Ann Math Artif Intell (2014) 72:3–22
DOI 10.1007/s10472-014-9414-x

Impact of Boolean factorization as preprocessing methods for classification of Boolean data

Radim Belohlavek · Jan Outrata · Martin Trnecka

Published online: 15 June 2014

Abstract We explore a utilization of Boolean matrix factorization for data preprocessing in classification of Boolean data. In our previous work, we demonstrated that preprocessing that consists in replacing the original Boolean attributes by factors, i.e. new Boolean attributes obtained from the original ones by Boolean matrix factorization, can improve classification quality. The aim of this paper is to explore the question of how the various Boolean factorization methods that were proposed in the literature impact the quality of classification. In particular, we compare five factorization methods, present experimental results, and outline issues for future research.

Keywords Matrix decomposition · Factor analysis · Formal concept analysis

Mathematics Subject Classifications (2010) 15A23 · 03C45 · 46L36 · 62H25 · 65F30 · 68T30 · 68W25

1 Problem setting

In Boolean data, objects are described by Boolean (binary, yes-no) attributes, i.e. each object either has or does not have a given attribute. When it comes to classification of Boolean data, one is interested in preprocessing of the input attributes to improve the quality of classification. As with other classification problems, this task may be done many ways. With Boolean input attributes, however, we might want to limit ourselves to preprocessing with a clear semantics. Namely, as is known, see e.g. [2, 7, 11], applying to Boolean

R. Belohlavek · J. Outrata (✉) · M. Trnecka
Data Analysis and Modeling Lab (DAMOL), Department of Computer Science, Palacky University, Olomouc, 17. listopadu 12, 771 46 Olomouc, Czech Republic
e-mail: jan.outrata@upol.cz

R. Belohlavek
e-mail: radim.belohlavek@acm.org

M. Trnecka
e-mail: martin.trnecka@gmail.com

data the methods designed originally for real-valued data distorts the meaning of the data and leads generally to results difficult to interpret. In [12, 13], we proposed a method for preprocessing Boolean data based on the Boolean matrix factorization (BMF) method, i.e. a decomposition method for Boolean matrices, developed in [2]. The method consists in using for classification of the objects new Boolean attributes. The new attributes are actually the factors computed from the original attributes. The factors are (some of the) formal concepts [6] associated to the input data. As is well known, formal concepts have a clear meaning and are easy to interpret [2]. Moreover, there exists a natural transformation of the objects between the space of the original attributes and the space of the factors [2] which is conveniently utilized by the method. As with the other factor analysis methods which create new attributes from the original ones, BMF belongs to the class of feature (new attribute) extraction techniques for reducing dimensionality of data. However, as mentioned above, the traditional well-known methods of feature extraction like principal component analysis (and all its non-linear extensions) or latent semantic analysis, as well as factor analysis, are designed originally for real-valued data and the resulting new attributes may be difficult to interpret. Designed specifically to discrete attributes and thus relevant to us is the multifactor dimensionality reduction technique from gene research [4, 16] but by considering all possible combinations of values of (each two) attributes it is computationally rather intensive. BMF is thus a unique feature extraction method, designed specifically for Boolean data, that transforms the given Boolean attributes to new attributes which are Boolean as well. It has been demonstrated in [12, 13] that preprocessing based on BMF makes it possible to perform classification using a smaller number of input variables (factors instead of original attributes) and yet reasonably improve the quality of classification. The papers provide an experimental analysis of when data with factors as new Boolean attributes enables a better classification than data with the original attributes. In addition to the method from [2], which is utilized in [12, 13], there exist several other BMF methods described in the literature. In the present paper, we therefore look at the question of whether and how the choice of these methods influences the quality of classification. In particular, we focus on five BMF methods and provide an experimental evaluation using a similar scenario as in [12, 13]. Note also that in addition to using the basic BMF method from [2], [12, 13] propose a modification of this basic method and that this modification further improves the quality of classification, in total by 3–5 %. In particular, the modification lies in employing entropy of class labels assigned to objects in the greedy selection of factors to produce factors which are, as new Boolean attributes, more suitable for the subsequent classification. A brief demonstration of this improvement is included in the evaluation in this paper. Nevertheless, in the evaluation of the five BMF method, we use the five methods unmodified in order to compare the methods as they are. The modification of these methods to still improve the quality of classification, as in the case of the above-mentioned modification of [2], is a subject of further research. Such modifications are not trivial matters, see [12, 13], but are a rather promising challenge in view of the results for the method from [2].

In the rest of this paper, we use the following notation. We denote by $X = \{1, \ldots, n\}$ a set of objects which are given along with their input Boolean attributes forming the set $Y = \{1, \ldots, m\}$, and a class attribute c. The input attributes are described by an $n \times m$ Boolean matrix I with entries I_{ij} (entry at row i and column j), i.e. $I_{ij} \in \{0, 1\}$ for every i, j. Alternatively, I may be considered as representing a binary relation between X and Y and, hence, we may speak of a formal context $\langle X, Y, I \rangle$, see [6]. Since there is no danger of confusion, we conveniently switch between the matrix and relational way of looking at things. The class attribute c may be conceived as a mapping $c : X \to C$ assigning to every

object $i \in X$ its class label $c(i)$ in the set C of all class labels (note that C may contain more than two labels).

The preprocessing method along with the five particular methods of Boolean matrix factorization is described in Section 2. Section 3 describes the experiments and provides their results. In Section 4 we conclude the paper and provide some directions for future research.

2 Boolean matrix factorization and its utilization in classification

2.1 General BMF problem

We denote by $\{0, 1\}^{n \times m}$ the set of all $n \times m$ Boolean matrices and by $I_{i\_}$ and $I_{\_j}$ the ith row and jth column, respectively, of matrix I. In BMF, the general aim is to find for a given $I \in \{0, 1\}^{n \times m}$ (and possibly other parameters, see Problem 1 and Problem 2) matrices $A \in \{0, 1\}^{n \times k}$ and $B \in \{0, 1\}^{k \times m}$ for which

$$I \text{ is (approximately) equal to } A \circ B, \tag{1}$$

with $\circ$ being the Boolean matrix product given by

$$(A \circ B)_{ij} = \bigvee_{l=1}^{k} A_{il} \cdot B_{lj}, \tag{2}$$

where $\bigvee$ denotes the maximum and $\cdot$ the ordinary product. Such an exact or approximate decomposition of I into $A \circ B$ corresponds to a discovery of k factors (new Boolean variables) that exactly or approximately explain the data. Namely, factor $l = 1, \ldots, k$, may be represented by $A_{\_l}$ (column l of A) and $B_{l\_}$ (row l of B): $A_{il} = 1$ indicates that factor l applies to object i while B_{lj} indicates that attribute j is a particular manifestation of factor l (think of person P as object, "being fluent in English" as attribute, and "having good education" as factor). The least k for which an exact decomposition $I = A \circ B$ exists is called the Boolean (or Schein) rank of I [2, 8, 11]. Then, according to (2), the factor model reads "object i has attribute j if and only if there exists factor l such that l applies to i and j is a particular manifestation of l". The matrices I, A, and B are usually called the object-attribute matrix, the object-factor (or usage) matrix, and the factor-attribute (or basis vector) matrix [2, 11].

