Proceedings of the 7th International Conference on Applied Informatics
Eger, Hungary, January 28–31, 2007. Vol. 2. pp. 377–384.

Generating decision tree from lattice for classification

László Kovács

ME University of Miskolc, Department of Information Technology
e-mail: kovacs@iit.uni-miskolc.hu

Abstract

The Formal Concept Analysis is a powerful tool to create a generalization lattice for a given context. Beside this main application, the concept lattice can be used as a tool to generate all closed attribute sets and to measure the relationships between the class lables and the attribute sets. In our approach, the generated lattice will be used directly to perform the class assignment. During the preparation phase, a decision tree will be generated from the lattice.

1. Formal concept analysis basics

In a wide area of soft computing, the methods of formal concept analysis (FCA) are increasingly used to discover the object clusters and the generalization relationships inherent in the corresponding attribute values. Wille [14] proposed first to consider the elements in a Galois lattice as concepts. The Galois lattice is based on a binary relationship between the objects and attributes. The edges of the lattice represent the generalization connection. The FCA is used nowdays as a powerful tool in a wide variety of soft computing areas among others for data analysis, information retrieval, knowledge discovery, association rule discovery, software engineering. In this section, a short recall of the basic notations of FCA theory will be given [14].

The set of all o objects is denoted with O. The objects are assigned to sets of attributes. The symbol a denotes an attribute and the A is the set of all attributes.

Definition 1.1. A context K is given with a $K(O_K, A_K, I_K)$ triplet, where $O_K \subseteq O$, $A_K \subseteq A$ and $I_K \subseteq O_K \times A_K$ is the binary relationship between the objects and attributes.

The K context is usually given with an object-attribute matrix where every cell has a 1 or 0 value. For a given $X \subseteq O_K$ set, the context determines a set of attributes common in each objects in X. In the reverse direction, a $Y \subseteq A_K$

relates to a set of objects owing each attributes in Y. These mappings are given in the following functions.

Definition 1.2. The function h maps a set of objects to the corresponding set of attributes:

$$h_K : P(O_K) \to P(A_K) : h_K(X) = \{a | a \in A_K, \forall o \in X : oI_K a\}.$$

The function g maps a set of attributes to the corresponding set of objects:

$$g_K : P(A_K) \to P(O_K) : g_K(X) = \{o | o \in O_K, \forall a \in X : oI_K a\}.$$

The $P(X)$ symbol denotes here the powerset of X. The chaining of the h and g functions denotes a closure as it means the maximal set of objects (or attributes) having the common attributes (or objects).

Definition 1.3. The closure function:

$$c_K : P(O_K) \to P(O_K) : c_K(X) = g_K(h_K(X))$$

$$c'_K : P(A_K) \to P(A_K) : c'_K(X) = h_K(g_K(X)).$$

The unit elements of the closure operation, i.e. where $X = c_K(X)$ or $X = c'_K(X)$ are called closed subsets. The pair of closed object and attribute subsets is called formal concept.

Definition 1.4. A formal concept is a couple (X, Y) where $X \in P(O_K)$, $Y \in P(A_K)$, $h_K(X) = Y$ and $g_K(Y) = X$. The X is the extent and Y is the intent part of the concept.

It follows from the definition that $X = c_K(X)$ and $Y = c'_K(Y)$ are also met. Taking the set of all concepts, an ordering relationship can be defined based on the inclusion of extent parts.

Definition 1.5. The ordering ($\leqslant_K$) of the concepts: $(X_1, Y_1) \leqslant_K (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2$.

The inclusion relationship between two extent parts implies an inverse ordering relationship between the intent parts:

$$Y_1 \supseteq Y_2 \Leftrightarrow X_1 \subseteq X_2.$$

The set of concepts with the $\leqslant_K$ partial ordering meets the definition of a complete lattice.

Property 1.6. *The set of concepts in a $K(O_K, A_K, I_K)$ context is a complete lattice where the infima and suprema of any subset of concepts are also elements of the lattice. The infima an suprema concepts are defined on the following way:*

$$\wedge(X_i, Y_i) = (\cap X_i, c'_K(\cup Y_i)$$

$$\vee(X_i, Y_i) = (c_K(\cup X_i), \cap Y_i).$$

The generated concept lattice for a $K(O_K, O_K, I_K)$ context is usually presented with a Hasse diagram where only the neighbouring concepts are connected with each others. As an example, let us take following input values from the example in [6]:

$$O = \{o_1, \ldots, o_5\}$$

$$A = \{a_1, \ldots, a_9\}$$

$$I = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

There are twelve non-trivial concepts generated from this context. The corresponding Hasse diagram is shown in Figure 1.

