have shown some experimental results to show how the approach is useful.

In other works, we discussed post analysis for assessment of DTH by using beliefs. The discovered results may often contradicts with the beliefs and they should be interesting to the users.

# References

[1] Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules, *VLDB* (1994), 487-499.

[2] Diettrich, T.: Overfitting and Undercomputing in Machine Learning, *ACM Comp.Survey* 27-3 (1995), pp.326-327

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

[4] Han, J. and Fu, Y.: Discovery of Multiple-Level Association Rules from Large Databases, *VLDB* (1995), 420-431

[5] Miura, T. and Shioya, I.: Mining Type Schemes in Databases, *Conference and Workshop of DEXA* (1996), 369-384

[6] Miura, T. and Shioya, I.: Paradigm for Scheme Discovery, proc.*CODAS* (1996), pp.101-108.

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

[8] Miura, T. and Shioya, I.: Incremental Update of Decision Trees for Temporal Information, Proc.*KRDB* (1999)

[9] Miura, T., Shioya, I. and Mori, M.: Making Decision Trees More Accurate By Losing Information, *Foundations of Information and Knowledge Systems* (2000), 213-225

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

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

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

[13] Shioya, I. and Miura,T. : Clustering Concepts from Databases, proc. *ICTAI* (1998), pp.95-102

[14] Srikant, R. and Agrawal, R.: Mining Generalized Association Rules, *VLDB* (1995)

[15] Srikant, R. and Agrawal, R.: Mining Quantitative Association Rules in Large Relational Tables, *SIGMOD* (1996)