Web: www.witpress.com Email witpress@witpress.com
Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

A method for knowledge representation and discovery based on composing and manipulating logical equations

D.E. Sitnikov[1], B.D'Cruz[2] & P.E. Sitnikova[3]
[1]*Kharkov State Academy of Culture, Ukraine*
[2]*University College Northampton, United Kingdom*
[3]*Ukrainian People's Academy, Ukraine*

Abstract

When information about objects and processes is formalised in knowledge bases, there is often the need to deal with features that have discrete values, for example gender (male/female) or marital status (married/single/divorced). When the mutual influence of discrete features is investigated, it is important to find out how closely the variables representing these features are connected and how strongly they influence each other. It is also important to obtain analytical dependencies in the data. This paper focuses on the logical correlation in knowledge bases, and suggests a method for knowledge representation and discovery of dependencies between selected data features. A knowledge base containing links between information features is represented in the form of logical equations with finite predicates. Special classes of equations describing complicated hierarchical data structures are considered. New knowledge about logical links in the data are obtained by eliminating variables from these equations through procedures involving the use of both quantifiers, and the role of each quantifier is discussed. The results of these procedures are dependencies between the features that are easier to interpret than the dependencies represented by the original equations. The elimination procedure does not lead to an increase in the size of the original knowledge base, and there are possibilities for parallel calculations. The knowledge base can be written in a text file with the help of a subset of the extensible mark-up language (XML), which makes it easy to transfer and manipulate the data. Some examples are discussed for the purpose of illustration.

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

Introduction

When information about objects and processes is formalised in knowledge bases, there is often the need to deal with features that have discrete values, for example gender (male/female), marital status (married/single/divorced), or financial state (low, medium, high). When such information has a complicated logical structure, various methods that use discrete mathematics and logical equations can be applied to formally represent this. Logical equations are often applied to pattern recognition problems that have complicated logical dependencies. Methods of pattern recognition using logical equations usually deal with variables taking on the values 0 or 1 depending on whether or not a given object has a particular property. Such Boolean variables denote the properties and features in the objects being recognised by considering definite combinations of these variables, from which the presence or absence of object properties can be determined [1]. Dependencies between the given variables are written in the form of logical equations, and these are then transformed mathematically to discover new knowledge about the objects and their associated properties [2].

A discrete feature can take on any value from a finite set, e.g. the feature *colour* can be "black", "white", "blue", or "green" etc. For our purposes, it is convenient to use finite predicates for the representation of logical dependencies. The formal language of finite predicate algebra allows the combination of an algebraic approach to pattern recognition with different tools of predicate calculus [3]. Finite predicate algebra utilises the operations of conjunction, disjunction and negation as used in Boolean algebra, but these are applied to predicates of the variables that can take on discrete values, and not directly to the variables as used in Boolean algebra. When formulae in finite predicate algebra are constructed, so-called "recognition" predicates are used. For example, x^a is a predicate that is equal to 1 if and only if the variable x takes on the value a. It is said that this predicate "recognises" the symbol a. Equations of finite predicate algebra are a natural generalization of Boolean algebra equations, and they allow manipulating arbitrary feature variables defined on finite sets of elements [3]. Using such equations for building inferences in knowledge bases allows an extension to the potential of logical methods for pattern recognition and knowledge discovery through the identification of salient features [4]. This has implications for data mining defined as: "*the process of extracting and refining useful knowledge from databases ... extracted information can be used to form a prediction or classification model, identify trends and associations, refine an existing model, or provide a summary of the database(s) being mined.*" [5]. This paper describes a mathematical method for knowledge representation and discovery by composing and manipulating logical equations, and this has various potential applications for knowledge discovery in data mining.