The methods described in the literature are usually designed for two particular problems. Consider the matrix metric [8, 11] (arising from the L_1-norm $||\cdot||$ of matrices, or Hamming weight in case of Boolean matrices) given by

$$E(C, D) = ||C - D|| = \sum_{i=1,j=1}^{m,n} |C_{ij} - D_{ij}|. \tag{3}$$

$E(I, A \circ B)$ may be used to assess how well the product $A \circ B$ approximates the input matrix I.

Problem 1

input: $I \in \{0, 1\}^{n \times m}$, positive integer k
output: $A \in \{0, 1\}^{n \times k}$ and $B \in \{0, 1\}^{k \times m}$ minimizing $||I - A \circ B||$.

This problem is called the *discrete basis problem* (DBP) in [11]. In [2], the following problem is considered:

Problem 2

input: $I \in \{0,1\}^{n\times m}$, positive integer ε
output: $A \in \{0,1\}^{n\times k}$ and $B \in \{0,1\}^{k\times m}$ with k as small as possible such that $||I - A \circ B|| \leq \varepsilon$.

The two problems reflect two important views on BMF, the first one emphasizing the importance of the first k (presumably most important) factors, the second one emphasizing the need to account for (and thus to explain) a prescribed portion of data. Note that the problem of finding an exact decomposition of I with the least number k of factors possible is a particular instance of Problem 2 (put $\varepsilon = 0$). Neither a solution to Problem 1 nor to Problem 2 is unique, which is easy to observe. Namely, according to the geometric view presented in [2], finding a decomposition of I using k factors is equivalent to finding k rectangular areas in I filled with 1s such that every 1 in I is contained in at least one of this areas. Clearly, there are in general several such sets of k rectangular areas. Note also that it follows from the known results that both Problem 1 and Problem 2 are NP-hard optimization problems, see e.g. [2, 11], and hence approximation algorithms are needed to obtain (suboptimal) solutions.

2.2 BMF in preprocessing of Boolean data

The idea may be described as follows. For a given set X of objects, set Y of attributes, Boolean matrix I, and class attribute c, we compute $n \times k$ and $k \times m$ Boolean matrices A and B, respectively, for which $A \circ B$ approximates I reasonably well (either according to the scenario given by Problem 1 or Problem 2). Then, instead of the original instance $\langle X, Y, I, c\rangle$ of the classification problem, we consider a new instance given by $\langle X, F, A, c\rangle$, with $F = \{1, \ldots, k\}$ denoting the factors, i.e. new Boolean attributes. Any classification model developed for $\langle X, F, A, c\rangle$ may then be used to classify the objects described by the original Boolean attributes from Y. Namely, one may utilize natural transformations $g : \{0,1\}^m \to \{0,1\}^k$ and $h : \{0,1\}^k \to \{0,1\}^m$ between the space of the original attributes and the space of factors which are given by

$$(g(P))_l = \bigwedge_{j=1}^{m} (B_{lj} \to P_j) \quad \text{and} \quad (h(Q))_j = \bigvee_{l=1}^{k} (Q_l \cdot B_{lj})$$

for $P \in \{0,1\}^m$ and $Q \in \{0,1\}^k$ ($\bigwedge$ and $\to$ denote minimum and implication). These transformations are described in [2] to which we refer for more information. In particular, given an object represented by $P \in \{0,1\}^m$ (vector of values of the m input attributes), we apply the classification method developed for $\langle X, F, A, c\rangle$ to $g(P)$, i.e. to the object representation in the space of factors. Any classification model $M_F : \{0,1\}^k \to C$ for $\langle X, F, A, c\rangle$ therefore induces a classification model $M_Y : \{0,1\}^m \to C$ by $M_Y(P) = M_F(g(P))$ for any $P \in \{0,1\}^m$.

Note that since the number k of factors of I is usually smaller than the number m of attributes (see [2], which means a reduction of dimensionality of data) and the transforma-

tion of objects from the attribute space to the factor space is not an injective mapping, we need to solve the problem of assigning a class label to objects in $\langle X, F, A, c\rangle$ with equal $g(P)$ representations transformed from objects in $\langle X, Y, I, c\rangle$ with different P representations and different assigned class labels. We adopt the common solution of assigning to such objects in $\langle X, F, A, c\rangle$ the majority class label of class labels assigned to the objects in $\langle X, Y, I, c\rangle$.

2.3 Five methods for Boolean matrix factorization used in our experiments

TILING/GRECON This algorithm is proposed in [7] and, independently but in a considerably more efficient way, in [2]. The idea is the following. The algorithm iteratively selects tiles, which is just a different name for formal concepts, in I until the tiles/formal concepts cover all the 1s in I. The selection is performed in a greedy manner. In every step, we select the tile/formal concept that covers most of the yet uncovered 1s in I. The selected tiles/formal concepts are utilized in a simple way (in this algorithm as well as in GRECOND, GREESS, and GREESSQ): The (characteristic vectors of the) extents and intents of the formal concepts form the columns and rows of A and B. That is, $A_{il} = 1$ iff the extent of the lth formal concept contains object i, $B_{lj} = 1$ iff the intent of the lth formal concept contains attribute j. At the end, we obtain A and B for which $I = A \circ B$. In [2], the set of all tiles/formal concepts is computed in advance. In our paper, we compute this set using the FCbO algorithm [14]. A possible difference in the algorithm from [7] and [2] may result from a possibly different way of resolving ties when more tiles/formal concepts cover the same number of uncovered 1s when a new tile/formal concept is selected in the above-described greedy manner.

GRECOND This algorithm, described in [2] where it is called Algorithm 2, utilizes again formal concepts of I as factors. As with TILING/GRECON, the algorithm is selecting formal concepts of I, one by one, until a decomposition of I into $A \circ B$ is obtained. The algorithm may be stopped after computing the first k concepts or whenever $||I - A \circ B|| \leq \varepsilon$, i.e. the algorithm may be used for solving Problem 1 as well as Problem 2, a feature shared with TILING/GRECON. The formal concepts are selected in a greedy manner to maximize the drop of the error function, in particular, on demand way, whence the name GRE(edy)CON(concepts on)D(emand).

