International Journal of General Systems
Vol. 38, No. 4, May 2009, 455–467

Taylor & Francis
Taylor & Francis Group

Inducing decision trees via concept lattices[1]

Radim Belohlavek[ac], Bernard De Baets[b], Jan Outrata[c]* and Vilem Vychodil[ac]

[a]*Department of Systems Science and Industrial Engineering, T. J. Watson School of Engineering and Applied Science, Binghamton University – SUNY, Binghamton, NY 13902, USA;* [b]*Department of Applied Mathematics, Biometrics, and Process Control, Ghent University, Coupure links 653, B-9000 Gent, Belgium;* [c]*Department of Computer Science, Palacky University Olomouc, Tomkova 40, CZ-779 00 Olomouc, Czech Republic*

(*Received 21 December 2007; final version received 7 October 2008*)

We present a novel method for the construction of decision trees. The method utilises concept lattices in that certain formal concepts of the concept lattice associated to input data are used as nodes of the decision tree constructed from the data. The concept lattice provides global information about natural clusters in the input data, which we use for selection of splitting attributes. The usage of such global information is the main novelty of our approach. Experimental evaluation indicates good performance of our method. We describe the method, experimental results, and a comparison with standard methods on benchmark datasets.

Keywords: decision trees; classification; machine learning; concept lattice; formal concept analysis

1. Introduction

Decision trees represent the most commonly used method in data mining and machine learning (Quinlan 1993, Dunham 2003, Tan *et al.* 2006). A decision tree is typically used for a classification of objects into a given set of classes based on the objects' attributes. Many algorithms for the construction of decision trees are proposed in the literature, see e.g. (Tan *et al.* 2006).

This paper presents a novel approach to decision tree construction, which is based on formal concept analysis (FCA) of the input data. Our approach utilises concept lattices in that certain formal concepts associated to input data are used as nodes of the decision tree constructed from the input data. The concept lattice provides global information about natural clusters, represented by formal concepts, in the input data. Using formal concepts as nodes of decision trees is a straightforward idea because both formal concepts and decision tree nodes represent collections of records (objects) in the input data defined by having the same values for certain attributes. A challenge exists in how to select good formal concepts for decision tree nodes. We attempt to consider a concept lattice (without the least formal concept) as a collection of overlapping trees. The construction of a decision tree is then reduced to the problem of selection of one of these trees.

FCA and concept lattices are utilised in several machine learning and decision tree induction algorithms proposed in the literature. For instance, Carpineto and Romano (1996) present GALOIS, a clustering method based on concept lattices, in which similarity

*Corresponding author. Email: jan.outrata@upol.cz

ISSN 0308-1079 print/ISSN 1563-5104 online

DOI: 10.1080/03081070902857563
http://www.informaworld.com

between objects and clusters is defined as the number of common attributes shared by objects. In (Mephu Nguifo and Njiwoua 2001), the authors use FCA in IGLUE, a method which selects relevant categorical attributes and transforms them into continuous numerical attributes which are then used to solve a decision problem by k-nearest neighbour clustering. Another approach utilising FCA is described in Kuznetsov (2004) which presents a model of learning from positive and negative examples. Fu *et al.* (2004) provides a survey and theoretical and experimental comparison of several FCA-based classification algorithms. Note that FCA-based approaches are commonly called lattice-based or concept-based learning techniques in data mining (Fayyad *et al.* 1996, Pasquier *et al.* 1999).

The paper is organised as follows. The next section contains preliminaries from decision trees and formal concept analysis. In Section 3 we present our approach including an algorithm for inducing decision trees. The algorithm is accompanied by an illustrative example. An experimental evaluation of our method and a comparison to standard methods on benchmark datasets is provided in Section 4. Section 5 presents conclusions and outlines topics for future research.

2. Preliminaries

2.1 *Decision trees*

A decision tree can be considered as a tree representation of a function over attributes which takes a finite number of values. The goal is to construct a tree that approximates a given function, partially described by a table containing records in its rows, with a desired accuracy. Every record consists of particular values of the function input values (attribute values) and the corresponding output value (class label). For an object described by its attribute values, the value assigned by the decision tree to those attribute values is considered the label of a class to which the object belongs. A good decision tree is supposed to classify well both the data described by the table records as well as 'unseen' data.

Each non-leaf node of a decision tree is labelled by an attribute, called a splitting attribute for this node. Such a node represents a test, according to which records are split into n classes which correspond to n possible outcomes of the test. In the basic setting, the outcomes are represented by values of the splitting attribute. Leaf nodes of the tree represent collections of records all of which, or the majority of which, have the same function value (class label). For example, the table in Figure 1 (top) describes a partial function $f : A \times B \times C \to D$. The decision trees in Figure 1 (bottom) represent two functions, both of which are extensions of f.

A strategy commonly used in the existing algorithms for inducing decision trees from data consists of constructing a decision tree in a top-down fashion, from the root node to the leaves, by successively splitting existing nodes and creating new ones. For every node, a splitting attribute is chosen to split the collection of records covered by the node into smaller collections, which correspond to values of the splitting attribute. For every such value, a new node is attached as a child to the node for which the splitting attribute has been chosen. The process continues recursively until all the records corresponding to any leaf node, or a prescribed majority of them, belong to the same class. A critical point in this strategy is the selection of splitting attributes, for which many approaches were proposed. These include the well-known approaches based on entropy measures, Gini index, classification error, or other measures defined in terms of class distribution of the records before and after splitting (see Quinlan 1996, Murthy 1998, Tan *et al.* 2006 for overviews).