Discovering logical links between discrete features

When the mutual influence of discrete features is investigated, it is very important to find out how closely the variables representing these features are

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

connected and how strongly they influence each other. Statistical methods for knowledge discovery in relational databases address this question by regarding the *statistical correlation* between feature variables [6]. This paper focuses on the *logical correlation* in knowledge bases that can be represented by logical equations where these equations may be considered as closely related to *deductive databases,* given that they have data (feature values) and deductive inference rules (logic and algebraic rules for manipulating logical equations). With logical equations, it is possible to represent intentional data and construct deductive inferences in a similar manner as has been the case using deductive databases and machine learning approaches [7]. The main difference between these two methodologies lies in the fact that performing deductive reasoning with the help of logical equations supposes the solving of these equations by simplifying them instead of building chains of rules. Thus our approach is algebraic rather than logical.

As far as the quantitative measure of mutual influence of features is concerned, it is usually said that the smaller the number of sets of feature values that satisfy the equation, the stronger the formal logical link between the features. If any possible sets of values of the variables satisfy the equation, then there is no logical link between the features [4]. In other words, the fewer the degrees of freedom allowed by the variables in a logical equation, the stronger the logical link. This measure is rather formal and does not reflect the logical nuances hidden in the knowledge base. For example, a particular value of feature x may strictly define the only possible value of feature y. However, another value of the same feature x may be co-located with any value of feature y. In some cases the values of feature variables are known *a priori*, and this knowledge may alter previous hypotheses regarding the other variables. In this case the following question arises: how strong is the logical dependence between two or more selected features, and what does this dependence look like?

To answer this question, we eliminate from the original equation all the non-salient variables [4] using a quantifier $\exists$. We can then investigate the resulting equation with a smaller number of variables that describe all possible combinations of values for the target features.

The main difficulty encountered in trying to eliminate discrete variables lies in the fact that the formal application of the operation $\exists$ to a predicate leads to an exponential increase in the size of the original formula. To simplify the elimination procedure we used the following properties of the quantifier $\exists$:

1. $\exists x\, x^a = 1$
2. $\exists x\, (P(x)\ V\ Q(x)) = \exists x\, P(x)\ V\ \exists x\, Q(x)$
3. $\exists x\, (P(x)\ \&\ Q(y)) = \exists x\, P(x)\ \&\ Q(y)$
4. $\exists y\, (P(x) \rightarrow Q(y)) = P(x) \rightarrow \exists y\, Q(y)$
5. Suppose $P_i(x)\ \&\ P_j(x) = 0,\ i \neq j,\ i, j = 1, 2, \ldots, k$. Then:

$$\exists y\, ((P_1(x) \rightarrow Q_1(y))\ \&\ (P_2(x) \rightarrow Q_2(y))\ \&\ \ldots\ \&\ (P_k(x) \rightarrow Q_k(y))) =$$

$$= (P_1(x) \rightarrow \exists y\, Q_1(y))\ \&\ (P_2(x) \rightarrow \exists y\, Q_2(y))\ \&\ \ldots\ \&\ (P_k(x) \rightarrow \exists y\, Q_k(y)).$$

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

6. If the identity $P_i(x) \equiv 0$ is not true for $i = 1, 2, ..., k$ and $P_i(x) \,\&\, P_j(x) = 0$ for $i \neq j, i, j = 1, 2, ..., k$ then:

$$\exists x ((P_1(x) \rightarrow Q_1(y)) \,\&\, (P_2(x) \rightarrow Q_2(y)) \,\&\, ... \,\&\, (P_k(x) \rightarrow Q_k(y))) =$$
$$= Q_1(y) \vee Q_2(y) \vee ... \vee Q_k(y).$$

