Discovering Rules in Information Trees

Zbigniew W. Ras

Univ. of North Carolina, Comp. Science, Charlotte, N.C. 28223, USA
and
Polish Academy of Sciences, Comp. Science, 01-237 Warsaw, Poland

Abstract: The notion of an information tree has been formulated and investigated in 1982-83 by K.Chen and Z. Ras [1,2,3]. In [1] we have defined a notion of an optimal tree (the number of edges is minimal) in a class of trees which are semantically equivalent and shown how optimal trees can be constructed. Rules can be discovered almost automatically from information trees. In this paper we propose a formalized language used to manipulate information trees, give its semantics and a complete and sound set of axioms. This complete and sound set of axioms is needed for discovering rules in information trees assuming that conditional attributes and the decision attribute are not arbitrary but they are both provided by the user.

1 Introduction

Information trees and their query answering systems have been proposed and investigated in 1982-83 by K.Chen and Z. Ras [1,2,3]. The main difference between information trees and decision trees [5,6] lies in the interpretations and applications. Their structures and methods of constructions are often transferable to each other. In decision trees nodes are labeled by queries, edges by responses to these queries and leaves by some objects uniquely identified by the path from the root to the leaf. In an information tree, internal nodes are labeled by attributes and terminal nodes by sets of objects. A path from the root to a terminal node is interpreted as a description of objects labeling that node.

In [1] we proposed a heuristic polynomial algorithm to construct a minimal information tree with respect to the storage cost (storage cost was defined by us as a number of edges in a tree). It is worth to note that the problem of constructing even a minimal binary tree with respect to the storage cost is to be known as NP-complete. Information trees are quite useful in KDD area. Many certain rules can be discovered from the information tree in $0(k)$ steps where k is the height of the tree (number of attributes). Many possible rules can be generated in $0(n)$ steps where n is the number of nodes in a tree. The label of any edge (let us say e) in an information tree can be seen as a decision value of a rule. The conjunction of labels of all edges forming a path from the root of the tree to the edge e gives the condition part of the rule. Questions we would like to state in this paper say: can we discover rules from an information tree if condition attributes and a decision attribute are given by a user? Can we manipulate information trees algebraically into forms that are more convenient for a rule discovery than the initial trees?

To answer these questions we devise a representation of information trees as terms in a formal theory (theory of information trees) and present rules to manipulate

J.M. Zytkow and J. Rauch (Eds.): PKDD'99, LNAI 1704, pp. 518-523, 1999.

them. Next we show that the rules proposed by us are complete in constructing equivalent information trees. We also give a strategy for constructing equivalent information trees to a given information tree which are more convenient for discovering rules when conditional and decision attributes are given.

2 Basic Definitions

In this section we recall the definition of an information tree. Next we introduce the notion of equivalence of two information trees and the notion of one tree being covered by another.

Let U be a finite set of attributes called the universe. For each $a \in U$, let V_A be the set of attribute values of A. We assume that V_A is finite, for any $A \in U$. By an information tree on the universe U, we mean a tree $T=(N,E)$ such that:

(a) each interior node is labeled by an attribute from U,
(b) each edge is labeled by an attribute value of the attribute that labels the initial node of the edge,
(c) along a path, all nodes (except the leaf) are labeled with different attributes,
(d) all edges leaving a node are labeled with different attribute values (of the attribute that labels the node),
(e) a subset N_l of N is given, each node in N_l is called an object node.

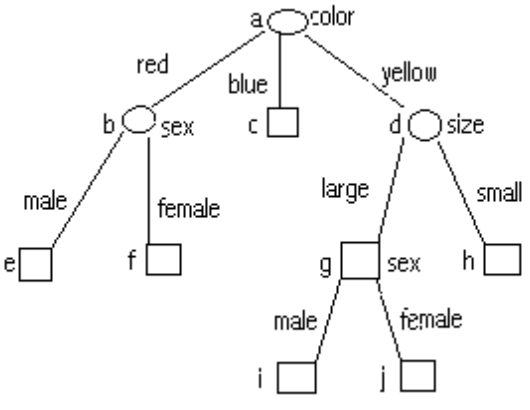

Figure 1. Information Tree

So, an information tree can be thought of as a triple (T, l, N_l) where $T = (N,E)$ is a tree, $N_l \subseteq N$ and l is the labeling function from $N_l \cup E$ into $U \cup (\bigcup\{V_A : A \in U\})$. The set N_l is a set of internal nodes in T.