ASSO [11] works as follows. From the input $n \times m$ matrix I, the required number k of factors, and parameters τ, w^+, and w^-, the algorithm computes an $m \times m$ matrix C in which $C_{ij} = 1$ if the confidence of the association rule $\{i\} \Rightarrow \{j\}$ is at least τ. The rows of C are then the candidate rows for matrix B. The actual k rows of B are selected from the rows of C in a greedy manner using parameters w^+ and w^-. During the greedy selection, the k columns of A are selected along with the k rows of B. This way, one obtains from I two matrices A and B such that $A \circ B$ approximates I. Asso is designed for Problem 1. There is no guarantee that ASSO computes an exact factorization of I even for $k = m$, see [3]. In our experiments, we used $\tau = 1$ and $w^+ = w^- = 1$ because such choice guarantees that for $k = m$ all the 1s in I will be covered by the computed factors.

GREESS [3] utilizes the formal concepts of I as factors again. However, it makes use of so-called essential part of I which is a matrix J of the same dimension as I which contains only certain 1s of I and has the property that it is minimal such that the coverage of all 1s

in J guarantees the coverage of all 1s in I. If I is clarified (i.e. does not contain duplicit rows and columns), which is a reasonable assumption since clarification is easy to perform, the essential part is unique. The 1s of the essential matrices correspond to certain minimal intervals in the concept lattice of I. The fact that the coverage by factors of all such 1s guarantees the coverage of all 1s in I appeared for the first time in [5], later from a somewhat different but close point of view in [3]. Denoting by $\mathcal{E}(I)$ the essential part of I, GreEss computes the factors of $\mathcal{E}(I)$ in a greedy manner, but not necessarily to obtain an exact factorization of $\mathcal{E}(I)$. Each such factor (formal concept of $\mathcal{E}(I)$) defines a particular interval in the concept lattice of I. In the search for factors of I, the factors are selected from these intervals. Due to the properties of this strategy, it is sufficient to select at most one formal concept from each such interval. GreEss may be used to solve Problem 1 and 2.

GreEssQ [3] utilizes attribute concepts of the essential part $\mathcal{E}(I)$ of I (see above) and an additional information that distinguishes the entries of I containing 1s. To compute an exact decomposition of I, the algorithm first computes the essential matrix $\mathcal{E}(I)$ and a new $n \times m$ matrix, Q, whose entries are positive integers. For $I_{ij} = 1$, Q_{ij} is the product of the number of rows of I that contain row i (including row i itself) and the number of columns of I that contain column j (including column j); if $I_{ij} = 0$ then $Q_{ij} = 0$. It then selects in a greedy manner as factors of I the attribute concepts of I as follows. For every attribute j with a non-empty column in $\mathcal{E}(I)$, we compute its $^{\downarrow_I\uparrow_I}$-closure $\langle C, D\rangle$ (see [6]). We select such $\langle C, D\rangle$ that maximizes the sum of $\sum_{i\in C, j\in D} Q_{ij}$ and the number of yet uncovered 1s in $\mathcal{E}(I)$. These steps are repeated until the selected factors cover I. The number of factors obtained by GreEssQ for an exact decomposition of I is always less than or equal to the number of the original attributes. Again, GreEssQ may be used to solve both Problem 1 and 2.

As far as the performance of the five algorithms is concerned, we refer the reader to [3] for details. In particular, as far as the quality of decompositions delivered is concerned, for decompositions with a relatively small error (corresponding to coverage of about 60 % of input data and higher), GreEss is the best algorithm, followed by GreConD, Tiling/GreCon, which are of almost the same quality. GreEssQ performs well for exact decompositions and almost exact decompositions but if only a small coverage is required, i.e. a relatively large error is allowed, it fares poorly. The reason is that it utilizes attribute concepts as factors which are rather narrow and many of them are needed to cover a reasonable portion of data. On the other hand, if we are interested in the first couple of factors only, Asso performs very well and is comparable to GreEss or even better for the first two or three factors. Asso, however, is often not able to obtain an exact or almost exact decomposition for the reasons described in [3]. When it comes to computational complexity of the algorithms, GreConD is the fastest, followed by GreEssQ which is about 20 % slower then GreConD, GreEss, which is about 2× slower, and Asso, which is about 3–4× slower than GreConD.

While the above four algorithms have a polynomial time complexity, Tiling/GreCon has an exponential time complexity in the worst case because it browses through the whole concept lattice of the input Boolean matrix, which has an exponential size in the worst case. Correspondingly, Tiling/ GreCon is by far the slowest algorithm. On the other hand, this algorithm is of interest because it performs a greedy selection of formal concepts in the whole concept lattice, implementing thus in full the idea of the basic set cover algorithm.

Table 1 Characteristics of datasets used in experiments

Dataset	No. of attributes (Boolean)	No. of objects	Class distribution
breast-cancer	9 (51)	277	196/81
car	6 (21)	1728	1210/384/69/65
kr-vs-kp	36 (73)	3196	1669/1527
mushroom	22 (124)	5644	3488/2156
solar-flare_2	10 (42)	1066	147/211/239/95/43/331
tic-tac-toe	9 (27)	958	332/626
vote	16 (32)	232	124/108
zoo	15 (30)	101	41/20/5/13/4/8/10

3 Experiments

3.1 Datasets, classification methods, measures of classification performance

We performed a series of experiments to evaluate the impact of the five selected Boolean matrix factorization methods described in Section 2.3 on classification of Boolean data when using factors as new attributes. In the experiments, we evaluate and compare the classification performance of the models created by selected machine learning (ML) algorithms from the data with the original attributes replaced by factors. The factors are computed from input data by the five selected factorization methods. The ML algorithms used in the comparison are: the decision tree algorithms ID3 and C4.5 (entropy and information gain based), the instance-based learning method NN (Nearest Neighbor) and Naive Bayes learning (NB) [10, 15]. 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.

The experiments were done on selected public real-world datasets from the UCI Machine Learning Repository [1]. The selected datasets are from various areas (medicine, industry, games, biology, astronomy, politics, zoology). All the datasets contain only categorical attributes with one class attribute. The datasets were cleared of objects containing missing attribute values (we consider this approach, called listwise deletion and usually the most preferable one, the most reasonable for Boolean matrix factorization). Basic characteristics of the datasets are depicted in Table 1. Note that "9 (51)" means 9 original categorical and 51 Boolean attributes obtained from the categorical attributes by plain nominal scaling. The classification performance is evaluated using the 10-fold stratified cross-validation test [9] and the results below are based on averaging 10 execution runs on each dataset with randomly ordered objects.