Properties 1 to 6 allow us to define special predicate types that can be more easily processed with algorithms for variable elimination. In a previous paper we recursively constructed the predicate class Δ_x [4]. For any predicate of this class one can easily eliminate the variable x with the quantifier $\exists$. The algorithm for the elimination of the variable x looks for blocks connected with the conjunction or disjunction operations, and spreads the quantifier $\exists$ throughout the formula using properties 1 to 6. We must note here that knowledge of the hierarchical structure of the original formula facilitates the elimination of variables, and if we do not know the structure of the formula in advance we will therefore have to recognize it. At present, we do not have an effective algorithm that would allow us to find out whether a predicate belongs to the class Δ or not. Nevertheless, where we can represent knowledge about discrete object features as a predicate formula belonging to the above class, we will be able to quickly eliminate variables. In such a case the complexity of the algorithm grows linearly with respect to the length of the formula, and there is the possibility of parallel calculations dramatically reducing the time required for elimination. For example, if we consider the expression:

$$P_1 \vee P_2 \vee ... \vee P_n,$$

where the predicates $P_1, P_2, ..., P_n$ belong to Δ_x. we can see that the elimination of the variable x can be performed for each predicate in parallel. Hence if we have tools for parallel calculations, the time required for elimination will be equal to the maximum time of the elimination of the variable in the predicates $P_1, P_2, ..., P_n$ taken separately.

It may be argued that the requirement for exact information about the formula structure is too strong a restriction to be placed on the knowledge base. The counter argument is that for relational representations, there are much stronger limitations on the knowledge structure anyway. Let us consider the following relational table with the fields x, y and z:

x	y	z
a_1	b_1	c_1
a_2	b_2	c_2

We can represent this table as a logical equation in the perfect disjunctive normal form:

$$x^{a1} y^{b1} z^{c1} \vee x^{a2} y^{b2} z^{c2} = 1.$$

In the general case it is obvious that any relational table can be represented in the form of a logical equation, the formula on the left side of the equation being in the perfect disjunctive normal form. The class Δ of finite predicates described in

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

[4] includes all possible disjunctive (not necessarily perfect) normal forms. This means that the way of knowledge representation in the form of logical equations containing formulae that belong to the class Δ is more general than relational representation. Of course using such a logical method makes sense only in cases where there are many complicated logical dependencies in data features.

As a result of the elimination of variables with the help of the quantifier $\exists$, we obtain the sets of values of the salient features for which *there exists at least one possible set* of values of the other features. However, if we want to obtain the sets of the salient features that satisfy the equation *irrespective of* the values that the other features take on, we should eliminate variables with the help of the quantifier $\forall$. When applying this quantifier we encounter the same complexity problems as in the case of using the quantifier $\exists$. To resolve the problems connected with exponential growth in the size of the formula in a logical equation, we use the following properties of the quantifier $\forall$:

7. $\forall x\, x^a = 0$
8. $\forall x\, (P(x) \,\&\, Q(x)) = \forall x\, P(x) \,\&\, \forall x\, Q(x)$
9. $\forall x\, (P(x) \vee Q(y)) = \forall x\, P(x) \vee Q(y)$
10. $\forall y\, (P(x) \,\&\, Q(y)) = P(x) \,\&\, \forall y\, Q(y)$
11. Suppose $P_i(x) \,\&\, P_j(x) = 0,\ i \neq j,\ i, j = 1, 2, \ldots, k$. Then

$$\forall y\, ((P_1(x) \,\&\, Q_1(y)) \vee (P_2(x) \,\&\, Q_2(y)) \vee \ldots \vee (P_k(x) \,\&\, Q_k(y))) =$$

$$= (P_1(x) \,\&\, \forall y\, Q_1(y)) \vee (P_2(x) \,\&\, \forall y\, Q_2(y)) \vee \ldots \vee (P_k(x) \,\&\, \forall y\, Q_k(y))$$

12. If the identity $P_i(x) \equiv 0$ is not true for any $i = 1, 2, \ldots, k$ and $P_i(x) \,\&\, P_j(x) = 0$ for $i \neq j,\ i, j = 1, 2, \ldots, k$ then:

$$\forall x\, ((P_1(x) \,\&\, Q_1(y)) \vee (P_2(x) \,\&\, Q_2(y)) \vee \ldots \vee (P_k(x) \,\&\, Q_k(y))) =$$

$$= Q_1(y) \,\&\, Q_2(y) \,\&\, \ldots \,\&\, Q_k(y).$$

The predicates $P_i(x)$ in properties 11 and 12 can be interpreted as hypotheses for possible values of the feature x, the predicates $P_i(x)$ defining domains for values of this feature that do not have common elements.

