Preprocessing input data for machine learning by FCA

Jan Outrata[⋆]

Department of Computer Science, Palacky University, Olomouc, Czech Republic
Tř. 17. listopadu 12, 771 46 Olomouc, Czech Republic
jan.outrata@upol.cz

Abstract. The paper presents an utilization of formal concept analysis in input data preprocessing for machine learning. Two preprocessing methods are presented. The first one consists in extending the set of attributes describing objects in input data table by new attributes and the second one consists in replacing the attributes by new attributes. In both methods the new attributes are defined by certain formal concepts computed from input data table. Selected formal concepts are so-called factor concepts obtained by boolean factor analysis, recently described by FCA. The ML method used to demonstrate the ideas is decision tree induction. The experimental evaluation and comparison of performance of decision trees induced from original and preprocessed input data is performed with standard decision tree induction algorithms ID3 and C4.5 on several benchmark datasets.

1 Introduction

Formal concept analysis (FCA) if ofted proposed to be used as a method for data preprocessing before the data is processed by another data mining or machine learning method [15, 8]. The results produced by these methods indeed depend on the structure of input data. In case of relational data described by objects and their attributes (object-attribute data) the structure of data is defined by the attributes and, more particularly, by dependencies between attributes. Data preprocessing in general then usually consits in transformation of the set of attributes to another set of attributes in order to enable the particular data mining or machine learning method to achieve better results [13, 14].

The paper presents a data preprocessing method utilizing formal concept analysis in a way that certain formal concepts are used to create new attributes describing the original objects. Selected formal concepts are so-called factor concepts obtained by boolean factor analysis, recently described by means of FCA in [1]. First, attributes defined by the concepts are added to the original set of attributes, extending the dimensionality of data. New attributes are supposed to aid the data mining or machine learning method. Second, the original attributes are replaced by the new attributes which usually means the reduction

⋆ Supported by grant no. P202/10/P360 of the Czech Science Foundation

of dimensionality of data since the number of factor concepts is usually smaller than the number of original attributes. Here, a main question arises, whether the reduced number of new attributes can better describe the input objects for the subsequent data mining or machine learning method to produce better results.

There have been several attempts to transform the attribute space in order to improve the results of data mining and machine learning methods. From the variety of these methods we focus on decision tree induction. The most relevant to our paper is are methods known as constructive induction or feature construction [7], where new compound attributes are constructed from original attributes as conjunctions and/or disjunctions of the attributes [11] or arithmetic operations [12] or the new attributes are expressed in m-of-n form [9]. An oblique decision tree [10] is also connected to our approach in a sense that multiple attributes are used in the splitting condition (see section 3.1) instead of single attribute at a time. Typically linear combinations of attributes are looked for, e.g. [2]. Learning the condition is, however, computationally challenging.

Interestingly, we have not found any paper solely on this subject utilizing formal concept analysis. There have been several FCA-based approaches on construction of a whole learning model, commonly called lattice-based or concept-based machine learning approaches, e.g. [6], see [3] for a survey and comparison, but the usage of FCA to transform the attributes and create new attributes to aid another machine learning method is discussed very marginally or not at all. The present paper is thus a move to fill the gap.

The remainder of the paper is organized as follows. The next section contains preliminaries from FCA and introduction to boolean factor analysis, including the necessary tranformations between attribute and factor spaces. The main part of the paper is section 3 demonstrating the above sketched ideas on selected machine mearning method – decision tree induction. An experimental evaluation on selected data mining and machine learning benchmark datasets is provided in section 4. Finally, section 5 draws the conclusion.

2 Preliminaries

2.1 Formal Concept Analysis

In this section we summarize basic notions of FCA. For further information we refer to [4]. An object-attribute data table is 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 $\times$ or 1; otherwise it contains blank symbol or 0). 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

$$\begin{aligned} A^{\uparrow} &= \{y \in Y \,|\, \text{for each } x \in A: \ \langle x, y\rangle \in I\}, \\ B^{\downarrow} &= \{x \in X \,|\, \text{for each } y \in B: \ \langle x, y\rangle \in I\}. \end{aligned}$$

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. 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

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

$\mathcal{B}(X, Y, I)$ 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, see [4].