A	B	C	f(A,B,C)
good	yes	false	yes
good	no	false	no
bad	no	false	no
good	no	true	yes
bad	yes	true	yes

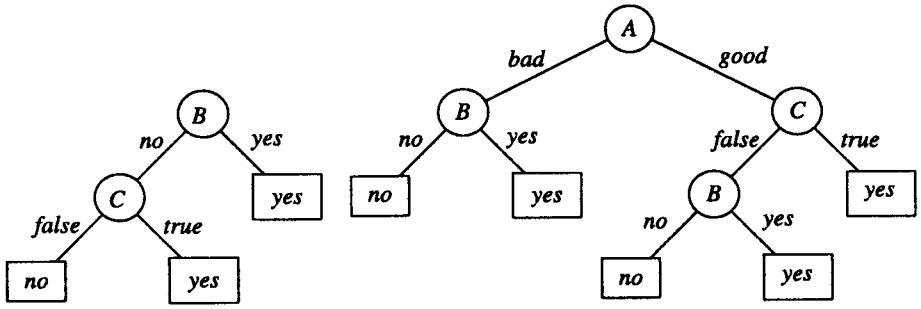


Figure 1. Two decision trees (bottom) representing functions which extend the partial function f (top).

2.2 *Formal concept analysis*

FCA is a method for analysis of object-attribute data (Ganter and Wille 1999, Carpineto and Romano 2004). Such data is usually described by a table with rows and columns representing objects and attributes, respectively, and with table entries containing attribute values, which the objects have. In the basic setting, FCA deals with binary attributes, i.e. every attribute applies or does not apply to a particular object. Many-valued attributes, such as nominal and ordinal attributes, are transformed to binary ones using so-called conceptual scaling. FCA produces two kinds of output from a given dataset. The first output is called a concept lattice. A concept lattice is a partially ordered collection of particular clusters called formal concepts. The second output consists of a non-redundant base of particular attribute dependencies called attribute implications.

We now summarise basic notions of FCA. An object-attribute data table can be identified with a triplet $\langle X, Y, I\rangle$ where X is a non-empty set of objects, Y is a non-empty set of attributes, and $I \subseteq X \times Y$ is an object-attribute relation. Objects and attributes correspond to table rows and columns, respectively, and $\langle x, y\rangle \in I$ indicates that object x has attribute y (table entry corresponding to row x and column y contains $\times$; if $\langle x, y\rangle \notin I$ the table entry contains blank symbol). In terms of FCA, $\langle X, Y, I\rangle$ is called a formal context. For every $A \subseteq X$ and $B \subseteq Y$ denote by $A^{\uparrow}$ a subset of Y and by $B^{\downarrow}$ a subset of X defined as

$$A^{\uparrow} = \{y \in Y \mid \text{for each } x \in A : \langle x, y\rangle \in I\}, \quad B^{\downarrow} = \{x \in X \mid \text{for each } y \in B : \langle x, y\rangle \in I\}.$$

That is, $A^{\uparrow}$ is the set of all attributes from Y shared by all objects from A (and similarly for $B^{\downarrow}$). A formal concept in $\langle X, Y, I\rangle$ is a pair $\langle A, B\rangle$ of $A \subseteq X$ and $B \subseteq Y$ satisfying $A^{\uparrow} = B$ and $B^{\downarrow} = A$. That is, a formal concept consists of a set A (so-called extent) of objects which are covered by the concept and a set B (so-called intent) of attributes which are covered by the concept such that A is the set of all objects sharing all attributes from B and, conversely, B is the collection of all attributes from Y shared by all objects from A. Alternatively, formal concepts can be defined as maximal rectangles of $\langle X, Y, I\rangle$ which are full of $\times$'s: For $A \subseteq X$ and $B \subseteq Y$, $\langle A, B\rangle$ is a formal concept in $\langle X, Y, I\rangle$ iff $A \times B \subseteq I$

and there is no $A' \supset A$ or $B' \supset B$ such that $A' \times B \subseteq I$ or $A \times B' \subseteq I$. Formal concepts represent clusters hidden in object-attribute data.

A set $\mathcal{B}(X, Y, I) = \{\langle A, B\rangle \mid A^{\uparrow} = B, B^{\downarrow} = A\}$ of all formal concepts in $\langle X, Y, I\rangle$ can be equipped with a partial order $\leq$. The partial order models a subconcept–superconcept hierarchy, e.g. *dog* $\leq$ *mammal*, and is defined by

$$\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle \quad \text{iff } A_1 \subseteq A_2 (\text{iff } B_2 \subseteq B_1). \tag{1}$$

Note that $^{\uparrow}$ and $^{\downarrow}$ form a Galois connection (Ganter and Wille 1999) and that $\mathcal{B}(X, Y, I)$ is in fact a set of all fixpoints of $^{\uparrow}$ and $^{\downarrow}$. $\mathcal{B}(X, Y, I)$ equipped with $\leq$ happens to be a complete lattice, called the concept lattice of $\langle X, Y, I\rangle$. The basic structure of concept lattices is described by the so-called basic theorem of concept lattices (Ganter and Wille 1999).

THEOREM 2.1. (1) The set $\mathcal{B}(X, Y, I)$ equipped with $\leq$ forms a complete lattice in which infima and suprema are given by

$$\bigwedge_{j\in J}\langle A_j, B_j\rangle = \left\langle \bigcap_{j\in J} A_j, \left(\bigcup_{j\in J} B_j\right)^{\uparrow\downarrow}\right\rangle, \quad \bigvee_{j\in J}\langle A_j, B_j\rangle = \left\langle \left(\bigcup_{j\in J} A_j\right)^{\downarrow\uparrow}, \bigcap_{j\in J} B_j\right\rangle.$$

(2) Moreover, an arbitrary complete lattice $\mathbf{V} = \langle V, \leq\rangle$ is isomorphic to $\mathcal{B}(X, Y, I)$ iff there are mappings $\gamma : X \to V$, $\mu : Y \to V$ such that

(i) $\gamma(X)$ is $\bigvee$-dense in V, $\mu(Y)$ is $\bigwedge$-dense in V;
(ii) $\gamma(x) \leq \mu(y)$ iff $\langle x, y\rangle \in I$.