We must also mention here that if the additional condition

$$\vee P_i(x) = 1 \tag{1}$$

is satisfied, then the expressions

$$(P_1(x) \rightarrow Q_1(y)) \,\&\, (P_2(x) \rightarrow Q_2(y)) \,\&\, \ldots \,\&\, (P_k(x) \rightarrow Q_k(y))) \tag{2}$$

and

$$((P_1(x) \,\&\, Q_1(y)) \vee (P_2(x) \,\&\, Q_2(y)) \vee \ldots \vee (P_k(x) \,\&\, Q_k(y)) \tag{3}$$

are formulae representing the same predicate (of course the predicates $P_i(x)$ should satisfy the conditions indicated in properties 6 and 12). This statement

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

allows us to simplify the elimination procedure for both quantifiers $\exists$ and $\forall$ since we can often transform logical equations into a form that is convenient for application of a particular quantifier. Condition (1) can be interpreted as the requirement for the hypotheses $P_1(x), P_2(x), \ldots, P_k(x)$ to cover all possible values of the discrete feature x.

Let us now recursively define a class of finite predicates for which the elimination of variables with the help of the quantifier $\forall$ does not lead to any increase in the size of the original formula. Consider the set Σ of finite predicates with the set of variables $\{x, y, \ldots, z\}$. Let us define a subset Φ_x of Σ as follows:

- All the predicates $x^a, x^b, \ldots, x^c$ recognizing symbols from the domain for the variable x belong to Φ_x.
- All the predicates that do not depend on the variable x belong to Φ_x.
- If predicates P_1 and P_2 belong to Φ_x then the predicate $P = P_1 \vee P_2$ belongs to Φ_x.
- If a predicate P_1 belongs to Φ_x, and a predicate P_2 does not depend on x, then the predicate $P = P_1 \vee P_2$ belongs to Φ_x.
- If a predicate P_1 does not depend on x, and a predicate P_2 belongs to Φ_x, then the predicate $P = P_1 \& P_2$ belongs to Φ_x.
- If predicates $P_1, P_2, \ldots, P_k$ do not depend on x; $P_i \& P_j = 0$ for $i \neq j, i, j = 1, 2, \ldots, k$; predicates $Q_1, Q_2, \ldots, Q_k$ belong to Φ_x; then the predicate $P = (P_1 \& Q_1) \vee (P_2 \& Q_2) \vee \ldots \vee (P_k \& Q_k)$ belongs to Φ_x.
- If predicates $P_1, P_2, \ldots, P_k$ depend only on x; $P_i \& P_j = 0$ for $i \neq j, i, j = 1, 2, \ldots, k$; for any $i = 1, 2, \ldots, k$ the identity $P_i \equiv 0$ is not true; predicates $Q_1, Q_2, \ldots, Q_k$ do not depend on x; then the predicate $P = (P_1 \& Q_1) \vee (P_2 \& Q_2) \vee \ldots \vee (P_k \& Q_k)$ belongs to Φ_x.

The algorithm for the elimination of the variable x looks for blocks connected with the conjunction or disjunction operations, and spreads the quantifier $\forall$ throughout the formula using properties 7 to 12. As in the case of the elimination procedure using the quantifier $\exists$, to effectively eliminate variables with the help of the operation $\forall$ we must know the hierarchical structure of the original formula in advance. If the structure is known, then the time of calculation grows linearly with respect to the length of the original formula. Similar to the case of using the quantifier $\exists$, there is the possibility of eliminating variables in parallel since properties 7 to12 allow us to apply the quantifier to a block of a formula without knowing the result of the elimination of variables in the other blocks.