2.2 Boolean Factor Analysis

Boolean factor analysis is a matrix decomposition method which provides a representation of an object-attribute data matrix by a product of two different matrices, one describing objects by new attributes or factors, and the other describing factors by the original attributes [5]. Stated as the problem, the aim is to decompose an $n \times m$ binary matrix I into a boolean product $A \circ B$ of an $n \times k$ binary matrix A and a $k \times m$ binary matrix B with k as small as possible. Thus, instead of m original attributes, one aims to find k new attributes, called factors.

Recall that a binary (or boolean) matrix is a matrix whose entries are 0 or 1. A boolean matrix product $A \circ B$ of binary matrices A and B is defined by

$$(A \circ B)_{ij} = \bigvee_{l=1}^{k} A_{il} \cdot B_{lj},$$

where $\bigvee$ denotes maximum and $\cdot$ is the usual product. The interperetations of matrices A and B is: $A_{il} = 1$ means that factor l applies to object i and $B_{lj} = 1$ means that attribute j is one of the manifestations of factor l. Then $A \circ B$ says: "object i has attribute j if and only if there is a factor l such that l applies to i and j is one of the manifestations of l". As an example,

$$\begin{pmatrix} 1\,1\,0\,0\,0 \\ 1\,1\,0\,0\,1 \\ 1\,1\,1\,1\,0 \\ 1\,0\,0\,0\,1 \end{pmatrix} = \begin{pmatrix} 1\,0\,0\,1 \\ 1\,0\,1\,0 \\ 1\,1\,0\,0 \\ 0\,0\,1\,0 \end{pmatrix} \circ \begin{pmatrix} 1\,1\,0\,0\,0 \\ 0\,0\,1\,1\,0 \\ 1\,0\,0\,0\,1 \\ 0\,1\,0\,0\,0 \end{pmatrix}.$$

The (solution to the) problem of decomposition binary matrices was recently described by means of formal concept analysis [1]. The description lies in an observation that matrices A and B can be constructed from a set $\mathcal{F}$ of formal concepts of I. In particular, if $\mathcal{B}(X, Y, I)$ is the concept lattice associated to I, with $X = \{1, \ldots, n\}$ and $Y = \{1, \ldots, m\}$, and

$$\mathcal{F} = \{\langle A_1, B_1\rangle, \ldots, \langle A_k, B_k\rangle\} \subseteq \mathcal{B}(X, Y, I),$$

then for the $n \times k$ and $k \times m$ matrices $A_{\mathcal{F}}$ and $B_{\mathcal{F}}$ defined in such a way that the l-th column $(A_{\mathcal{F}})_{\_l}$ of $A_{\mathcal{F}}$ consists of the characteristic vector of A_l and the l-th row $(B_{\mathcal{F}})_{l\_}$ of $B_{\mathcal{F}}$ consists of the characteristic vector of B_l the following universality theorem holds:

Theorem 1. *For every I there is $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ such that $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$.*

Moreover, decompositions using formal concepts as factors are optimal in that they yield the least number of factors possible:

Theorem 2. *Let $I = A \circ B$ for $n \times k$ and $k \times m$ binary matrices A and B. Then there exists a set $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ of formal concepts of I with*

$$|\mathcal{F}| \leq k$$

such that for the $n \times |F|$ and $|F| \times m$ binary matrices $A_{\mathcal{F}}$ and $B_{\mathcal{F}}$ we have

$$I = A_{\mathcal{F}} \circ B_{\mathcal{F}}.$$

Formal concepts $\mathcal{F}$ in the above theorems are called factor concepts. Each factor concept determines a factor. For the constructive proof of the last theorem, examples and further results, we refer to [1].

2.3 Transformations between attribute and factor spaces

For every object i we can consider its representations in the m-dimensional Boolean space $\{0,1\}^m$ of original attributes and in the k-dimensional Boolean space $\{0,1\}^k$ of factors. In the space of attributes, the vector representing object i is the i-th row of the input data matrix I, and in the space of factors, the vector representing i is the i-th row of the matrix A.