2. Classification with concept analysis

The main goal of the classification algorithms is the assignment of a class to the objects based on the object's attribute values. Usually, the different attributes have different correlation with the class labels. The concept lattice can be used as a tool to generate all closed attribute sets and to measure the relationships between the class labels and the attribute sets. One of the first proposals to apply a concept lattice for classification problems is [15]. In this model, one of the attributes is marked as class label. A classification rule describes the dependeny of the class labels from the logical formulas.

Definition 2.1. Taking the $K(O_K, A_K, I_K)$ context, a classification rule is given in the form $f \Rightarrow c$ where a_c is a class attribute, c is class-label and $f \in F_K^*$. The goodness of a rule is measured with the confidencie and generality properties, where

$$conf(f \Rightarrow c) = \frac{|m(f \cap (a_c = c)|)}{|m(f)|}$$

$$generality(f \Rightarrow c) = \frac{|m(f))|}{|O_K|}.$$

The higher is the confidence value the more accurate is the classification rule.

Definition 2.2. A consistent classification rule is a classification rule with a confidence value 1, i.e. $|m(f \cap c)| = |m(f)|$.

The $m(f)$ symbol denotes the set of objects meeting the f predicate. A conjuctive concept (X, f) is called a consistent concept if it implies a unique class label and the confidence value is equal to 1.

A most general consistent concept is a consistent concept where neither of the super concepts is consistent.

It can be shown that the set of most general consistent concepts is covering all of the universe. Thus the set of most general consistent concepts is enough to perform the classification process.

In a brute-force approach to generate this set, the lattice of all concepts is built up first. Then, the sets of consistent concepts and the set of most general consistent concepts are filtered out. The most ineffective part of the algorithm is the first step, the generation the whole lattice. In [15], the PRISM algorithm is proposed to generate the most general consistent concepts on a heursitic way. The algorithm consists of the following steps:

1. Generating the set of consistent classification rules $\{\phi\}$ using the following algorithm:
   (a) Loop for every class labels c.
   (b) Calculating the probability of $(c|\phi)$.
   (c) Selection of rules with maximal probability values.
2. Generating a consistent definition pair for every rule $\{(g_K(\phi), \phi)\}$
3. Constructing the set of conjuctive consistent concepts $\{(g_K(\phi), h_K(g_K(\phi)))\}$
4. Eliminating the conjuctive consistent concepts that are not most general concepts.

The experiments [15] show that the classificication based on concept lattice can achieve a higher description accuracy than the widely used ID3 algorithm. The main benefits of the lattice-based approach is the systematic processing of all possible closed groups of formulas to filter out the formulas with highest correlation to a class label.

3. Converting lattice into decision tree

The generated lattice is used to perform a classification task. The goal is to predict the class label from the attributes for any object $w \in W$. In our approach, the generated lattice will be used directly to perform the class assignment. During the preparation phase, a decision tree will be generated from the lattice. The classification process is based on the following considerations.

Let Λ denote the concept lattice generated from the given context on the usual way. The lattice is extended with labels to perform the classification task. The resulted extended lattice is denoted by Λ^*. From efficency reasons, a default class label is also introduced for the concepts.

Definition 3.1. Given a Λ=$(\{A_i, B_i\}, \leqslant)$ lattice the Λ^* extended lattice is a tuple $(\{A_i, B_i\}, \leqslant, s, c, d)$ where s and c are the support and class labels of the concepts:

$$s : T \to N^+$$

$$c : T \to C.$$

The T symbol denotes the set of (A_i, B_i) concepts in Λ. The support value is equal to the number of dominated concepts (using the $\leqslant$ relation). The class value is defined as the intersection of the nonempty class values on the set of descendants. The d function denotes the default class value:

$$d : T \to C.$$

The d value is defined as the nonempty class value with highest support within the descendants.

In the classification, the concepts with unambiguous class label are the basic information sources for the class assignment. These concepts are called consistent concepts. The maximal consistent concepts are such consistent concept that have no consistent dominator concepts. The maximal consistent concepts can be used to determine the class label for a string pattern.

Definition 3.2. A c_1 concept in Λ^* is consistent if $|c(c_1)| = 1$. The set of consistent concepts in Λ^* is denoted with C^c_Λ.
A c_1 concept in Λ^* is called maximal consistent concept if

$$c_1 \in C^c_\Lambda, \neg\exists c_2 \in C^c_\Lambda : c_1 \leqslant c_2.$$

The class assignment algorithm is based on the following considerations:

Lemma 3.3. *The following implication rules are met in the Λ^* lattice:*

- *If there exists only one maximal consistent concept for a class value c and the intent part of this concept is d, then:*

  $$d \Leftrightarrow c$$

  This means, that c occurs if and only if , the d occurs. In this case, the corresponding decision tree may contain only one decision node for the class c.