Let us consider now a simplified example that illustrates elimination of variables with the help of both quantifiers. Suppose it is known that a patient's disease can be A or B or C, and the following discrete features can be observed for these diseases or different strains of the same disease: temperature (low, normal, high); blood pressure (low, normal, high); pulse (slow, normal, quick). It is known that these are the only possible diseases, and that two diseases cannot happen at the same time:

- If disease A occurs, the temperature is low or normal; if the temperature is low then the blood pressure is low and the pulse is slow

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

- If disease B occurs, the temperature is high, the pulse is slow or normal; if the pulse is slow then the blood pressure is low
- If disease C occurs, the pulse is quick, the temperature is high and no information is available on the blood pressure.

We can formalise these dependencies as follows. Let us introduce the variables *d, t, b,* and *p*, where *d* is the disease, *t* is the temperature, *b* is the blood pressure and *p* is the pulse. The variable *d* takes on values from the set {A, B, C}, the variables *t* and *b* can take on values from the set {low, normal, high}, the variable *p* can take on values from the set {slow, normal, quick}. The above dependencies can be described in the following way (we omit the conjunction sign between the elementary "recognition" predicates):

$$(d^{A} \to (t^{low} \vee t^{normal}) \& (t^{low} \to b^{low} p^{slow})) \& (d^{B} \to t^{high} (p^{slow} \vee p^{normal}) \& (p^{slow} \to b^{low})) \& (d^{C} \to p^{quick} t^{high}) = 1.$$

The solutions of the above equation (the sets of the variables' values satisfying the equation) form a relational table with 3x3x3x3=81 rows, which demonstrates the fact that using logical equations instead of relational tables allows us to describe complicated logical links concisely. On simplifying this equation we get:

$$(d^{A} \to (t^{normal} \vee t^{low} b^{low} p^{slow})) \& (d^{B} \to t^{high} (p^{slow} b^{low} \vee p^{normal})) \& (d^{C} \to p^{quick} t^{high}) = 1. \quad (4)$$

Let us denote

$S = (d^{A} \to (t^{normal} \vee t^{low} b^{low} p^{slow})) \& (d^{B} \to t^{high} (p^{slow} b^{low} \vee p^{normal})) \& (d^{C} \to p^{quick} t^{high})$, $S1 = d^{A} \to (t^{normal} \vee t^{low} b^{low} p^{slow})$, $S2 = d^{B} \to t^{high} (p^{slow} b^{low} \vee p^{normal})$, $S3 = d^{C} \to p^{quick} t^{high}$, $S11 = d^{A}$, $S12 = t^{normal} \vee t^{low} b^{low} p^{slow}$,

$S121 = t^{normal}$, $S122 = t^{low} b^{low} p^{slow}$, $S1221 = t^{low}$, $S1222 = b^{low}$, $S1223 = p^{slow}$ *etc.*

Then

$S = S1 \& S2 \& S3$, $S1 = S11 \to S12$, $S12 = S121 \vee S122$, $S122 = S1221 \& S1222 \& S1223$ etc.

and we can see that our model has a tree-like hierarchical structure where the "leaves" are elementary "recognition" predicates.

Let us investigate the logical links between the temperature and blood pressure features. If we want to get *all possible links* between the features in question we will need to use the quantifier $\exists$. The hierarchical structure of the original model allows us to "spread" the quantifier across the formula without any increase in its size as all the predicates *S, S1, S2, S3* etc. obviously belong to the class Δ for any variable. Using property 6, we can eliminate variable *d* from the original equation to get:

$$(t^{normal} \vee t^{low} b^{low} p^{slow}) \vee t^{high} (p^{slow} b^{low} \vee p^{normal}) \vee p^{quick} t^{high} = 1. \quad (5)$$

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

Using the properties 1 to 3 we can eliminate variable p to get:

$$(t^{\text{normal}} \vee t^{\text{low}} b^{\text{low}}) \vee t^{\text{high}} (b^{\text{low}} \vee 1) \vee t^{\text{high}} = t^{\text{normal}} \vee t^{\text{low}} b^{\text{low}} \vee t^{\text{high}} \vee t^{\text{high}}$$
$$= t^{\text{normal}} \vee t^{\text{low}} b^{\text{low}} \vee t^{\text{high}} = 1. \quad (6)$$

We can see from eqn (6) that only when the temperature is low can we be sure that the blood pressure will be low. If the temperature is high or normal, we cannot say anything about the blood pressure.