To measure classification performance we use classification accuracy and weighted F-measure (or F-score). Recall that classification accuracy (of classification model M_Y on $\langle X, Y, I, c\rangle$) is the ratio of the number of correctly classified objects (for all class labels) to the number of all objects, i.e.

$$ca(M_Y, \langle X, Y, I, c\rangle) = \sum_{l \in C} \frac{tp_l}{|X|},$$

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

where $tp_l = |i \in X\,;\ M_Y(P_i) = c(i) = l|$ (true positive classifications) and $P_i \in \{0,1\}^m$ is the representation of object i by the vector of values of the m input attributes; The F-measure (or, more precisely, F1-measure) for class label l is defined by

$$fm_l(M_Y, \langle X, Y, I, c\rangle) = 2 \cdot \frac{tpr_l \cdot ppv_l}{tpr_l + ppv_l} = \frac{2 \cdot tp_l}{2 \cdot tp_l + fn_l + fp_l},$$

where $tpr_l = \frac{tp_l}{tp_l + fp_l}$ is the ratio of the number of correctly classified objects having l to the number of all objects classified with l (so-called true positive rate in classification), $ppv_l = \frac{tp_l}{tp_l + fn_l}$ is the ratio of the number of correctly classified objects having l to the number of all objects having l (so-called positive predictive value), and $fp_l = |i \in X\,;\ M_Y(P_i) = l \neq c(i)|$ is the number of false positive classifications (type I classification errors) and $fn_l = |i \in X\,;\ M_Y(P_i) \neq c(i) = l|$ is the number of false negative classifications (type II classification errors). Hence, the F-measure for a class label l is the harmonic mean of tpr_l and ppv_l. A weighted F-measure is then the average of F-measures for each class label weighted by the class label rate among objects, i.e.

$$\overline{fm}(M_Y, \langle X, Y, I, c\rangle) = \sum_{l \in C} \frac{|i \in X\,;\ c(i) = l|}{|X|} \cdot fm_l(M_Y, \langle X, Y, I, c\rangle).$$

3.2 Results

The results of the experiments are depicted in Fig. 1, 2, 3, 4, 5, 6, 7, and 8. Each figure contains two columns of four graphs for the four ML algorithms used and one additional graph. The graphs in the two columns show the classification accuracy (left column) and weighted F-measure (right column) on the preprocessed data, i.e. the data $\langle X, F, A, c\rangle$ described by factors, cf. Sections 2.2 and 2.3 for each of the five Boolean matrix factorization methods. The measure rates of TILING/GRECON, GRECOND, ASSO, GREESS and GREESSQ are depicted by the dots-and-dashed, dot-and-dashed, dotted, short-dashed and dashed lines, respectively (note that the measure rates for the kr-vs-kp dataset and TILING/GRECON algorithm are missing due to the number of formal concepts nearly 1 billion which the algorithm needs to compute all). Measure rate for the original data $\langle X, Y, I, c\rangle$ is depicted in each graph by a constant solid line. The x-axis in all graphs corresponds to the factor decompositions obtained by the algorithms and, in particular, measures the quantity

$$\frac{E(I, (A \circ B))}{|\{\langle i, j\rangle\,;\ I_{ij} = 1\}|},$$

see (3), i.e. a kind of relative error w.r.t. 1s of the input matrix I. The values on the x-axis range from 0.9 (corresponding to a factorization with a small number of factors with high $E(I, (A \circ B))$) to 0 (corresponding to the number of factors which decompose I exactly, i.e. $I = A \circ B$). Note also that the measure rates for the ASSO algorithm often do not go to the error rate 0 due to the fact the algorithm commits errors of overcovering I (factors can cover zeroes in I). Then the algorithm may not achieve the prescribed error, i.e. stops at the minimum error it can achieve. The additional graph below the two columns shows the number of factors of the preprocessed data in dependency on the same relative error. The graph is useful in combination with the classification quality graphs in showing the number of factors sufficient for a given quality of classification, in particular the classification on the preprocessed data better than on the original data.

