Data Mining II, C.A. Brebbia & N.F.F. Ebecken (Editors)

Discovering salient data features by composing and manipulating logical equations

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

Abstract

The paper suggests a method of representing knowledge in the form of logical equations of a special type. We show that deductive inferences about the salient data features in a knowledge base can be represented by a set of logical equations. We describe how logical equations with finite predicates can be successfully used for the description of logical links between discrete features, and how this can be applied to pattern recognition and data mining. New knowledge about logical links between discrete features in the data can be obtained by eliminating variables from these equations with the help of the operation $\exists$, and we describe this process in detail. We consider the application of the operation $\exists$ to a logical equation as an analogue to a query in a database. The process results in a dependence between the features subsequent to application of the elimination procedure that is easier to interpret than the dependence represented in the original equations, and is obtained without the need for exhaustive searching. We recursively define a class of finite predicates that allow eliminating variables without increasing the size of the original equation, and show how this can be applied for knowledge discovery using logical data modelling.

1. Introduction

Researchers in data mining have long been interested in making accurate generalisations from a few scattered facts, or discovering patterns from

Data Mining II, C.A. Brebbia & N.F.F. Ebecken (Editors)

seemingly chaotic patterns of observations [1]. There are currently numerous data mining and pattern recognition algorithms that attempt to do this, but an important stumbling block to using many of these is the way in which these algorithms define the logical structure of the problem [2]. Some algorithms employ inductive inference techniques whereby the problem is formulated so that the system must discover a set of rules representing knowledge or features in a given data set. The system generates and validates various hypotheses on the basis of facts or observations from the data set, and this is essentially extracting or "mining" for new rules or features. In this paper, we describe a method that represents the knowledge in a data set as logical equations, and how these equations are then manipulated using finite predicate algebra to eliminate non-salient and irrelevant components. This is the basis for discovering the more salient features and making new inferences about the data. The benefits and limitations of the technique are considered, and we suggest that this approach is beneficial in terms of reducing the complexity of the discovery process and also in interpreting the outcome.

2. Logical modelling for pattern recognition

Methods of pattern recognition using logical modelling usually deal with Boolean 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 [3]. Dependencies between the given variables are written in the form of logical equations that are then transformed to discover new knowledge about the objects and their properties [4]. When dealing with logical pattern recognition problems, the logical equation should not be interpreted as an expression into which some identity must be substituted to get a solution, but as a premise from which it is possible to draw logical inferences given restrictions on the form of representation being imposed. Attempts to develop effective methods for solving logical equations of arbitrary type have failed because there are many difficulties associated with sorting intermediate solutions, essentially a strongly branching process. To overcome these difficulties, it has been useful to classify logical equations according to their features, and then develop methods for their solution taking into account specific features of each class. As far as universal methods of solving Boolean equations are concerned, one of the main methods used involves rearranging the Boolean expression in an equation into its Perfect Disjunctive Normal Form, and this requires exhaustive searching in many cases [5]. Such a method does not allow specific inferences to be drawn from the original equation, but merely allows the enumeration of sets of values for the variables. An earlier paper [6] suggested an approach to obtaining general analytical solutions to logical

Data Mining II, C.A. Brebbia & N.F.F. Ebecken (Editors)

equations, however this approach does not give an effective algorithm that can process large numbers of variables. In the next section we present an approach to logical data modelling for knowledge discovery based on eliminating feature variables without the need for exhaustive searching.

3. Identifying and Extracting Salient Features

A discrete feature can take on not only the values 0 and 1 but any value from a finite set, e.g. the feature "colour" can be "black", "white", "blue", "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 [7]. 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 letter a.

The necessity for exhaustive searching is one of the main problems arising when logical equations with finite predicates are solved. We suggest that one of the possible ways to solve such a problem is by the elimination (using the operation $\exists$) of feature variables that are not needed for the solution of a specific recognition problem. A similar elimination approach for data mining was applied to decision tables using probabilistic rough sets where IF-THEN rules were extracted by means of eliminating duplicate values and superfluous attributes from the decision table [8]. In terms of logical equations, elimination means that we are able to obtain an equation that contains the smallest number of variables necessary for solving the problem, such an equation being inferred from the original one. Besides, such a method identifies any logical link between the remaining variables after the elimination. If after applying the elimination procedure the identity 1=1 is obtained, then such a link does not exist. Similarly, the smaller the number of sets of feature values that satisfy the resulting equation, the stronger the link between the feature variables remaining after the elimination. In particular, in order to learn whether two arbitrary features depend on each other, we should eliminate all the other variables from the original equation. We can thus describe the problem of discovering new knowledge about dependencies between feature variables as follows:

Consider the predicate equation:

$$P(x_1, x_2, \ldots, x_n) = 1. \qquad (1)$$

Data Mining II, C.A. Brebbia & N.F.F. Ebecken (Editors)