Recall that the cover relation on $\mathcal{B}(X, Y, I)$ is defined as follows. A formal concept $\langle A_1, B_1\rangle$ covers a formal concept $\langle A_2, B_2\rangle$ if $\langle A_2, B_2\rangle \leq \langle A_1, B_1\rangle$ and there is no $\langle A_3, B_3\rangle$ distinct from both $\langle A_1, B_1\rangle$ and $\langle A_2, B_2\rangle$ such that $\langle A_2, B_2\rangle \leq \langle A_3, B_3\rangle \leq \langle A_1, B_1\rangle$.

For detailed information on formal concept analysis we refer to Carpineto and Romano (2004) and Ganter and Wille (1999) where the reader can find theoretical foundations, methods and algorithms, and applications in various areas.

3. Decision tree induction based on FCA

In this section, we describe our algorithm for the induction of decision trees. As mentioned above, the algorithm utilises a concept lattice associated to input data, i.e. a partially ordered set of formal concepts in the sense of FCA. In particular, we attempt to consider the concept lattice (without the least formal concept) as a collection of overlapping trees and to select one tree from this collection as the resulting decision tree.

Input data and its transformation. We consider input data with categorical attributes in our paper. To derive a concept lattice from the input data, we need to transform the categorical attributes to binary attributes because, in its basic setting, FCA works with binary attributes. A transformation of input data, which consists in replacing non-binary attributes into binary ones is called conceptual scaling in FCA (Ganter and Wille 1999). Note that in our case, we need not transform the class label attribute, i.e. the attribute determining to which class the record belongs, because we build the concept lattice over the input attributes only. Throughout this section, we use the input data from Table 1 (top) to illustrate the main issues involved. The data table contains sample animals described by attributes *body temperature*, *gives birth*,

Table 1. Input data table (top) and corresponding data table for FCA (bottom).

Name	*body temp.*	*gives birth*	*four-legged*	*hibernates*	*mammal*
cat	warm	yes	yes	no	yes
bat	warm	yes	no	yes	yes
salamander	cold	no	yes	yes	no
eagle	warm	no	no	no	no
guppy	cold	yes	no	no	no

Name	*bt cold*	*bt warm*	*gb no*	*gb yes*	*fl no*	*fl yes*	*hb no*	*hb yes*	*mammal*
cat	0	1	0	1	0	1	1	0	yes
bat	0	1	0	1	1	0	0	1	yes
salamander	1	0	1	0	0	1	0	1	no
eagle	0	1	1	0	1	0	1	0	no
guppy	1	0	0	1	1	0	1	0	no

four-legged, *hibernates*, and *mammal*, with the last attribute being the class label attribute. After an obvious transformation (nominal scaling) of the input attributes, we obtain the data depicted in Table 1 (bottom). A concept lattice which we use in our method is derived from data which we obtain after such transformation.

Next, we describe our algorithm. Step 1 describes how we compute the (part of) concept lattice of the input data. Step 2 describes the selection of a tree from the concept lattice computed in Step 1. Step 3 describes how we build the decision tree from the tree computed in Step 2.

Step 1. In this step, we compute a part of the concept lattice associated to the data corresponding to input attributes. For this purpose, we use the well-known Lindig's algorithm (Lindig 2000), which we modify in two respects. The original Lindig's algorithm, in its top-down version, generates all formal concepts of a concept lattice associated to input data and the cover relation. It starts with the largest formal concept and recursively generates all formal concepts that are covered by largest formal concept, then generates all formal concepts which are covered by those covered by the largest formal concept, and so on until all formal concepts have been computed. Our modification of Lindig's algorithm consists of two steps.

First, we do not generate lower neighbors of formal concept whose extent contains objects that have all the same class label. That is, if c is the class label attribute, we do not generate lower neighbors of formal concepts $\langle A, B\rangle$ such that for every $x_1, x_2 \in A$, the value of c on x_1 equals the value of c on x_2.

Second, contrary to Lindig's algorithm which computes all formal concepts and the cover relation, our modification computes a relation on the set of the computed formal concepts which is in general larger than the cover relation. In the original Lindig's algorithm, procedure NEXTNEIGHBOR generates set *(Next)Neighbors* of $\langle A, B\rangle$ defined by

$$(Next)Neighbors \text{ of } \langle A, B\rangle = \{\langle C, D\rangle \mid D = (B \cup \{y\})^{\downarrow\uparrow},$$

$$y \in Y - B \text{ such that } (B \cup \{z\})^{\downarrow\uparrow} = D \text{ for all } z \in D - B\}.$$

It can be shown that *(Next)Neighbors* of $\langle A, B\rangle$ is just the set of all formal concepts covered by $\langle A, B\rangle$. In our modification, the procedure NEXTNEIGHBOR generates the set

(Our)Neighbors of $\langle A, B\rangle$ defined by

$$(Our)Neighbors \text{ of } \langle A, B\rangle = \{\langle C, D\rangle \mid D = (B \cup \{y\})^{\downarrow\uparrow},\ y \in Y - B\}.$$

Clearly, *(Our)Neighbors* of $\langle A, B\rangle$ is (in general) larger than *(Next)Neighbors* of $\langle A, B\rangle$ because it may contain formal concepts which result by adding a single attribute $y \in Y - B$ but are not covered by $\langle A, B\rangle$.