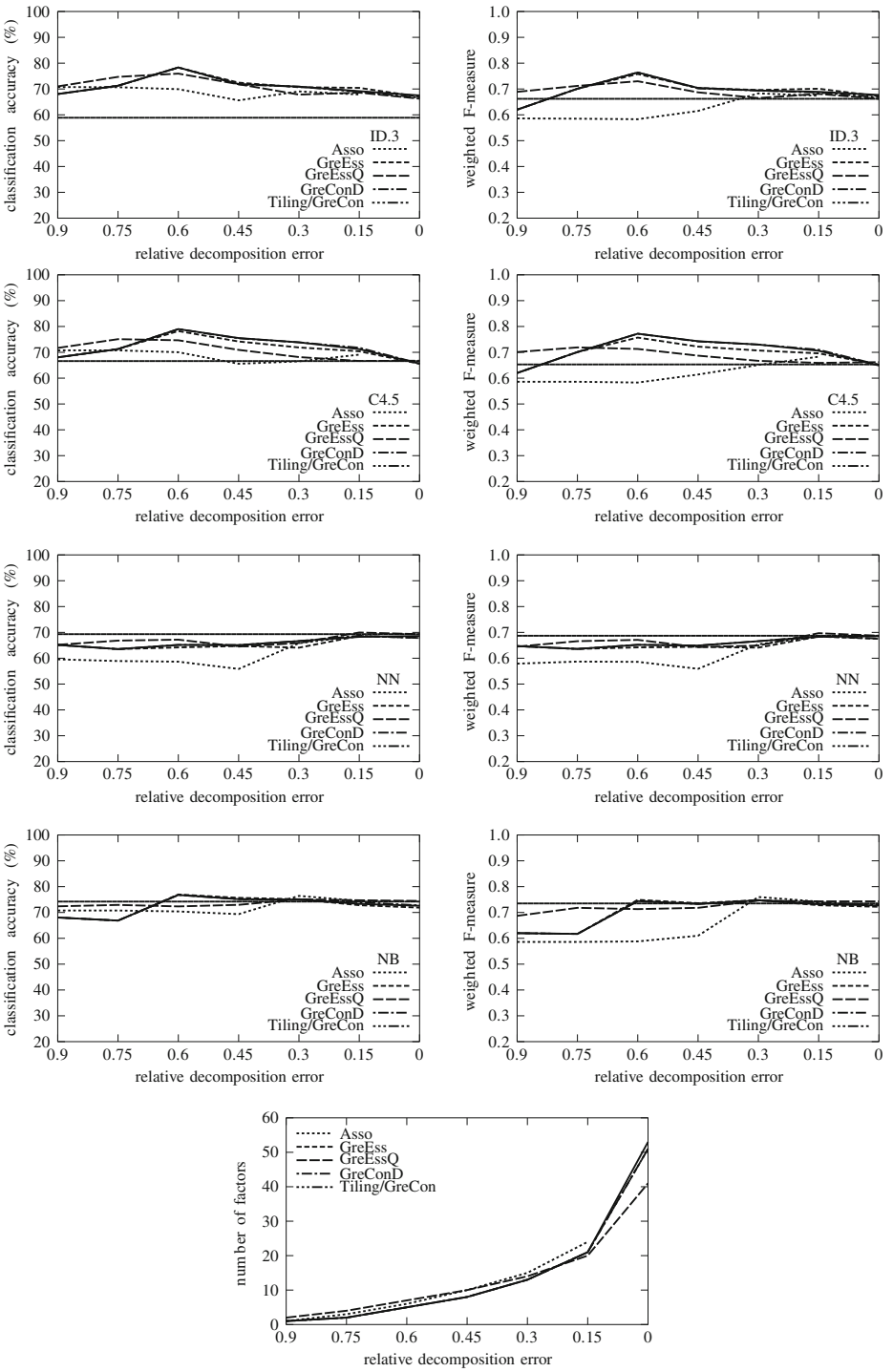

Fig. 1 *Breast-cancer dataset*: classification accuracy (*left column*), weighted F-measure (*right column*) and number of factors (*bottom figure*)

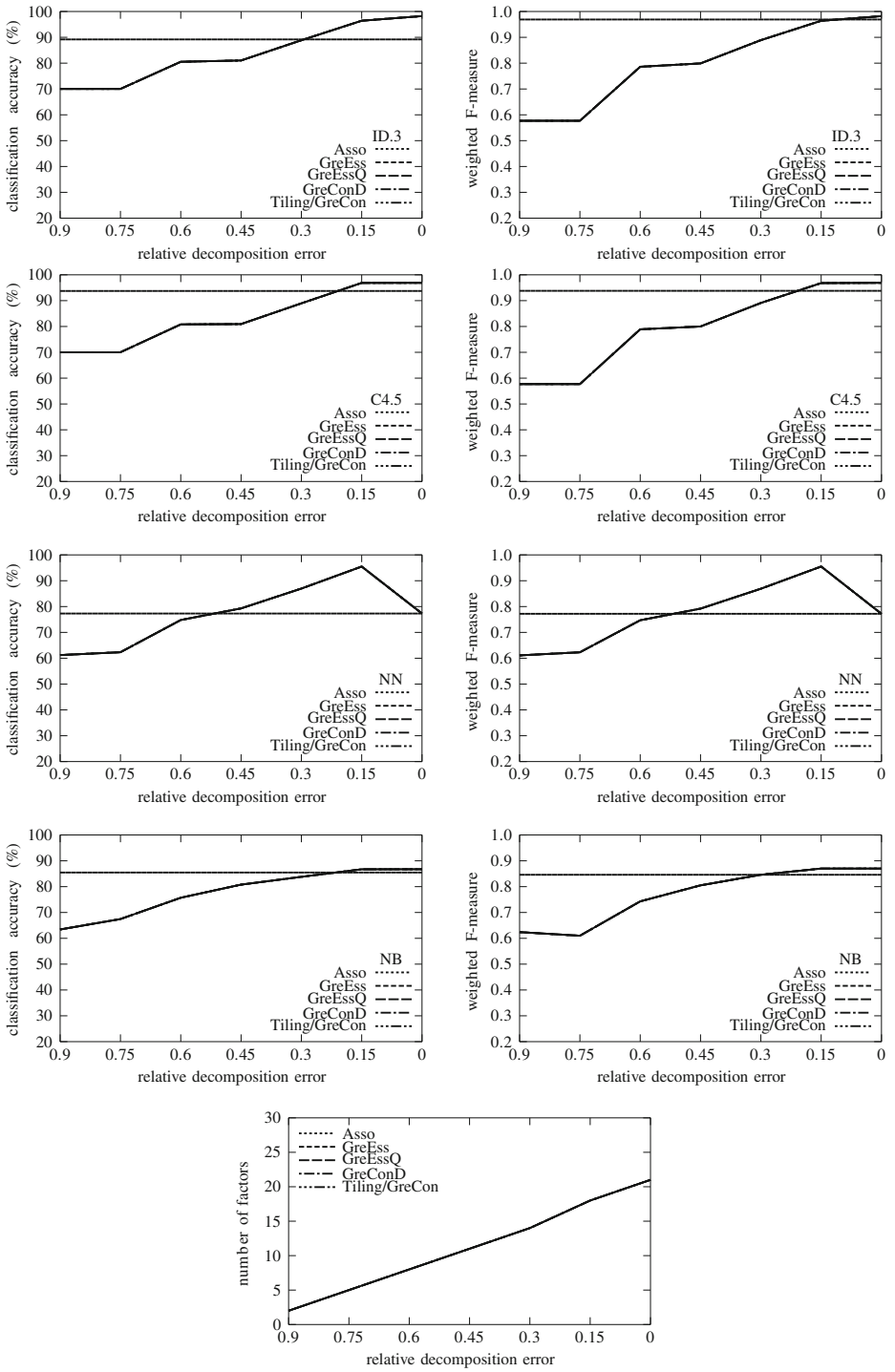

Fig. 2 *Car dataset*: classification accuracy (*left column*), weighted F-measure (*right column*) and number of factors (*bottom figure*); all BMF algorithms have exactly the same performance