In case we wish to investigate *unconditional logical links* between discrete features i.e. links that do not depend on values of the other features, we should use the quantifier $\forall$. Let us investigate unconditional links between the temperature and the disease in the above example. Before we apply the quantifier, let us transform the formula (4) using the fact that if condition (1) is satisfied, then (2) is equivalent to (3). In our example $d^{\text{A}} \vee d^{\text{B}} \vee d^{\text{C}} = 1$, therefore we can represent (4) in the following form:

$$d^{\text{A}}(t^{\text{normal}} \vee t^{\text{low}} b^{\text{low}} p^{\text{slow}}) \vee d^{\text{B}} t^{\text{high}} (p^{\text{slow}} b^{\text{low}} \vee p^{\text{normal}}) \vee d^{\text{C}} p^{\text{quick}} t^{\text{high}} = 1.$$

Using the hierarchical structure of this formula (obviously belonging to the class Φ for any variable) we spread the quantifier without any increase in the size of the formula. First we eliminate the variable b using property 11:

$$\forall b(d^{\text{A}}(t^{\text{normal}} \vee t^{\text{low}} b^{\text{low}} p^{\text{slow}}) \vee d^{\text{B}} t^{\text{high}} (p^{\text{slow}} b^{\text{low}} \vee p^{\text{normal}}) \vee d^{\text{C}} p^{\text{quick}} t^{\text{high}})$$
$$= d^{\text{A}}(t^{\text{normal}} \vee \forall b(t^{\text{low}} b^{\text{low}} p^{\text{slow}})) \vee d^{\text{B}} t^{\text{high}} (\forall b(p^{\text{slow}} b^{\text{low}}) \vee p^{\text{normal}}) \vee d^{\text{C}} p^{\text{quick}} t^{\text{high}}$$
$$= d^{\text{A}} t^{\text{normal}} \vee d^{\text{B}} t^{\text{high}} p^{\text{normal}} \vee d^{\text{C}} p^{\text{quick}} t^{\text{high}}.$$

Using properties 9 and 12 we can eliminate the variable p:

$$\forall p(d^{\text{A}} t^{\text{normal}} \vee d^{\text{B}} t^{\text{high}} p^{\text{normal}} \vee d^{\text{C}} p^{\text{quick}} t^{\text{high}}) = d^{\text{A}} t^{\text{normal}} \vee \forall p(d^{\text{B}} t^{\text{high}} p^{\text{normal}} \vee$$
$$d^{\text{C}} p^{\text{quick}} t^{\text{high}}) = d^{\text{A}} t^{\text{normal}} \vee (d^{\text{B}} t^{\text{high}} \,\&\, d^{\text{C}} t^{\text{high}}) = d^{\text{A}} t^{\text{normal}} \vee 0 = d^{\text{A}} t^{\text{normal}}.$$

Thus if d = A then t = normal, and if t = normal then d = A *for any values* of the features b and p, which means that we have obtained the unconditional link between the target features.