Let m be an object node, $n_1, n_2, n_3, \ldots, n_k$ with $n_k = m$ be the path from the root n_1 to m. Objects node m determines an object type $0(m)=\{[l(n_i),l([n_i, n_{i+1}])]: i=1,2,\ldots,k-1\}$ where $l(n_i)$ is the label of the node n_i, which is an attribute. $l([n_i, n_{i+1}])$ is the label of the edge $[n_i, n_{i+1}]$ which is an attribute value of the attribute $l(n_i)$.

An information tree $S = ((N,E), l, N_l)$ determines a set of object types $0(S) = \{0(m): m \in N_l\}$. Two information trees S_1, S_2 are said to be equivalent if and only if $0(S_1) = 0(S_2)$. If $0(S_1) \subseteq 0(S_2)$, we say that S_1 is covered by S_2.

Figure 1 represents an information tree $S = ((N,E), l, N_l)$, where $N=\{a,b,c,d,e,f,g,h,i,j\}$, $E = \{[a,b], [b,e], [b,f], [a,c], [a,d], [d,g], [g,i], [g,j], [d,h]\}$, $l(a)$ = color, $l([a,b])$ = red, $l([a,c])$ = blue, $l([a,d])$ = yellow, $l(b)$ = sex, $l([b,e])$ = male, $l([b,f])$ = female, $l(d)$ = size, $l([d,g])$ = large, $l([d,h])$ = small, $l(g)$ = sex, $l([g,i])$ = male, $l([g,j])$ = female, $N_l = \{e,f,c,g,i,j,h\}$.

The object type of the node f is $\{[color, red], [sex, female]\}$. This information tree classifies seven different object types (determined by the seven nodes in N_l).

Assume that $\{[n,n_1], [n,n_2],\ldots, [n,n_k]\}$ is the set of all outgoing edges from the node n. Node n is called a-active (a is an attribute) if $l(n_1) = l(n_2) = l(n_k) = a$. Also, we say that k is the degree of the node n. The degree of an information tree is k if no node in the tree has the degree greater than k and there is a node in the tree of the degree k.

Information tree is called semi-complete if on any path from the root to a leaf of the tree the same attributes are listed (their order is immaterial).

Lemma 1. Assume that information tree is semi-complete. If a child of a node n is a leaf, then the parent of n is a-active for some attribute a.

Rules can be automatically generated from an information tree. For instance, let us assume that object node i in the information tree represented by Figure 1 contains 6 objects. Also, assume that object node j contains 4 objects and the object node h contains 3 objects. The following rules can be automatically generated:

1. (color = yellow) $\wedge$ (size = large) $\rightarrow$ (sex = male) with confidence 6/10
2. (color = yellow) $\rightarrow$ (size = large) with confidence 10/13.

However, if the user looks for a definition of attribute color in terms of attributes size and sex, the tree in Figure 1 has to be transformed into an equivalent tree which will be more suitable for the required knowledge extraction.

3 Formal Theory of Information Trees

In this section we shall define a formal syntax for representing information trees which is motivated by the LISP representation in [3] and co-algebra representation in [4]. We will introduce axioms and rules of inference for the formal theory of information trees.

Let us use the information tree from Figure 1 as a starting point in this section. Assume V_{color} = {red, blue, green, yellow} is ordered as red, blue, green, yellow; V_{sex} = {male, female} is ordered as male, female; V_{size} = {large, medium, small} is ordered as large, medium, small. Then the following term

color(sex(-, -), -, *, size(sex(-, -), *, -))

retains all the information about the information tree from Figure 1. An underline means the node is an object node. A star means the corresponding subtree is empty.

To describe an information tree, we use the general scheme attribute($subtree_1$, $subtree_2$, …, $subtree_n$) assuming the attribute has n different values.