Natural transformations between the space of attributes and the space of factors is described by the mappings $g : \{0,1\}^m \to \{0,1\}^k$ and $h : \{0,1\}^k \to \{0,1\}^m$ defined for $P \in \{0,1\}^m$ and $Q \in \{0,1\}^k$ by

$$(g(P))_l = \bigwedge_{j=1}^{m} (B_{lj} \to P_j), \tag{1}$$

$$(h(Q))_j = \bigvee_{l=1}^{k} (Q_l \cdot B_{lj}), \tag{2}$$

for $1 \leq l \leq k$ and $1 \leq j \leq m$. Here, $\to$ denotes the truth function of classical implication ($1 \to 0 = 0$, otherwise 1), $\cdot$ denotes the usual product, and $\bigwedge$ and $\bigvee$ denote minimum and maximum, respectively. (1) says that the l-th component of $g(P) \in \{0,1\}^k$ is 1 if and only if for every attribute j, $P_j = 1$ for all positions j for which $B_{lj} = 1$, i.e. the l-th row of B is included in P. (2) says that the j-th component of $h(Q) \in \{0,1\}^m$ is 1 if and only if there is factor l such that $Q_l = 1$ and $B_{lj} = 1$, i.e. attribute j is a manifestation of at least one factor from Q.

For results showing properties and describing the geometry behind the mappings g and h, see [1].

3 Boolean Factor Analysis and Decision Trees

The machine learning method which we use in this paper to demonstrate the ideas presented in section 1 is decision tree induction.

3.1 Decision Trees

Decision trees represent the most commonly used method in data mining and machine learning [13, 14]. A decision tree can be considered as a tree representation of a function over attributes which takes a finite number of values called class labels. The function is partially defined by a set of vectors (objects) of attribute values and the assigned class label, usually depicted by a table. An example function is depicted in Fig. 1. The goal is to construct a tree that approximates the function with a desired accuracy. This is called a decision tree induction. An induced decision tree is typically used for classification of objects into classes, based on the objects' attribute values. A good decision tree is supposed to classify well both objects described by the input data table as well as "unseen" objects.

Each non-leaf tree node of a decision tree is labeled by an attribute, called a splitting attribute for this node. Such a node represents a test, according to which objects covered by the node are split into v subcollections which correspond to v 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 objects all of which, or the majority of which, have the same class label. An example of a decision tree is depicted in Fig. 4.

Many algorithms for the construction of decision trees were proposed in the literature, see e.g. [14]. A strategy commonly used consists of constructing a decision tree recursively in a top-down fashion, from the root node to the leaves, by successively splitting existing nodes into child nodes based on the splitting attribute. A critical point in this strategy is the selection of splitting attributes in nodes, 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 objects before and after splitting, see [14] for overviews.

Remark 1. In machine learning, and in decision trees at particular, the input data attributes are very often categorical attributes. To utilize FCA with 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 [4]. Note that we need not transform the class attribute, i.e. the attribute determining to which class the object belongs, because we transform the input attributes only in our data preprocessing method.

Throughout this paper, we use input data from Fig. 1 (top) to illustrate the data preprocessing. The data table contains sample animals described by attributes *body temperature*, *gives birth*, *fourlegged*, *hibernates*, and *mammal*, with the last attribute being the class. After an obvious transformation (nominal scaling) of the input attributes, we obtain the data depicted in Fig. 1 (bottom). Boolean factor analysis which we use in our method is applied on data which we obtain after such transformation. For illustration, the decision tree induced from the data is depicted in Fig. 4 (left).

Name	*body temp.*	*gives birth*	*fourlegged*	*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

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

3.2 Extending the collection of attributes

The first approach proposed in our data preprocessing method is the extension of the collection of attributes by new attributes which are created using boolean factor analysis. In praticular, the new attributes are represented by factors obtained from the decomposition of input data table.

Let $I \subseteq X \times Y$ be input data table describing objects $X = \{x_1, \ldots, x_n\}$ by binary attributes $Y = \{y_1, \ldots, y_m\}$. Considering I as a $n \times m$ binary matrix, we find a decomposition $I = A \circ B$ of I into the $n \times k$ matrix A describing objects by factors $F = \{f_1, \ldots, f_k\}$ and $k \times m$ matrix B explaining factors F by attributes. The decomposition of example data table in Fig. 1 is depicted in Fig. 2. The new collection of attributes Y' is then defined to be $Y' = Y \cup F$ and the extended data table $I' \subseteq X \times Y'$ is defined by $I' \cap (X \times Y) = I$ and $I' \cap (X \times F) = A$. Hence the new collection of attributes is the union of original attributes and factors and the extended data table is the apposition of original data table and the table representing the matrix describing objects by factors. Fig. 3 depicts the extended data table.