P is a formula in finite predicate algebra that describes the recognised links between the features $x_1, x_2, \ldots, x_n$ for each of the variables taking on a finite number of values. We should eliminate the variables $x_i, x_j, \ldots, x_k$ $(1 \leq i < j < \ldots < k \leq n)$ by using an effective algorithm to obtain the equation:

$$\exists x_i \exists x_j \ldots \exists x_k P(x_1, x_2, \ldots, x_n) = 1. \quad (2)$$

Eqn (2) should not contain the variables $x_i, x_j, \ldots, x_k$ after the application of the elimination operation $\exists$ to the left side of the original equation. The formal application of the operation $\exists$ to the predicate P leads to an increase in the size of the original formula. For instance if we let $x_1 \in \{a, b, c\}$, then:

$$\exists x_1 G(x_1, x_2) = G(a, x_2) \vee G(b, x_2) \vee G(c, x_2). \quad (3)$$

In this case the formula size has increased three times. When large numbers of variables are eliminated, it is practically impossible to obtain eqn (2) from eqn (1). However, in some equations we can eliminate variables without any increase in the formula size by using specific properties of the operation $\exists$. Here are some of these properties:

1. $\exists x\, x^a = 1$

2. $\exists x (P(x) \vee Q(x)) = \exists x P(x) \vee \exists x Q(x)$

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. Let $P_i(x) \& P_j(x) = 0$, $i \neq j$, $i, j = 1, 2, \ldots, k$ then :

$$\begin{aligned} &\exists y ((P_1(x) \rightarrow Q_1(y)) \& (P_2(x) \rightarrow Q_2(y)) \& \ldots \& (P_k(x) \rightarrow Q_k(y))) = \\ &= (P_1(x) \rightarrow \exists y\, Q_1(y)) \& (P_2(x) \rightarrow \exists y\, Q_2(y)) \& \ldots \\ &\& (P_k(x) \rightarrow \exists y\, Q_k(y)). \end{aligned} \quad (4)$$

6. 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:

$$\begin{aligned} &\exists x ((P_1(x) \rightarrow Q_1(y)) \& (P_2(x) \rightarrow Q_2(y)) \& \ldots \\ &\& (P_k(x) \rightarrow Q_k(y))) = \\ &= Q_1(y) \vee Q_2(y) \vee \ldots \vee Q_k(y). \end{aligned} \quad (5)$$

Data Mining II, C.A. Brebbia & N.F.F. Ebecken (Editors)

In eqns (4) and (5), the predicates $P_i(x)$ (i = 1, 2, …, k) can be interpreted as hypotheses on possible values of x. The properties 1 to 4 are obvious, let us now prove property 5. For an arbitrary x there are two options:

- There exists such i that $P_i(x) = 1$. In this case $P_j(x) = 0$ for any $j \neq i$, and eqn (4) turns into the identity $\exists x (P_i(x) \rightarrow Q_i(y)) = P_i(x) \rightarrow \exists y\, Q_i(y)$
- For any i = 1, 2, … k, $P_i(x) = 0$, then eqn (4) turns into the identity 1 = 1

Let us also prove property 6. For any predicateG(x) defined on the finite set {a, b, …, c}:

$$\exists x\, G(x) = G(a)\ V\ G(b)\ V \ldots V\ G(c). \tag{6}$$

Using this property, we apply the operation ∃ to the expression on the left side of eqn (5):

$$(P_1(x) \rightarrow Q_1(y)) \,\&\, (P_2(x) \rightarrow Q_2(y)) \,\&\, \ldots \\ \&\, (P_k(x) \rightarrow Q_k(y)). \tag{7}$$

It is clear that for each replacement of the variable x with one of its values, the left part of eqn (4) turns into one of the predicates $Q_i(y)$ (i = 1, 2, … k). Since each of the predicates $P_i(x)$ (i = 1, 2,…, k) is not equal to 0 for some x, then each of the predicates $Q_i(y)$ (i = 1, 2, …k) will be present in the resulting expression.

Properties 1 to 6 allow us to define special predicate types that can be more easily processed with algorithms for variable elimination. Consider the set Σ of finite predicates with the set of variables {x, y, …, z}. Let us define a subset Δ_x of Σ as follows:

- All the predicates x^a, x^b, …, x^c "recognizing" letters from the domain for the variable x belong to Δ_x
- All the predicates that do not depend on the variable x belong to Δ_x
- If predicates P_1 and P_2 belong to Δ_x then the predicate $P = P_1\ V\ P_2$ belongs to Δ_x
- If a predicate P_1 belongs to Δ_x, and a predicate P_2 does not depend on x, then the predicate $P = P_1 \,\&\, P_2$ belongs to Δ_x
- If a predicate P_1 does not depend on x, and a predicate P_2 belongs toΔ_x, then the predicate $P = P_1 \rightarrow P_2$ belongs to Δ_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, …, k; predicates $Q_1, Q_2, \ldots, Q_k$ belong to Δ_x; then the predicate $P = (P_1 \rightarrow Q_1) \,\&\, (P_2 \rightarrow Q_2) \,\&\, \ldots \,\&\, (P_k \rightarrow Q_k)$ belongs to Δ_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,…, k; for any i = 1, 2, …, 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 $P = (P_1 \to Q_1) \,\&\, (P_2 \to Q_2) \,\&\, \ldots \,\&\, (P_k \to Q_k)$ belongs to Δ_x.