Now we are ready to introduce a formal theory of information trees over an attribute universe U. There are two constant symbols * , - which have standard interpretation: empty tree and single node tree (tree with one node being an object node) respectively.

For each attribute A in U with $|V_A| = n$ (where $|V_A|$ denotes the number of attribute values of A), there are two n-ary function symbols f_A , $\underline{f}_A$. The standard interpretation of $f_A(t_1, t_2, \ldots, t_n)$ is the information tree with the root labeled A (drawn as circle) and the next level subtrees $t_1 , t_2 , \ldots, t_n$.

The standard interpretation of $\underline{f}_A(t_1, t_2, \ldots, t_n)$ is the information tree with the root labeled A which is an object node (drawn as square) and next level subtrees $t_1 , t_2 , \ldots, t_n$.

Function symbol f_A is called a *type 0* function symbol, $\underline{f}_A$ is called a *type 1* function symbol. There is one predicate symbol ≡ . Statement $t_1 \equiv t_2$ in the standard interpretation says that t_1 and t_2 are equivalent.

Terms are defined by the following recursive definition:

DEFINITION OF TERMS.

(a) constant symbols are terms,

(b) if g is n-ary function symbol, $t_1, t_2, \ldots, t_n$ are terms not containing g or its dual type function symbol, then $g(t_1, t_2, \ldots, t_n)$ is a term.

Intuitively, each term represents an information tree.

If a term does not contain any type 1 function symbol or the constant symbol -, it is called *null object* term.

The nested level h(t) of a term t is defined as follows:

$h(*) = h(-) = 0$,

$h(f_A(t_1, t_2, \ldots, t_n)) = h(\underline{f}_A(t_1, t_2, \ldots, t_n)) = \max\{h(t_i): n \geq i\} + 1$.

Let t be a term, we use I(t) to denote the standard interpretation of t, i.e., the information tree that t represents.

We have: H(t)=n if and only if the height of I(t) is n.

If t is a term then by $\underline{t}$ we mean a new term defined below:

$\underline{t}$ = [if t = * , then -] else

[if $t = f_A(t_1, t_2, \ldots, t_n)$, then $\underline{f}_A(t_1, t_2, \ldots, t_n)$] else t

The formulas are defined by the following recursive definition:
(a) $t_1 \equiv t_2$ is a formula for any two terms t_1, t_2
(b) $p \wedge q, p \vee q, p \rightarrow q, p \leftrightarrow q, \neg p$ are formulas if p, q are formulas.

Our **formal theory** has the following axiom schemata:
A1. (reflexive) $t \equiv t$ is an axiom for any term t ,
A2. (nullity) $* \equiv t$ for any null object term t ,
A3. (change the order of branching)
$f(g(t_{1,1}, t_{1,2}, \ldots, t_{1,m}), g(t_{2,1}, t_{2,2}, \ldots, t_{2,m}), \ldots, g(t_{n,1}, t_{n,2}, \ldots, t_{n,m}))$
$\equiv f(g(t_{1,1}, t_{2,1}, \ldots, t_{n,1}), g(t_{1,2}, t_{2,2}, \ldots, t_{n,2}), \ldots, g(t_{1,m}, t_{2,m}, \ldots, t_{n,m}))$
is an axiom for any two type 0 function symbols f, g where f is n-ary, g is m-ary and for any n.m terms $t_{i,j}$ $(i \leq n, j \leq m)$ not containing f, g or their type 1 duals ,
A4. $p \rightarrow (q \rightarrow p)$ for any formulas p, q ,
A5. $(p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$ for any formulas p, q, r ,
A6. $(\neg p \rightarrow \neg q) \rightarrow (q \rightarrow p)$ for any formulas p, q .

The **rules of inference** for our formal system are the following:
R1. from $p \rightarrow q$ and p we deduce q for any formulas p, q ,
R2. from $t_1 \equiv t_2$ we deduce $t(t_1) \equiv t(t_2)$, where $t(t_1)$ is a term containing t_1 and $t(t_2)$ comes from $t(t_1)$ by replacing some of the occurrences of t_1 with t_2 ,
R3. From $t_1 \equiv t_2$, we can deduce $\underline{t}_1 \equiv \underline{t}_2$

Let t be a term, we shall use $I(t)$ to denote the information tree represented by t under the standard interpretation. Then we have the following completeness theorem.