The key part is the decomposition of the original data table. In the decomposition of binary matrices the aim is to find the decomposition with the number of factors as small as possible. However, since the factors, as new attributes, are used in the process of decision tree induction in our application, we are looking

$$\begin{pmatrix} 0&1&0&1&0&1&1&0 \\ 0&1&0&1&1&0&0&1 \\ 1&0&1&0&0&1&0&1 \\ 0&1&1&0&1&0&1&0 \\ 1&0&0&1&1&0&1&0 \end{pmatrix} = \begin{pmatrix} 0&0&1&0&0&1 \\ 0&0&1&0&1&0 \\ 0&0&0&1&0&0 \\ 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \end{pmatrix} \circ \begin{pmatrix} 0&1&1&0&1&0&1&0 \\ 1&0&0&1&1&0&1&0 \\ 0&1&0&1&0&0&0&0 \\ 1&0&1&0&0&1&0&1 \\ 0&1&0&1&1&0&0&1 \\ 0&1&0&1&0&1&1&0 \end{pmatrix}$$

Fig. 2. Boolean matrix decomposition of input data in Fig. 1

Name	*bc*	*bw*	*gn*	*gy*	*fn*	*fy*	*hn*	*hy*	f_1	f_2	f_3	f_4	f_5	f_6	*mammal*
cat	0	1	0	1	0	1	1	0	0	0	1	0	0	1	yes
bat	0	1	0	1	1	0	0	1	0	0	1	0	1	0	yes
salamander	1	0	1	0	0	1	0	1	0	0	0	1	0	0	no
eagle	0	1	1	0	1	0	1	0	1	0	0	0	0	0	no
guppy	1	0	0	1	1	0	1	0	0	1	0	0	0	0	no

Fig. 3. Extended data table for input data in Fig. 1

also for the factors which have a good "decision ability", i.e. that the factors are good candidates to be splitting attributes. To compute the decomposition we can use the algorithms presented in [1], with modified criterion of optimality of computed factors. In short, the algorithms apply a greedy heuristic approach to search in the space of all formal concepts for the factor concepts which cover the largest area of still uncovered 1s in the input data table. The criterion function of optimality of a factor is thus the "cover ability" of the corresponding factor concept, in particular the number of uncovered 1s in the input data table which are covered by the concept, see [1]. The function value is, for the purposes of this paper, translated to the interval $[0, 1]$ (with the value of 1 meaning the most optimal) by dividing the value by the total number of still uncovered 1s in the data table.

The new criterion function $c\colon 2^{X \times Y} \to [0, 1]$ of optimality of factor concept $\langle A, B\rangle$ is:

$$c(\langle A, B\rangle) = w \cdot c_A(\langle A, B\rangle) + (1 - w) \cdot c_B(\langle A, B\rangle), \tag{3}$$

where $c_A(\langle A, B\rangle) \in [0, 1]$ is the original criterion function of the "cover ability" of factor concept $\langle A, B\rangle$, $c_B(\langle A, B\rangle) \in [0, 1]$ is a criterion function of the "decision ability" of factor concept $\langle A, B\rangle$ and w is a weight of preference among the functions c_A and c_B. Let us focus on the function c_B. The function measures the goodness of the factor, defined by the factor concept, as splitting attribute. As was mentioned in section 3.1, in decision trees, a common approaches to selection of splitting attribute are based on entropy measures. In these approaches, an attribute is the better splitting attribute the lower is the weighted sum of entropies of subcollections of objects after splitting the objects based on the attribute. We thus design the function c_B to be such a measure:

$$c_B(\langle A,B\rangle) = 1 - \left(\frac{|A|}{|X|}\cdot\frac{E(\text{class}|A)}{-\log_2\frac{1}{|V(\text{class}|A)|}} + \frac{|X\setminus A|}{|X|}\cdot\frac{E(\text{class}|X\setminus A)}{-\log_2\frac{1}{|V(\text{class}|X\setminus A)|}}\right), \tag{4}$$

where $V(\text{class}|A)$ is the set of class labels assigned to objects A and $E(\text{class}|A)$ is the entropy of objects A based on the class defined as usual by:

$$E(\text{class}|A) = -\sum_{l\in V(\text{class}|A)} p(l|A)\cdot\log_2 p(l|A), \tag{5}$$

where $p(l|A)$ is the fraction of objects A with assigned class label l. The value of $-\log_2\frac{1}{|V(\text{class}|A)|}$ in (4) is the maximal possible value of entropy of objects A in the case the class labels $V(\text{class}|A)$ are assigned to objects A evenly and the purpose of it is to normalize the value of c_B to the interval $[0,1]$. Note that we put $\frac{0}{0}=0$ in calculations in (4).