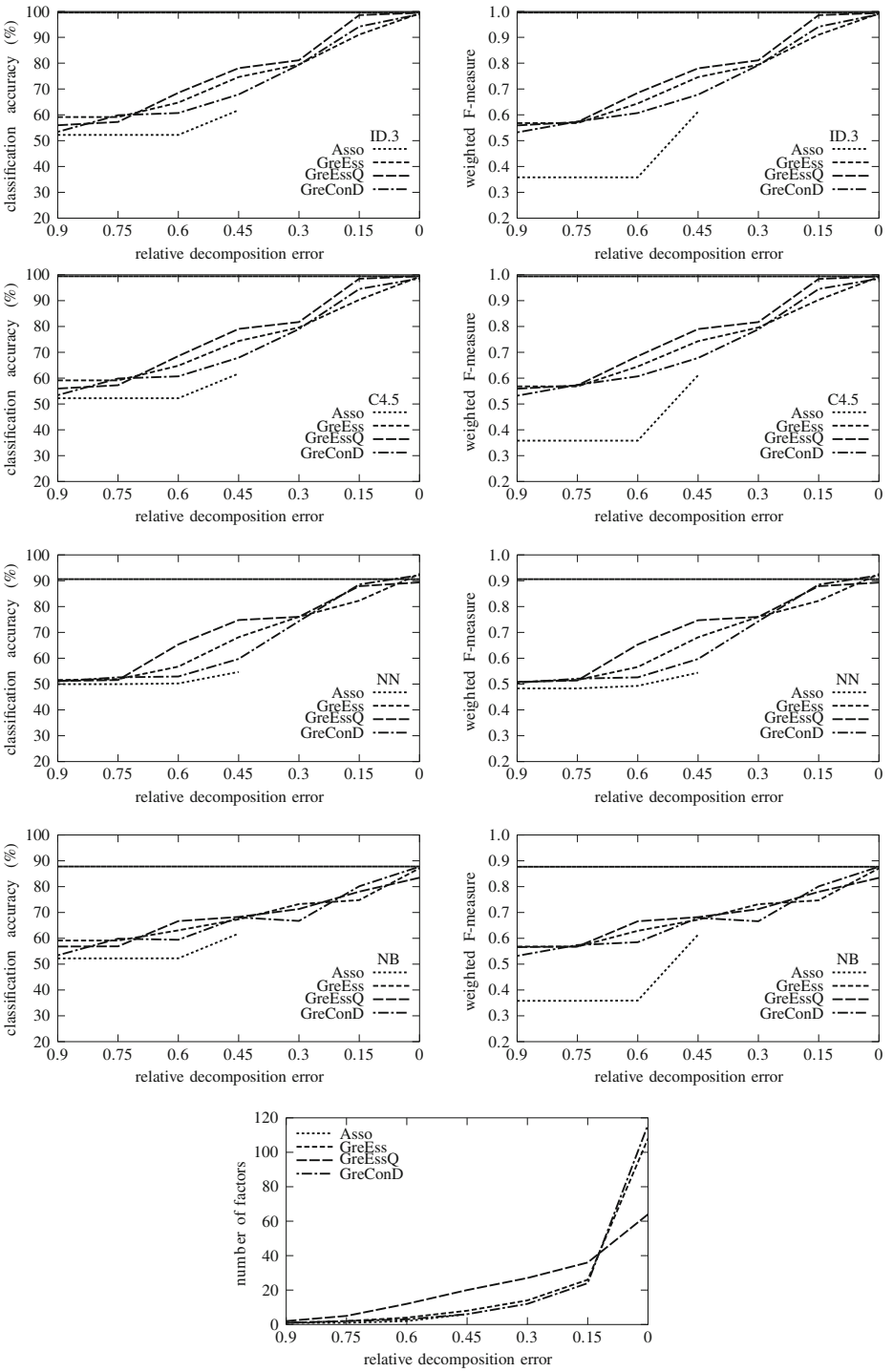

Fig. 3 *Kr-vs-kp dataset*: classification accuracy (*left column*), weighted F-measure (*right column*) and number of factors (*bottom figure*)

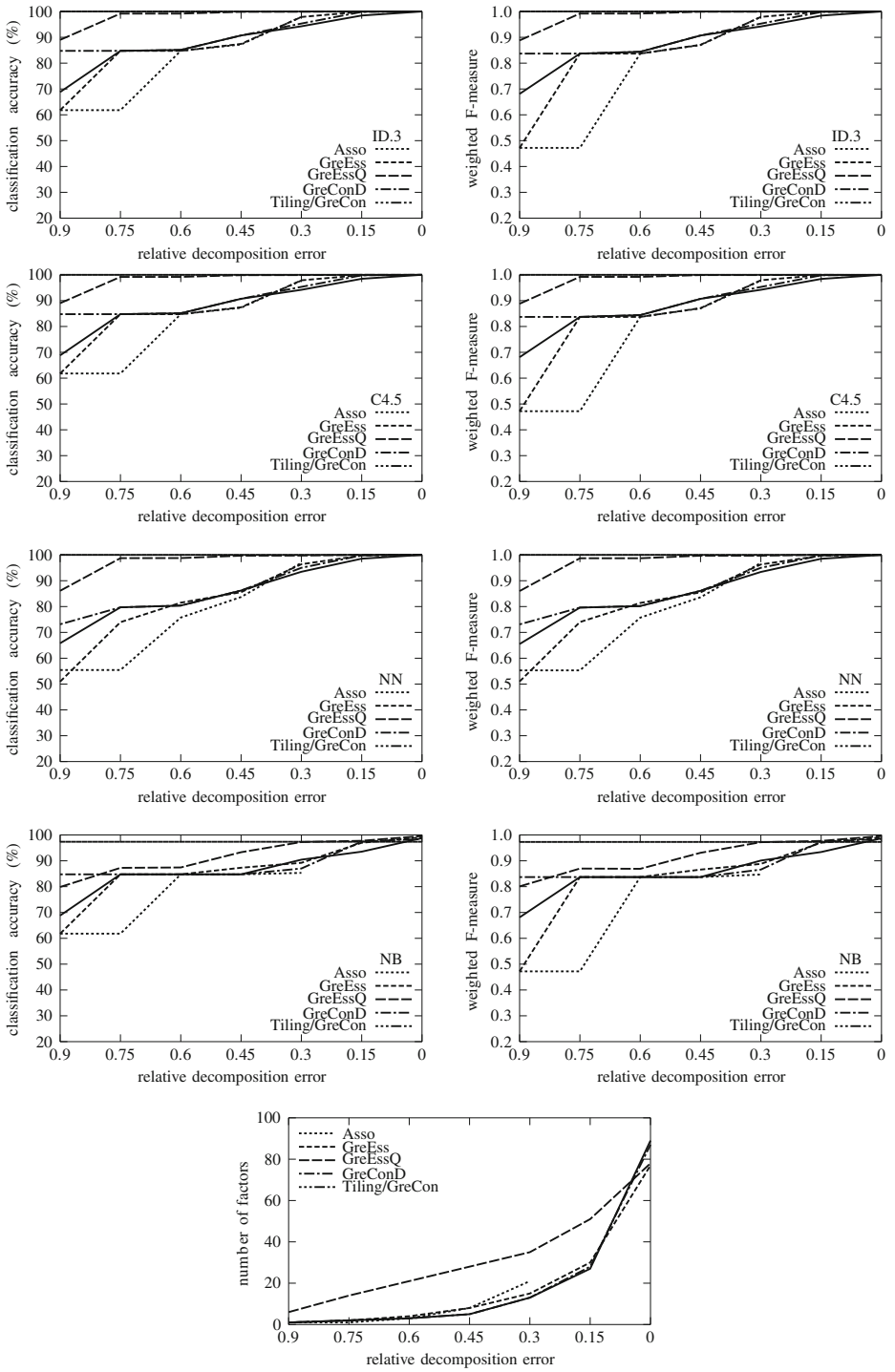

Fig. 4 *Mushroom dataset*: classification accuracy (*left column*), weighted F-measure (*right column*) and number of factors (*bottom figure*)