Theorem 1. $t_1 \vdash t_2$ if and only if $I(t_1)$ is equivalent to $I(t_2)$.

4 Discovering Rules in Information Trees

In this section we suggest a strategy for discovering rules in information trees when condition and decision attributes are provided by the user.

We start with an example of an information tree represented by the term
A(B(C(-,-),C(-,-)), *, C(B(-,*),*), C(*,B(-,*))) .
Assume now that our plan is to describe A (decision attribute) in terms of B and C (classification attributes). We use axioms A2, A3 and rule R2 repeatedly to replace the term above by a new equivalent term
B(C(A(-,*,-,*), A(-,*,*,-)), C(A(-,*,*,*), A(-,*,*,*))).
The goal of our strategy is to move attribute A below all classification attributes. In a case of attributes B and C, we prefer to place B above C because there is an equal number of object nodes having property c_1 and c_2 whereas there are two object nodes having property b_2 and four object nodes having property b_1. Now, if a node labeled by attribute A has only one outgoing edge then edges along a path from the root to that node define the classification part of a rule. In our example we get two rules: $b2 \wedge c1 \rightarrow a1$ and $b2 \wedge c2 \rightarrow a1$.

Now, we apply axioms A2, A3 and rule R2 again to move up the nodes labeled by attribute A assuming that these nodes both currently and in a resulting tree have only one outgoing edge. Term

B(C(A(-,*,-,*), A(-,*,*,-)), A(C(-,-),*,*,*))

represents the final resulting information tree (called [A;B,C]-optimal) which is equivalent to the initial tree. We have only one rule describing A in terms of B and C which is: $b2 \rightarrow a1$. Clearly this rule is optimal.

Lemma 2. Assume that information tree is semi-complete and its degree is s. Let $n_1, n_2,\ldots, n_k$ be all children of the node n and $l(n_1)=a$. The problem of converting n to a-active node is in the worst case $0(s^k)$ where k is the height of the complete subtree with a root n_1. We count here the number of times the rules of inference are applied.

5 Conclusion

Information trees investigated in this paper satisfy the assumption that on the path from the root of a tree to a leaf there cannot be two nodes labeled by the same attribute. Trees allowing repeated attributes on a path from the root of a tree to a leaf were investigated by Cockett [4] and used in the implementation of CASCADE system. Formal theory of information trees and its completeness theorem with respect to the predicate ≡ allows us to manipulate information trees using axioms A2, A3, rules R2, R3 and preserve its semantical meaning. Also, we know that any two information trees which are semantically equivalent can be transformed from one to another using only axioms A2, A3, A4, A5, A6 and rules R1, R2, R3. Assume that a quest q which requires to find an optimal description of an attribute A in terms of attributes A_1, $A_2,\ldots$, A_k queries an information tree T. Our goal is to find $[A;A_1,A_2,\ldots,A_k]$-optimal information tree which is semantically equivalent to T. Example in the last section of this paper gives us some ideas how such trees can be constructed using the formal theory of information trees.

References

1. Chen K, Ras Z.W., Homogeneous information trees, *Fundamenta Informaticae*, 8, 1985, 123-149.
2. Chen K, Ras Z.W., DDH approach to information systems, *Proceedings of the 1982'CISS in Princeton, N.J.*, 521-526.
3. Chen K, Ras Z.W., Dynamic hierarchical data bases, *Proceedings of the 1983'ICAA in Taipei, Taiwan*, 450-456.
4. Cockett R., The algebraic co-theory of decision processes, *Technical Report CS-84-58, University of Tennessee*.
5. Quinlan, J.R., Induction of decision trees, *Machine Learning, 1, 1986, 81-106. Reprinted in J.W. Shavlik & T.G. Dietterich (Eds.), Readings in machine learning, San Francisco, CA, Morgan Kaufmann, 1990*
6. Quinlan, J.R., Generating production rules from decision trees, Proceedings of the Tenth International Joint Conference on Machine Learning, Morgan Kaufmann, 1987, 304-307