Such a recursive definition allows us to construct and recognize quite complicated formulae from which the variable x can be readily eliminated. Let us consider a simplified example that illustrates possible ways of using logical equations with finite predicates for encoding and manipulating information features. 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 kinds of the same disease: temperature (low, normal, high); blood pressure (low, normal, high); pulse (slow, normal, quick). It is known *a priori* 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
- 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, 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:

$$(d^{\mathrm{A}} \to (t^{\mathrm{normal}} \vee t^{\mathrm{low}} b^{\mathrm{low}} p^{\mathrm{slow}})) \,\&\, (d^{\mathrm{B}} \to t^{\mathrm{high}} (p^{\mathrm{slow}} b^{\mathrm{low}} \vee p^{\mathrm{normal}})) \\ \&\, (d^{\mathrm{C}} \to p^{\mathrm{quick}} t^{\mathrm{high}}) = 1. \qquad (8)$$

Let us investigate the logical links between the temperature and blood pressure features. Using property 6, we can eliminate variable *d* from the original equation to get:

$$(t^{\mathrm{normal}} \vee t^{\mathrm{low}} b^{\mathrm{low}} p^{\mathrm{slow}}) \vee t^{\mathrm{high}} (p^{\mathrm{slow}} b^{\mathrm{low}} \vee p^{\mathrm{normal}}) \vee p^{\mathrm{quick}} t^{\mathrm{high}} = 1. \qquad (9)$$

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

$$(t^{\mathrm{normal}} \vee t^{\mathrm{low}} b^{\mathrm{low}}) \vee t^{\mathrm{high}} (b^{\mathrm{low}} \vee 1) \vee t^{\mathrm{high}} \\ = t^{\mathrm{normal}} \vee t^{\mathrm{low}} b^{\mathrm{low}} \vee t^{\mathrm{high}} \vee t^{\mathrm{high}} \\ = t^{\mathrm{normal}} \vee t^{\mathrm{low}} b^{\mathrm{low}} \vee t^{\mathrm{high}} = 1. \qquad (10)$$

Data Mining II, C.A. Brebbia & N.F.F. Ebecken (Editors)

We can see from eqn (10) 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. Thus eliminating some of the variables from the original equation we can discover some hidden patterns for particular combinations of features. The elimination of variables looks like a query to a database, but using relational tables for the description of complicated logical links between discrete features results in a huge number of rows. Even in this simplified example we would need to write out 3x3x3x3 = 81 rows to construct the truth table for the finite predicate on the left side of eqn (10), and if we had ten variables taking on three values each we would need to investigate 3^{10}= 59049 rows. As can be seen above, using special types of logical equations with finite predicates allows us to describe logical links concisely and get necessary information on particular features quickly and effectively.

4. Summary

In this paper, we have suggested that finite predicate algebra is a convenient tool for the description of logical links between discrete features, and is highly applicable to knowledge discovery in data mining. This method allows us to make deductive inferences about the salient data features in a knowledge base as represented by the logical equations. New knowledge about logical links between discrete features in the data can be obtained by eliminating variables from the equations with the help of the operation $\exists$. The formal application of the operation $\exists$ to a finite predicate leads to a significant increase in the size of the original formula and is therefore not effective for eliminating large numbers of variables. However, some of the properties of the operation $\exists$ allow us to effectively eliminate variables from logical equations of a special type without increasing the size of the formula. The process results in a dependence between the features subsequent to application of the elimination procedure that is easier to interpret than the dependence represented in the original equations, and is obtained without the need for exhaustive searching through relational tables that would require a large number of rows.

References

[1] Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. & Uthurusamy, R. (eds). *Advances in Knowledge Discovery and Data Mining*, AAAI/The MIT Press: Cambridge, Mass., 1996.

[2] Wayner, P. AI is Becoming a Reality as Pattern-Recognition Programs Can Now Prove, *Byte*, August 1995.

[3] Gorelik, A.L., Skripkin, V.A. *Recognition Methods*, Moscow: Visha Shkola, pp.82-120, 1984.

[4] Gorelik, A.L., Gurevich, I.B., Skripkin, V.A. *The Current State of the Recognition Problem*, Moscow: Radio I Svyaz, pp.98-103, 1985.

[5] Zakrevsky, A.D. *Logical Equations*, Minsk: Nauka I Technika, 1975.

[6] Sitnikov, D.E., Shabanov-Kushnarenko, U.P. About Solving Boolean Algebra Equations, *Problems of Bionics,* 40, pp.15-19, 1988.

[7] Shabanov-Kushnarenko, U.P. *Theory of Intelligence: Mathematical Tools*, Kharkov: Visha Shkola, pp.12-41, 1984.

[8] Zhong, N., Dong, J-Z., Ohsuga, S. Data Mining: A Probabilistic Rough Set Approach. Polkowski, L., Skowron, A. (eds.) *Rough Sets in Knowledge Discovery*, Warsaw: Physica-Verlag, pp. 127-144, 1998.