The reason for our modification is the following. As mentioned above, formal concepts correspond to the nodes of a decision tree in our approach. Let a formal concept $\langle A, B\rangle$ correspond to a node n in a decision tree. Let $y \in Y - B$ be a binary attribute corresponding to value v_y of a categorical attribute a_y. That is, a_y has value v_y for object x in the original input data if and only if y has value 1 for x in the transformed data with binary attributes. If a_y is the splitting attribute for node n, then $\langle (B \cup \{y\})^{\downarrow}, (B \cup \{y\})^{\downarrow\uparrow}\rangle$ is the formal concept corresponding to node n_y which is connected to n via an edge representing the test 'is the value of a_y equal to v_y?' In order to keep the possibility of having nodes n and n_y in the resulting decision tree, we need to generate both $\langle A, B\rangle$ and $\langle (B \cup \{y\})^{\downarrow}, (B \cup \{y\})^{\downarrow\uparrow}\rangle$ even if $\langle (B \cup \{y\})^{\downarrow}, (B \cup \{y\})^{\downarrow\uparrow}\rangle$ is not covered by $\langle A, B\rangle$ in the concept lattice. This is why *(Our)Neighbors* of $\langle A, B\rangle$ is in general different from *(Next)Neighbors* of $\langle A, B\rangle$.

Algorithm 1 contains pseudocode of the modified Lindig's algorithm. NEXTNEIGHBOR is the main procedure. Formal concepts computed by the algorithm are stored in the variable $\mathcal{F}$. Variable $\langle A, B\rangle_*$ stores the lower neighbours of $\langle A, B\rangle$. The procedure DECIDED prevents computing lower neighbours of formal concepts whose objects have the same class label. Procedure NEIGHBORS computes the set *(Our)Neighbors* of $\langle A, B\rangle$. The algorithm also computes for every formal concept $\langle A, B\rangle$ the number $L_{\langle A,B\rangle}$ explained and utilised in Step 2 below. The part of the concept lattice built from the data table in Table 1 (bottom) computed by Algorithm 1 is shown in Figure 2. Note that the new lower neighbour relationship is displayed by dashed lines. The solid lines indicate a tree to be selected from the part of the concept lattice using a procedure described in Step 2.

Step 2. In this step, we select a tree of formal concepts from the part of a concept lattice computed in Step 1. For this purpose, we compute for every formal concept $\langle A, B\rangle$ computed in Step 1 the number $L_{\langle A,B\rangle}$ of all formal concepts which can be reached from $\langle A, B\rangle$ by following certain labelled paths from $\langle A, B\rangle$ downward. As mentioned in Step 1, the numbers $L_{\langle A,B\rangle}$ are computed in Algorithm 1. The paths consist of labelled edges corresponding to the lower neighbour relation described by *(Our)Neighbors* of .... In particular, an edge

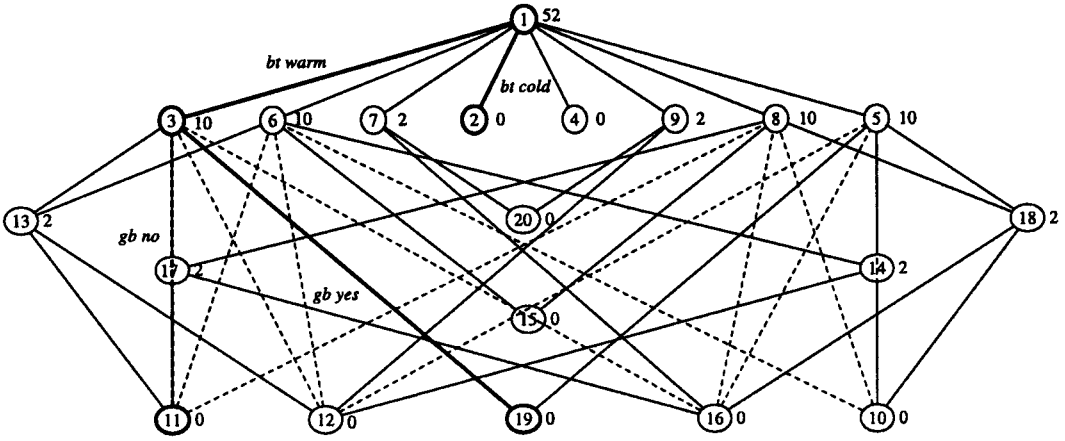


Figure 2. Part of the concept lattice and tree of concepts (solid lines) of data table in Table 1.

with label y goes from $\langle C_1, D_1\rangle$ to $\langle C_2, D_2\rangle$ if $D_2 = (D_1 \cup \{y\})^{\downarrow\uparrow}$, i.e. if $\langle C_2, D_2\rangle$ belongs to *(Our)Neighbors* of $\langle C_1, D_1\rangle$ due to attribute y. Therefore, $L_{\langle A,B\rangle}$ is the number of formal concepts which can be reached from $\langle A, B\rangle$ via the lower neighbour relation in a way that counts a lower neighbour $\langle C_2, D_2\rangle$ of $\langle C_1, D_1\rangle$ multiple times. Namely, $\langle C_2, D_2\rangle$ is counted k times where k is the number of attributes due to which $\langle C_2, D_2\rangle \in$ *(Our)Neighbors* of $\langle C_1, D_1\rangle$, i.e. $k = |\{y; D_2 = (D_1 \cup \{y\})^{\downarrow\uparrow}\}|$. The multiple counting of formal concepts is ensured by the labelling used in $\langle\langle C, D\rangle, y\rangle$ in Algorithm 1.