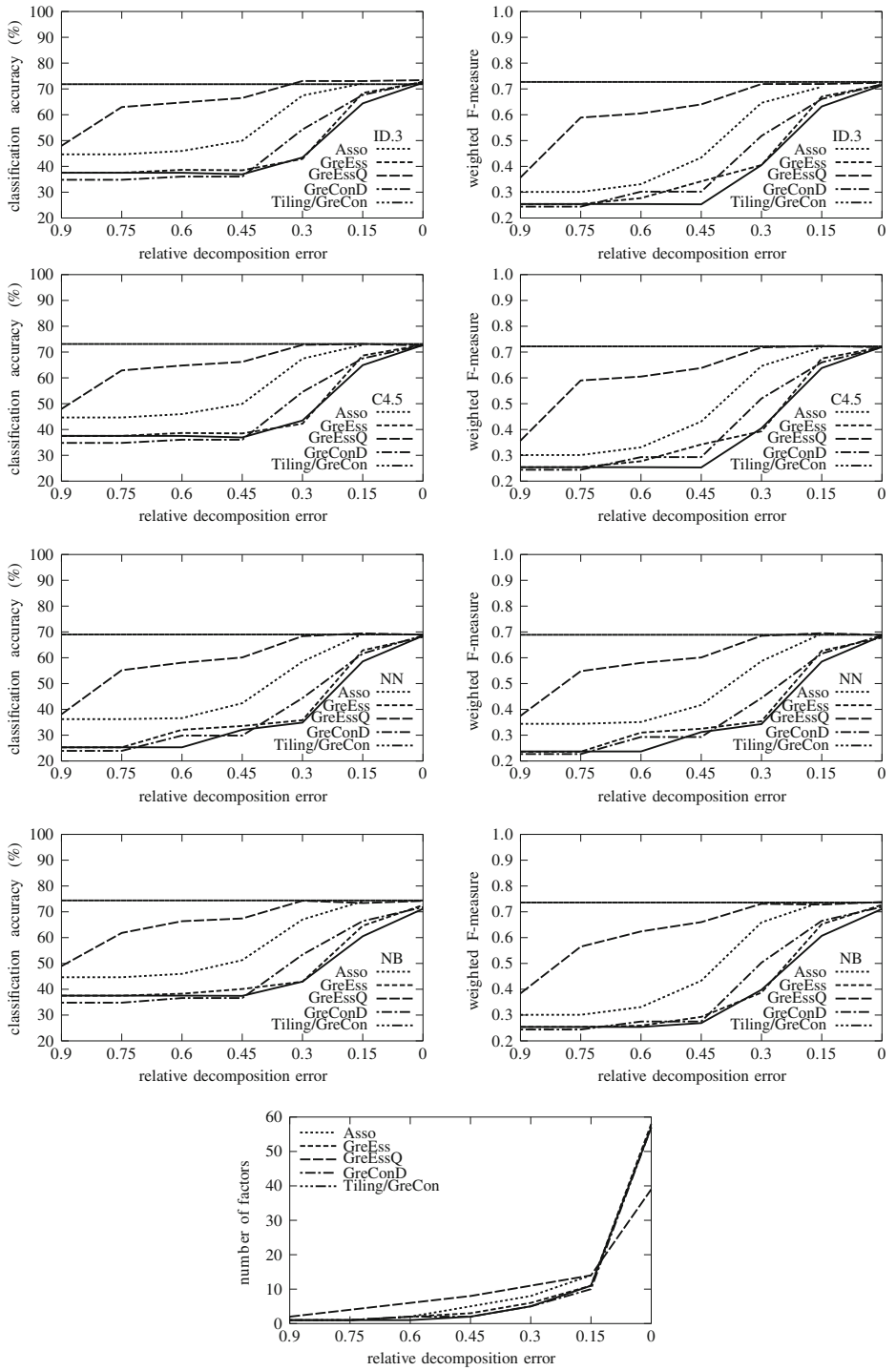

Fig. 5 *Solar-flare_2 dataset*: classification accuracy (*left column*), weighted F-measure (*right column*) and number of factors (*bottom figure*)