A method for knowledge representation

The logical models considered in this paper allow us to develop a method for knowledge representation and discovery, which promises to be quite effective when dealing with data of complicated logical structure. It can be easily seen that the classes Δ [4] and Φ of predicate formulae are much more general than disjunctive normal forms. If we take into consideration the fact that any relational table can be represented as a logical equation in the perfect disjunctive form (we have given a simple example before), then it becomes clear that the above classes of equations allow us to describe more complex data structures than with relational tables. Moreover, it is very convenient to build deductive

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

inferences based on logical equations of this type. Nevertheless, a practical question arises: how can we represent the above data structures in computer programs? To answer this question we have developed a special mark-up language LHXML, which is a subset of XML (extensible mark-up language). In this language special tags are used to denote:

- different levels of the hierarchy of a logical formula,
- logical operations (we use "empty" tags for this purpose)
- names of features (we use "opening" and "closing" tags)

Using LHXML we can write down our example (4) in a text file:

```
<root_level>
        <level1>
                <d>A</d><implication/>
                <level2>
                        <t>normal</t>
                        <or/>
                        <level3>
                        <t>low</t><and/><b>low</b><and/><p>slow</p>
                        </level3>
                </level2>
        </level1>
        <and/>
        <level1>
        <d>B</d><implication/>
                <level2>
                        <t>high</t>
                        <and/>
                        <level3>
                                <level4>
                                <p>slow</p> <and/> <b>low</b>
                                </level4>
                                <or/>
                                 <p>normal</p>
                        </level3>
                </level2>
        </level1>
        <and/>
        <level1>
                <d>C</d><implication/>
                <level2>
                <p>quick</p> <and/> <t>high</t>
                </level2>
        </level1>
</root_level>
```

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

If we use LHXML then the process of obtaining inferences from such data structures just means manipulating XML nodes [8], which corresponds to the procedures for eliminating variables as described above. We would like to stress that the representation of logical dependencies with the help of XML allows us to use the whole capability of the XML object model (i.e. all the methods and properties of the XML elements) and hence quickly draw logical conclusions from data (applying standard XML techniques and tools). We are now in the process of developing specialized data structures based on our method that are expected to be used in the development of a new generation billing system.

Summary

In this paper we have suggested a method for knowledge representation and discovery based on composing and manipulating logical equations of special types. We have considered the process of discovering hidden patterns in complex data structures that are more general than relational databases. In order to discover new patterns in data, we apply logic quantifiers and obtain dependencies that are simpler and clearer than the dependence defined by the original knowledge base. We have demonstrated some possibilities of parallel calculations for obtaining deductive inferences from the model, which can dramatically reduce the speed of knowledge discovery. We have also developed a special mark-up language (a subset of XML) that allows us to store data in a text file and discover hidden patterns in the data by manipulating XML nodes.

References

[1] Gorelik, A.L., Skripkin, V.A.; (1984), *Recognition Methods*, Moscow: Visha Shkola, pp.82-120.
[2] Gorelik, A.L., Gurevich, I.B., Skripkin, V.A.; (1985), *The Current State of the Recognition Problem*, Moscow: Radio I Svyaz, pp.98-103.
[3] Shabanov-Kushnarenko, U.P.; (1984), *Theory of Intelligence: Mathematical Tools*, Kharkov: Visha Shkola, pp.12-41.
[4] Sitnikov, D.E.; D'Cruz, B.; Sitnikova, P.E.; (2000), *Discovering Salient Data Features Based on Composing and Manipulating Logical Equations*, in Ebecken, C.; Brebbia, C.A.; (eds.), Data Mining II: Proc. 2nd International Conference (Cambridge), WIT Press, pp.241-248.
[5] Simoudis, E.; Livezey, B.; Kerber, R.; (1996), *Integrating Inductive and Deductive Reasoning for Data Mining,* in Fayyad, U.M.; Piatetsky-Shapiro, G.; Smyth, P.; Uthurusamy, R.; (eds.), Advances in Knowledge Discovery and Data Mining, The MIT Press, pp.353-373.
[6] SAS Institute (1998): *SAS/Warehouse Administrator User's Guide Release 1.3*, First Edition, SAS Institute Inc., Cary, NC, USA.
[7] Flach, P.A.; Savnik, I.; (1999), *Database Dependency Discovery: A Machine Learning Approach*, AI Communications, 12(3), Nov, pp.139-160.
[8] Skonnard, A.; Gudgin, M.; (2001), *Essential XML Quick Reference*, Addison Wesley.