Furthermore, for every formal concept $\langle A, B\rangle$ we define collections $\mathcal{N}^a_{\langle A,B\rangle}$ of formal concepts that are candidates to become the children of $\langle A, B\rangle$ in the selected tree. a denotes a categorical attribute from the original input data. Note that every binary attribute y_v from the data obtained by transformation from the original input data corresponds to some value v of some categorical attribute a. $\mathcal{N}^a_{\langle A,B\rangle}$ is the collection of lower neighbour concepts of $\langle A, B\rangle$ defined as follows:

(a) for every value v of the categorical attribute a and the corresponding binary attribute y_v, if the formal concept $\langle C, D\rangle$ from *(Our)Neighbors* of $\langle A, B\rangle$ which results by adding attribute y_v belongs to the part of the concept lattice computed in Step 1, then $\langle C, D\rangle$ belongs to $\mathcal{N}^a_{\langle A,B\rangle}$;

(b) if some formal concept $\langle C, D\rangle$ from (a) does not belong to the part of the concept lattice computed in Step 1, $\mathcal{N}^a_{\langle A,B\rangle}$ contains the least formal concept $\langle Y^{\downarrow}, Y\rangle$ (in this case, the least formal concept 'replaces'' $\langle C, D\rangle$ in $\mathcal{N}^a_{\langle A,B\rangle}$), and we put $L_{\langle Y^{\downarrow},Y\rangle} = \infty$.

Algorithm 1 Computing part of concept lattice

```
procedure NEIGHBORS(⟨A, B⟩):
    ℒ := ∅
    for each y ∈ Y \ B:
      C := (B ∪ {y})↓
      D := C↑
      if D ≠ Y:
        add ⟨⟨C, D⟩, y⟩ to ℒ
    return ℒ

procedure DECIDED(⟨A, B⟩):
  if all x ∈ A have the same class label
    return true
  else
    return false

procedure GENERATEFROM(⟨A, B⟩):
  ⟨A, B⟩* := call NEIGHBORS(⟨A, B⟩)
  𝒩 := {⟨C, D⟩ | ⟨⟨C, D⟩, y⟩ ∈ ⟨A, B⟩* and
    ⟨C, D⟩ ∉ ℱ}
  for each ⟨C, D⟩ ∈ 𝒩:
    add ⟨C, D⟩ to ℱ
    L⟨C,D⟩ := 0
  for each ⟨C, D⟩ ∈ 𝒩:
    if not call DECIDED(⟨C, D⟩):
      call GENERATEFROM(⟨C, D⟩)
  for each ⟨⟨C, D⟩, y⟩ ∈ ⟨A, B⟩*:
    add 1 + L⟨C,D⟩ to L⟨A,B⟩

procedure NEXTNEIGHBOR:
  ℱ := ∅
  add ⟨∅↓, ∅↓↑⟩ to ℱ
  L⟨∅↓,∅↓↑⟩ := 0
  if not call DECIDED(⟨∅↓, ∅↓↑⟩):
    call GENERATEFROM(⟨∅↓, ∅↓↑⟩)
  return 𝒫 := ⟨ℱ, {⟨A, B⟩* | ⟨A, B⟩ ∈ ℱ}⟩
```

Next, we select a tree from the part of the concept lattice computed in Step 1 by iteratively going from the largest formal concept to the minimal ones. The selection is based on the numbers $L_{\langle A,B\rangle}$ defined above.

(1) The root node of the tree is the largest formal concept $\langle X, X^{\uparrow}\rangle$.

(2) This step corresponds to selection of the splitting attribute. For every formal concept $\langle A, B\rangle$ in the tree which we construct we select from among all the categorical attributes a categorical attribute a for which

$$\min\left\{L_{\langle C,D\rangle} \mid \langle C, D\rangle \in \mathcal{N}^{a}_{\langle A,B\rangle}\right\}$$

attains the minimum value, i.e. we select a for which $\mathcal{N}^{a}_{\langle A,B\rangle}$ contains a formal concept $\langle C, D\rangle$ with the smallest number $L_{\langle C,D\rangle}$. The idea behind this rule is that a small value of $L_{\langle C,D\rangle}$ indicates, in the optimistic scenario, a small number of decision steps necessary to classify objects from A provided we start with a decision based on a, because there is a short classification path in the decision tree going from the node corresponding to $\langle A, B\rangle$. In case of a tie, i.e. if $L_{\langle C_1,D_1\rangle} = L_{\langle C_2,D_2\rangle}$ for some $a_1 \neq a_2$ with $\langle C_1, D_1\rangle \in \mathcal{N}^{a_1}_{\langle A,B\rangle}$ and $\langle C_2, D_2\rangle \in \mathcal{N}^{a_2}_{\langle A,B\rangle}$, we select a_i for which the extent C_i is the largest. If there is still a tie, we break it arbitrarily. The resulting categorical attribute a is later used as the splitting attribute for the node of the decision tree that corresponds to formal concept $\langle A, B\rangle$.

(3) For every formal concept $\langle A, B\rangle$ in the tree and the categorical attribute a selected for $\langle A, B\rangle$ in (2) we connect $\langle A, B\rangle$ to each formal concept $\langle C, D\rangle$ from $\mathcal{N}^{a}_{\langle A,B\rangle}$ by an edge labelled by a binary attribute y for which $D = (B \cup \{y\})^{\downarrow\uparrow}$.

Algorithm 2 contains a pseudocode of the algorithm that selects a tree from the part of the concept lattice. SELECTTREE is the main procedure. Formal concepts of the selected tree are stored in variable $\mathcal{G}$. Edges between nodes are represented by variables $\langle A, B\rangle_{+}$. The procedure CHILDREN calculates $\mathcal{N}^{a}_{c}$.