Now, having the extended data table $I' \subseteq X\times(Y\cup F)$ containing new attributes F, the decision tree is induced from the extended data table instead of the original data table I. The class labels assigned to objects remain unchanged, see Fig. 3. For ilustration, the decision tree induced from data table in Fig. 3 is depicted in Fig. 4 (right). We can see that the data can be decided by a single attribute, namely, factor f_3 the manifestations of which are original attributes *bt warm* and *gb yes*. Factor f_3, as the combination of the two attributes, is a better splitting attribute in decision tree induction than the two attributes alone.

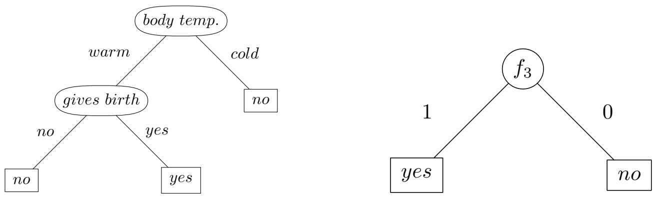

Fig. 4. The decision trees induced from original data table in Tab. 1 (left) and from extended data table in Tab. 3 (right)

The resulted decision tree is used as follows. When classifying an object x described by original attributes Y as a vector $P_x \in \{0,1\}^m$ in the (original) attribute space, we first need to compute the description of the object by new attributes/factors F as a vector $g(P_x)\in\{0,1\}^k$ in the factor space. This is accomplished by (1) using the matrix B explaining factors in terms of original attributes. The object described by concatenation of P_x and $g(P_x)$ is then classified by the decision tree in a usual way.

For instance, an object described by original attributes Y as vector $(10011010)_Y$ is described by factors F as vector $(010000)_F$. The object described by concatenation of these two vectors is classified by class label *no* by the decision tree in Fig. 4 (right).

3.3 Reducing the collection of attributes

The second approach consits in the replacement of original attributes by factors, i.e. discarding the original data table. Hence the new collection of attributes Y' is defined to be $Y' = F$ and the new data table $I' \subseteq X \times Y'$ is put to $I' = A$, where A is the $n \times k$ binary matrix describing objects by factor resulting from the decomposition $I = A \circ B$ of input data table I. Hence the new reduced data table for example data in Fig. 1 is a table depicted in Fig. 3 restricted to attributes $f_1, \ldots, f_6$.

Since the number of factors is usually smaller than the number of attributes, see [1], this transformation usually leads to the reduction of dimensionality of data. However, the transformation of objects from attribute space to the factor space is not an injective mapping. In particular, the mapping g from attribute vectors to factor vectors maps large convex sets of objects to the same points in the factor space, see [1] for details. Namely, for two distinct objects $x_1, x_2 \in X$ with different attributes, i.e. described by different vectors in the space of attributes, $P_{x_1} \neq P_{x_2}$, which have different class labels assigned, $\text{class}(x_1) \neq \text{class}(x_2)$, the representation of both x_1, x_2 by vectors in the factor space is the same, $g(P_{x_1}) = g(P_{x_2})$.