- *For every consistent concepts for a class value c where the intent part is d, the following implication holds:*

  $$d \Rightarrow c$$

  In this case, the decision tree will have a node for d having a leaf child node with c.

The given implication rules can be used to imply the value of the class attribute from the corresponding content attributes. The set of the derived implications is organized into a decision tree. The decision tree structure is easy to undertstand and it provides an algorithmic approach. In the proposed method, the decision tree is generated from the concept lattice using the following rules.

Definition 3.4. A concept lattice Λ can be converted into a decision tree Δ for determining the class attribute from the content attributes. The Δ is a binary rooted tree, where each node is assigned to a generalized word. A not leaf node has two children edges labeled with the 'True' and 'False' logical values. On a node with generalized word w^+ a w' word passes to the 'True' egde if and only if $w' \leqslant w^+$.

In the normal way of lattice transformation, the words outside of the training pool can not be assigned to any class. In the proposed version of the decision tree building algorithm, the leaves with unknown class value are eliminated. This leaf will be assigned to the class value with the highest probability within the parents domain. The domain a concept denotes the set of dominated concepts. In order to measure the probability, a support value is assigned to every concept node. The support value corresponds to the number of descendant objects. The default class within the domain of a concept is the class of the child with highest support value. Using this support value, not only the unknown class values are eliminated but also a tree reduction can be performed. In the reduced tree, all of the child nodes belonging to the most probably class labels are eliminated and only the exception nodes are preserved. These modifications require a new tree building algorithm.

The conversion is performed with the following valid transformation steps.

```
transform a lattice
L : input lattice
D : output decision tree
lattice_to_tree () {

  D.root = {}; /* empty word, it dominates all words */
  t = L.top;   /* the suprenum of all concepts, the 1 element */

  process (t,D.root, L,D);

}

transform a concept
t : concept
p : parent node in D
L : lattice
D : decision tree
process (t,p, L,D) {

 Ch = the set of children concepts of t;
 order Ch by decreasing support order;
 cd = the dominant class value for t;
 eliminate the concepts with class cd from Ch;
 foreach c in Ch
```

```
    d = the intent of c;
    create a node in D with d (=n);
    if this is the first child then
      join n to the 'True'  edge of p;
    else
      join n to the 'False' edge of p;
    endif
    if c is unmarked then
      if c is a maximal consistent concept then
         add a leaf with class value of c to the
         'True' edge of n;
      else
         process (c,L,D);
      endif
      p = n;
    endif
  endfor
  add a leaf with class value cd to the 'False'
  edge of last n;
  set all children of c to be marked ;
}
```

In order to make the decision tree more efficient some additional generalization elements were also introduced into the algorithm.

References

[1] Browing, M., Null Operator Constructions, Ph.D. thesis, *MIT*, (1987).

[2] Chomsky, N., Aspects of the Theory of Syntax, Cambridge, *MIT Press*, (1965).

[3] Chomsky, N., Formal properties of grammar, (1963).

[4] Ferre, S., Ridoux, O., A logical generalization of formal concept analysis, (2001).

[5] Gildea, D., Jurafsky, D., Automatic Induction of Finite State Transducer for Simple Phonological Rules, *Meeting of ACL*, (1995).

[6] Godin, R., Missaoui, R., Alaoui, H., Incremental concept formation algorithms based on Galois (concept) lattices, *Computational Intelligence*, (1995), 246–267.

[7] Harris, Z., Methods in Structural Linguistics, *University of Chicago Press*, (1951).

[8] Jurafsky, D., Martin, J. H., An introduction to speech recognition, computational linguistics and natural language processing, (2006).

[9] Krenn, B., Samuelsson C., The Linguistic's Guide to Statistics, (1997).

[10] Manning, C., Schütze, H., Foundations of Statistical Natural language Processing, *MIT Press*, (1999).

[11] Shannon, C., Prediction and entropy of printed English, *Bell System Technical Journal*, (1951).

[12] Valtchev, P., Missaoui, R., Building concept (Galois) lattices from parts: generalizing the incremental methods.

[13] Wallace, C., Seneca Morphology, *International Journal of American Linguistic*, (1960).

[14] Wille, R., Restructuring Lattice Theory: an Approach Based on Hierarchies of Concepts, *Ordered Sets*, Reidel, (1982).

[15] Zhao, Y., Yao, Y., Classification based on logical concept analysis, (2006).