To illustrate the previous description, consider the part of the concept lattice presented in Figure 2. Formal concepts of this part are represented by circles, which contain the numbers from 1 to 20 assigned to the formal concepts. For every formal concept $\langle A, B\rangle$, the number $L_{\langle A,B\rangle}$ is attached to the right of the circle representing $\langle A, B\rangle$. Algorithm 2 selects a tree of formal concepts as follows. The root node of the tree is the formal concept No. 1. Formal concepts No. 2 and 3 are then selected as the children of the root since they are the elements of the set $\mathcal{N}^{\text{body temp.}}_{1}$ of lower nodes corresponding to the attribute body temp. which satisfies the conditions described in (2) above. Note that in this case, we could have chosen $\mathcal{N}^{\text{gives birth}}_{1}$ instead of $\mathcal{N}^{\text{body temp.}}_{1}$, since both the formal concept No. 2 from $\mathcal{N}^{\text{body temp.}}_{1}$ and No. 4 from $\mathcal{N}^{\text{gives birth}}_{1}$ have the same minimal number L_2 and L_4 and both contain the same number of objects in their extents. The edges of the selected tree are labelled by binary attributes as described in (3) above. Similarly, the children of the formal concept No. 3 are the formal concepts No. 11 and No. 19. This ends the tree selection because the formal concepts No. 4, No. 11, and No. 19 have no lower neighbours. The resulting tree is depicted in Figure 2 by the solid lines.

Step 3. In this step, the tree obtained in Step 2 is transformed into a decision tree. This step is straightforward. We take the tree obtained in Step 2 and re-label its nodes and edges. An inner node is labelled by the categorical attribute selected in (2) of Step 2 for this node. For example, when constructing the decision tree from the tree selected in Figure 2, the node corresponding to the formal concept No. 3 is labelled by gives birth. An edge going from a node is labelled by the value of a categorical attribute corresponding to the

Algorithm 2 Selection of tree of formal concepts

procedure CHILDREN $(\langle\langle A, B\rangle, y\rangle)$:
  $a :=$ *categorical attribute for* y
  $\mathcal{N}^a_{\langle A,B\rangle} := \emptyset$
  for each *logical attribute* z *for* a:
    search for $\langle\langle C, D\rangle, x\rangle \in \langle A, B\rangle_*$ ***such that*** $x = z$
    if found:
      add $\langle\langle C, D\rangle, z\rangle$ ***to*** $\mathcal{N}^a_{\langle A,B\rangle}$
    else:
      add $\langle\langle Y^{\downarrow}, Y\rangle, z\rangle$ ***to*** $\mathcal{N}^a_{\langle A,B\rangle}$
  return $\mathcal{N}^a_{\langle A,B\rangle}$

procedure SELECTFROM $(\langle A, B\rangle)$:
  $S := \{y \mid \langle\langle C, D\rangle, y\rangle \in \langle A, B\rangle_* \wedge L_{\langle C,D\rangle}$ *is minimal*$\}$
  if $S = \emptyset$:
    return
  if $|S| \geq 1$ ***and all*** $y \in S$ *are for different categorical attribute* a:
    $S := \{y \mid y \in S \wedge \langle\langle C, D\rangle, y\rangle \in \langle A, B\rangle_* \wedge |C|$ *is maximal*$\}$
  select *arbitrary* y ***from*** S
  $\langle A, B\rangle_+ :=$ ***call*** CHILDREN $(\langle\langle A, B\rangle, y\rangle)$
  for each $\langle\langle C, D\rangle, y\rangle \in \langle A, B\rangle_+$:
    add $\langle C, D\rangle$ ***to*** $\mathcal{G}$
    call SELECTFROM $(\langle C, D\rangle)$

procedure SELECTTREE $(\mathcal{P})$:
  $\mathcal{G} := \emptyset$
  add $\langle\emptyset^{\downarrow}, \emptyset^{\downarrow\uparrow}\rangle$ ***to*** $\mathcal{G}$
  $L_{\langle Y^{\downarrow},Y\rangle} := \infty$
  call SELECTFROM $(\langle\emptyset^{\downarrow}, \emptyset^{\downarrow\uparrow}\rangle)$
  return $\langle\mathcal{G}, \{\langle A, B\rangle_+ \mid \langle A, B\rangle \in \mathcal{G}\}\rangle$

binary attribute used as a label of this edge in (3) of Step 3. For example, the edge labelled by *gb no* in Figure 2 is labelled by *no* in the resulting decision tree.

The last problem is the labelling of leaf nodes. Consider a leaf node of a tree obtained in Step 2 corresponding to formal concept $\langle A, B\rangle$. If all objects from A have the same class label c, or if c is the class label of a majority of objects from A, the corresponding node of the decision tree is labelled by c. If a leaf node n of a tree obtained in Step 2 corresponds to the least formal concept $\langle Y^{\downarrow}, Y\rangle$, cf. (b) of Step 2, the corresponding node of the decision tree is labelled by the label which would have been assigned to a leaf node corresponding to the formal concept of the parent node of n.

The resulting decision tree of the input data in Table 1 (top) which result by the transformation of the tree of formal concepts displayed in Figure 2, as described in Step 3, is depicted in Figure 3.