Consider the relation $\ker(g)$ (the kernel relation of g) describing such a situation. The class $[x]_{\ker(g)} \in X/\ker(g)$ for an object $x \in X$ contains objects represented in (original) attribute space which are mapped to the same object x represented in factor space. The class label assigned to each object $x \in X$ in the new data table I' is the majority class label for the class $[x]_{\ker(g)} \in X/\ker(g)$ defined as follows: a class label l is a majority class label for $[x]_{\ker(g)}$ if l is assigned to the most of objects from $[x]_{\ker(g)}$, i.e. if $l = \text{class}(x_1)$ for $x_1 \in [x]_{\ker(g)}$ such that for each $x' \in [x]_{\ker(g)}$ it holds:

$$|\{x_2 \in [x]_{\ker(g)} \mid \text{class}(x_2) = l\}| \geq |\{x_2 \in [x]_{\ker(g)} \mid \text{class}(x_2) = \text{class}(x')\}|.$$

Finally, the decision tree is induced from the transformed data table $I' \subseteq X \times F$, where class labels assigned to each object $x \in X$ is the majority class label for the class $[x]_{\ker(g)} \in X/\ker(g)$. Similarily as in the first approach in section 3.2, when classifying an object x described by original attributes Y as a vector $P_x \in \{0,1\}^m$ in the (original) attribute space, we first compute the description of the object by factors F as a vector $g(P_x) \in \{0,1\}^k$ in the factor space. The object described by $g(P_x)$ is classified by the decision tree. In our example, the decision tree induced from reduced data table (the table in Fig. 3 restricted to attributes $f_1, \ldots, f_6$) is the same as the tree induced from the extended data table, i.e. the tree depicted in Fig. 4 (right).

4 Experimental Evaluation

We performed series of experiments to evaluate our data preprocessing method. The experiments consist in comparing the performance of created machine learning models (e.g. decision trees) induced from original and preprocessed input data. In the comparison we used reference decision tree algorithms ID3 and C4.5 [13] (entropy and information gain based) and also an instance based learning method (IB1). The algorithms were borrowed and run from Weka [1], a software package that contains implementations of machine learning and data mining algorithms in Java. Default Weka's parameters were used for the algorithms.

Table 1. Characteristics of datasets used in experiments

Dataset	*No. of attributes (binary)*	*No. of objects*	*Class distribution*
breast-cancer	9(51)	277	196/81
kr-vs-kp	36(74)	3196	1669/1527
mushroom	21(125)	5644	3488/2156
tic-tac-toe	9(27)	958	626/332
vote	16(32)	232	124/108
zoo	15(30)	101	41/20/5/13/4/8/10

The experiments were done on selected public real-world datasets from UCI Machine Learning Repository. The selected datasets are from different areas (medicine, biology, zoology, politics, games). All the datasets contain only categorical attributes with one class label attribute and the datasets were cleared of objects containing missing values. Basic characteristics of the datasets are depicted in Tab. 1. The numbers of attributes are of original categorical attributes and, in brackets, of binary attributes after nominal scaling (see remark 1). The experiments were done using the 10-fold stratified cross-validation test. The following results are of averaging 10 execution runs on each dataset with randomly ordered records.

Due to the limited scope of the paper we show only the results of data preprocessing by reducing the original attributes to factors and the results for adding the factors to the collection of attributes are postponed to the full version of the paper. The results are depicted in Tab. 2. The tables show ratios of the average percentage rates of correct classifications for preprocessed data and original data, i.e. the values indicate the increase factor of correct classifications for preprocessed data. The values are for both training (upper number in the table cell) and testing (lower number) datasets for each algorithm and dataset being compared, plus the average over all datasets. In the case of top table the

[1] Waikato Environment for Knowledge Analysis, available at `http://www.cs.waikato.ac.nz/ml/weka/`

criterion of optimality of generated factors (3) was set to the original criterion function of the "cover ability" of factor concept, i.e. the original criterion used in the algorithms from [1]. This corresponds to setting $w = 1$ in (3). In the case of bottom table the criterion of optimality of generated factors was changed to the function of the "decision ability" described in section 3.2, i.e. $w = 0$ in (3).

Table 2. Classification accuracy for datasets from Tab. 1, for $w = 1$ (top) and $w = 0$ (bottom table) in (3)

training % *testing %*	*breast-cancer*	*kr-vs-kp*	*mushroom*	*tic-tac-toe*	*vote*	*zoo*	*average*
ID3	1.020	1.000	1.000	1.000	1.000	1.018	1.006
	1.159	0.993	1.000	1.123	0.993	0.962	1.038
C4.5	1.031	0.998	1.000	1.028	0.998	1.006	1.010
	0.989	0.994	1.000	1.092	0.994	0.940	1.002
IB1	1.020	1.000	1.000	1.000	1.000	1.020	1.007
	0.970	1.017	1.000	1.000	1.005	0.965	0.993