4. Experimental evaluation

In this section, we describe experiments with our algorithm and its comparison to reference algorithms for decision tree induction. Namely, we compared our algorithm with decision tree algorithms ID3 and C4.5 (entropy and information gain based), an instance based learning method (IB1), and a multilayer perceptron (MLP) neural network trained by back propagation (Mitchell (1997)). We implemented our method in the C language. The other algorithms were borrowed and run from Weka[2] (Waikato Environment for Knowledge Analysis; Witten and Frank 2005), a software package that contains implementations of machine learning and data mining algorithms in Java. Default Weka's

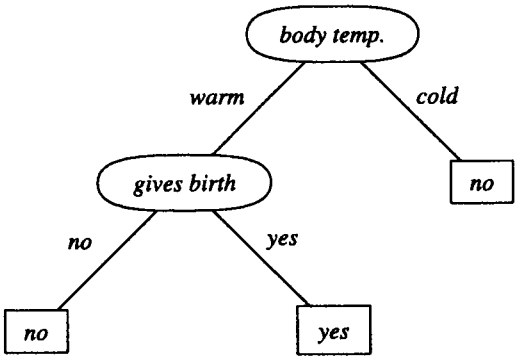


Figure 3. The decision tree of the input data in Table 1.

parameters were used for the other algorithms and decision tree pruning was turned off where available.

The experiments were done on selected public real-world datasets from UCI Machine Learning Repository (Newman *et al.* 1998). The selected datasets are from different areas (medicine, biology, zoology, politics, games). All the datasets contain only categorical attributes with one class label attribute. Due to computational time demands, the datasets were cleared of records containing missing values, of attributes with large numbers of attribute values, and of randomly selected records, to allow for repeated executions of the methods. Let us note in this context that our method is computationally more demanding than the other methods, because of the need to compute a possibly large number of formal concepts. Basic characteristics of the datasets are depicted in Table 2. For datasets with already defined training and testing set (spect and monks-problems) the upper numbers in the table cells relate to the training set and the lower numbers to the testing set. For datasets without training and testing sets, the experiments were done using the 10-fold stratified cross-validation test. The results of averaging 10 execution runs on each dataset with randomly ordered records are depicted in Table 3. The table shows the average percentage rates of correct classifications for both training (upper number in the table cell) and testing (lower number) datasets for each algorithm and

Table 2. Characteristics of datasets used in experiments.

Dataset $\begin{pmatrix}\text{training}\\ \text{testing}\end{pmatrix}$	*No. of attributes*	*No. of records*	*Class distribution*
breast-cancer	6	138	100/38
kr-vs-kp	14	319	168/151
mushroom	10	282	187/95
tic-tac-toe	27	239	156/83
vote	8	116	54/62
zoo	9	101	41/20/5/13/4/8/10
spect	22	80	40/10
		187	15/172
monks-problems-1	17	124	62/62
		432	216/216
monks-problems-2	17	169	105/64
		432	290/142
monks-problems-3	17	122	62/60
		432	204/228

Table 3. Classification accuracy for datasets from Table 2.

Training % *Testing %*	*FCA based*	*ID3*	*C4.5*	*IB1*	*MLP*
breast-cancer	**88.631**	88.630	86.328	84.887	88.550
	79.560	75.945	79.181	71.901	**79.939**
kr-vs-kp	84.395	**84.674**	82.124	79.132	84.426
	74.656	74.503	72.780	68.886	**74.880**
mushroom	96.268	**97.517**	97.163	96.556	97.234
	96.284	96.602	**96.671**	95.214	95.992
tic-tac-toe	98.991	**100.000**	95.165	**100.000**	**100.000**
	85.197	80.519	78.539	83.262	**97.827**
vote	**97.528**	**97.528**	94.883	97.020	95.545
	90.507	89.280	86.500	**91.303**	88.106
zoo	**98.019**	**98.019**	96.039	97.799	97.678
	96.036	95.036	92.690	94.463	95.536
spect	**92.250**	**92.250**	89.250	88.250	91.500
	55.187	54.866	59.679	59.251	**60.481**
monks-problems-1	**100.000**	**100.000**	96.532	**100.000**	99.193
	85.648	79.259	76.828	74.722	**95.833**
monks-problems-2	**100.000**	99.763	92.958	**100.000**	**100.000**
	63.518	59.976	62.314	68.055	**99.814**
monks-problems-3	**100.000**	**100.000**	98.360	**100.000**	99.016
	90.694	91.041	**92.870**	78.634	**92.870**
average	95.608	**95.838**	92.880	94.364	95.314
	81.729	79.703	79.805	78.569	**88.345**

Best performance is in bold.

dataset being compared, plus the average over all datasets. Boldface numbers denote the best results.

We can see that our method, which we call FCA based, outperforms C4.5 and IB1 and gains almost identical results to ID3 and MLP on the training sets of all datasets. On the testing sets this is also the case with the exception of tic-tac-toe, spect, and the monks-problems, on which MLP outperforms all other methods.

5. Conclusion and topics of future research

We presented a novel method of decision tree induction based on formal concept analysis. In this method, the decision tree is constructed using a selection of nodes and edges from a modified line diagram of a concept lattice associated to input data. A heuristic based on a global information provided by the concept lattice, namely, the numbers of particularly defined lower formal concepts, is used to select splitting attributes. An experimental evaluation suggests good performance. Our method outperformed the instance-based method IB1 and is comparable to entropy-based methods ID3 and C4.5 and neural network method MLP.

Future research should focus on the following topics:

- The main novelty in our approach consists in using global information regarding natural clusters in input data that are represented by the concept lattice extracted from the data. Further research, both experimental and theoretical is necessary to better utilise this global information with respect to the design of good decision trees.
- Explore the possibility to compute a smaller number of formal concepts from which the nodes of a decision tree are constructed.

- Explore the problems of overfitting in data and incomplete data, i.e. data having missing values for some attributes in some records. These problems were not considered in this paper.
- Explore the possibility of incremental update of the induced decision trees via incremental methods of constructing concept lattices.
- Explore computational efficiency of our method. Namely, the use of the global information used for selecting the splitting attributes requires to compute a possibly large number of formal concepts.