training % *testing %*	*breast-cancer*	*kr-vs-kp*	*mushroom*	*tic-tac-toe*	*vote*	*zoo*	*average*
ID3	1.020	1.000	1.000	1.000	1.000	1.018	1.006
	1.153	1.000	1.000	1.157	1.017	0.980	1.051
C4.5	1.047	1.000	1.000	1.033	1.000	1.006	1.014
	1.035	0.998	1.000	1.138	1.007	0.958	1.023
IB1	1.020	1.000	1.000	1.000	1.000	1.020	1.007
	0.951	1.083	1.000	1.213	1.033	0.967	1.041

We can see that while not inducing worse learning model at average on training datasets the methods have better performance at average on testing dataset for input data preprocessed by our methods (with the exception of dataset zoo which has more than two values of class attribute). For instance, ID3 method has better performance by 3.8 % (5.4 % without zoo) for criterion of optimality of generated factors being the original criterion function of the "cover ability" of factor concept, while for criterion of optimality of generated factors being the function of the "decision ability" the performance is better by 5.1 % (6.5 % without zoo). The results for adding the factors to the collection of attributes are very similar, with ± 1 % difference to the results for reducing the original attributes to factors, with the exception of dataset zoo, where the difference was +4 %.

5 Conclusion

We presented two methods of preprocessing input data to machine learning based on formal concept analysis (FCA). In the first method, the collection of attributes describing objects is extended by new attributes while in the second method, the original attributes are replaced by the new attributes. Both methods utilize boolean factor analysis, recently described by FCA, in that the new attributes are defined as factors computed from input data. The number of factors is usually smaller than the number of original attributes. The methods were demonstrated on the induction of decision trees and an experimental evaluation indicates usefullness of such preprocessing of data: the decision trees induced from preprocessed data outperformed decision trees induced from original data for two entropy-based methods ID3 and C4.5.

References

1. Belohlavek R., Vychodil V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. *J. Comput. System Sci* **76**(1)(2010), 3-20.
2. Breiman L., Friedman J. H., Olshen R., Stone C. J.: *Classification and Regression Trees.* Chapman & Hall, NY, 1984.
3. Fu H., Fu H., Njiwoua P., Mephu Nguifo E.: A comparative study of FCA-based supervised classification algorithms. In: Proc. ICFCA 2004, *LNAI* **2961**, 2004, pp. 313–320.
4. Ganter B., Wille R.: *Formal Concept Analysis. Mathematical Foundations.* Springer, Berlin, 1999.
5. Kim K. H.: *Boolean Matrix Theory and Applications.* M. Dekker, 1982.
6. Kuznetsov S. O.: Machine learning and formal concept analysis. In: Proc. ICFCA 2004, *LNAI* **2961**, 2004, pp. 287–312.
7. Michalski R. S.: A theory and methodology of inductive learning. *Artificial Intelligence* **20**(1983), 111–116.
8. Missaoui R., Kwuida L.: What Can Formal Concept Analysis Do for Data Warehouses? In Proc. ICFCA 2009, *LNAI* **5548**, 2009, 58–65.
9. Murphy P. M., Pazzani M. J.: ID2-of-3: constructive induction of M-of-N concepts for discriminators in decision trees. In Proc. of the Eight Int. Workshop on Machine Learning, 1991, 183–187.
10. Murthy S. K., Kasif S., Salzberg S.: A system for induction of oblique decision trees. *J. of Artificial Intelligence Research* **2**(1994), 1–33.
11. Pagallo G., Haussler D.: Boolean feature discovery in empirical learning. *Machine Learning* **5**(1)(1990), 71–100.
12. Piramuthu S., Sikora R. T.: Iterative feature construction for improving inductive learning algorithms. *Expert Systems with Applications* **36**(2, part 2)(2009), 3401–3406.
13. Quinlan J. R.: *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
14. Tan P.-N., Steinbach M., Kumar V.: *Introduction to Data Mining.* Addison Wesley, Boston, MA, 2006.
15. Valtchev P., Missaoui R., Godin R.: Formal concept analysis for knowledge discovery and data mining: The new challenges. In: Proc. ICFCA 2004, *LNAI* **2961**, 2004, pp. 352–371.