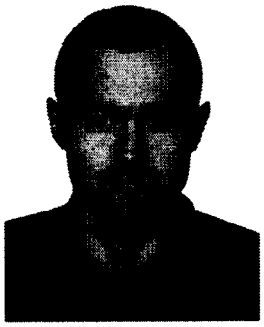
Acknowledgements

Supported by grant No. 1ET101370417 of GA AV ČR, by institutional support, research plan MSM 6198959214, and by the Bilateral Scientific Cooperation Flanders–Czech Republic, Special Research Fund of Ghent University (Project No. 011S01106).

Notes

1. The paper is an extended version of a conference paper presented at CLA 2007, Montpellier, France, 24–26 October 2007.
2. Weka is a free software available at http://www.cs.waikato.ac.nz/ml/weka/

Notes on contributors



Radim Belohlavek is a Professor of Systems Science at Binghamton University-State University of New York. His academic interests are in the areas of uncertainty and information, fuzzy logic and fuzzy sets, data and knowledge engineering, data analysis, formal concept analysis, systems theory. Radim is a Senior Member of IEEE and a member of ACM and AMS. Before joining Binghamton University, he was a Professor and a Head of Department of Computer Science, Palacky University, Olomouc (Czech Republic).

Bernard De Baets leads KERMIT, the research unit *Knowledge-Based Systems*. He serves on the Editorial Boards of various international journals, in particular as co-editor-in-chief of Fuzzy Sets and Systems. Bernard coordinates EUROFUSE, the EURO Working Group on Fuzzy Sets, and is member of the Board of Directors of EUSFLAT, the Technical Committee on Artificial Intelligence and Expert Systems of IASTED, and of the Administrative Board of the Belgian OR Society.

Jan Outrata is an Assistant Professor at the Department of Computer Science, Palacky University in Olomouc, Czech Republic. He has obtained a PhD in Mathematics from Palacky University in 2006. His research interests include fuzzy logic and fuzzy sets, formal concept analysis and relational data analysis, clustering and knowledge engineering. He has authored over 20 papers in conference proceedings and journals including Journal of Computer and System Sciences, International Journal of General Systems, and International Journal of Foundations of Computer Science.

Vilem Vychodil is an Assistant Professor at SUNY Binghamton. He obtained a PhD in Mathematics in 2004 from Palacky University, Olomouc. His professional interests include fuzzy logic, fuzzy relational systems, relational data analysis, uncertainty in data, mathematical logic, and logical foundations of knowledge engineering. He has authored one monograph (Springer) and over 70 papers in conference proceedings and journals including Archives for Mathematical Logic, Mathematical Logic Quarterly, Logic Journal of IGPL, Journal of Experimental and Theoretical Artificial Intelligence, Fuzzy Sets and Systems, Journal of Multiple-Valued Logic and Soft Computing. Vilem Vychodil is a member of the ACM and IEEE.

References

Carpineto, C. and Romano, G., 1996. A lattice conceptual clustering system and its application to browsing retrieval. *Machine learning*, 24, 95–122.

Carpineto, C. and Romano, G., 2004. *Concept data analysis. Theory and applications*. New York: Wiley.

Dunham, M.H., 2003. *Data mining. Introductory and advanced topics*. Upper Saddle River, NJ: Prentice Hall.

Fayyad, U.M., Piatetsky-Shapiro, G. and Smyth, P., 1996. From Data mining to knowledge discovery: an overview. *In*: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, eds. *Advances in knowledge discovery and data mining*. Melno Park, CA: AAAI Press, 3–33.

Fu, H., *et al.*, 2004. A comparative study of FCA-based supervised classification algorithms. *In*: P. Eklund, ed. *ICFCA 2004. Lecture notes in artificial intelligence 2961*, Berlin/Heidelberg: Springer-Verlag, 313–320.

Ganter, B. and Wille, R., 1999. *Formal concept analysis. Mathematical foundations*. Berlin: Springer.

Kuznetsov, S.O., 2004. Machine learning and formal concept analysis. *In*: P. Eklund, ed. *ICFCA 2004. Lecture notes in artificial intelligence 2961*, Berlin/Heidelberg: Springer-Verlag, 287–312.

Lindig, C., 2000. Fast concept analysis. *In*: G. Stumme, ed. *Working with conceptual structures – contributions to ICCS 2000*. Aachen: Shaker Verlag, 152–161.

Mephu Nguifo, E. and Njiwoua, P., 2001. IGLUE: a lattice-based constructive induction system. *Intelligent data analysis*, 5 (1), 73–91.

Mitchell, T.M., 1997. *Machine learning*. McGraw-Hill, New York, 1997.

Murthy, S.K., 1998. Automatic construction of decision trees from data. *Data mining and knowledge discovery*, 2, 345–389.

Newman, D.J., *et al.*, 1998. *UCI repository of machine learning databases*. Irvine, CA: University of California, Department of Information and Computer Science. Available from: http://www.ics.uci.edu/~mlearn/MLRepository.html.

Pasquier, N., *et al.*, 1999. Efficient mining of association rules using closed itemset lattices. *Information systems*, 24 (1), 25–46.

Quinlan, J.R., 1993. *C4.5: programs for machine learning*. San Francisco, CA: Morgan Kaufmann.

Quinlan, J.R., 1996. Learning decision tree classifiers. *ACM computing surveys*, 28 (1), 71–72.

Tan, P.N., Steinbach, M. and Kumar, V., 2006. *Introduction to data mining*. Boston, MA: Addison Wesley.

Witten, I.H. and Frank, E., 2005. *Data mining: practical machine learning tools and techniques*. 2nd ed. San Francisco, CA: Morgan Kaufmann.