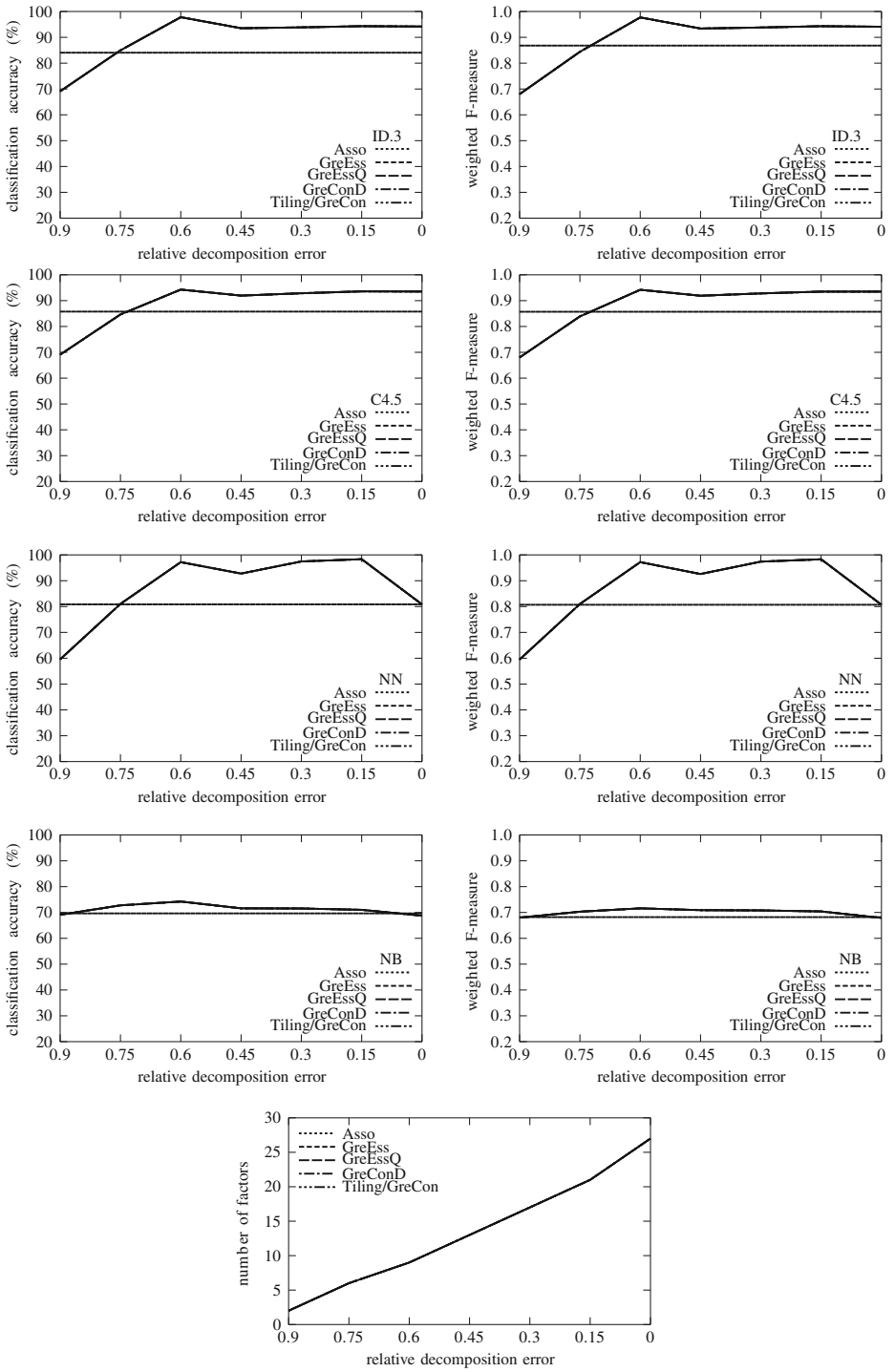

Fig. 6 *Tic-tac-toe dataset*: classification accuracy (*left column*), weighted F-measure (*right column*) and number of factors (*bottom figure*); all BMF algorithms have exactly the same performance

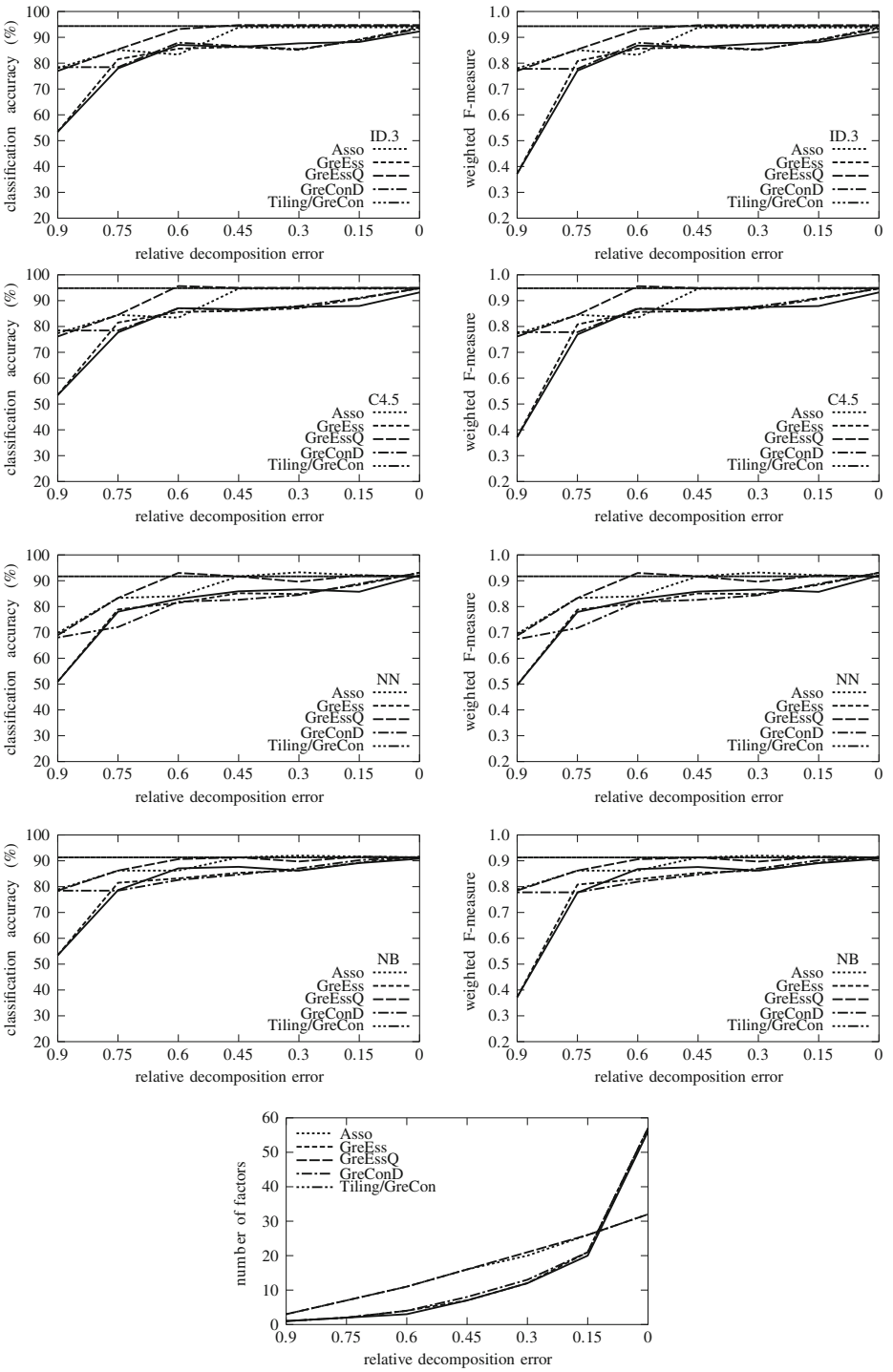

Fig. 7 *Vote dataset*: classification accuracy (*left column*), weighted F-measure (*right column*) and number of factors (*bottom figure*)

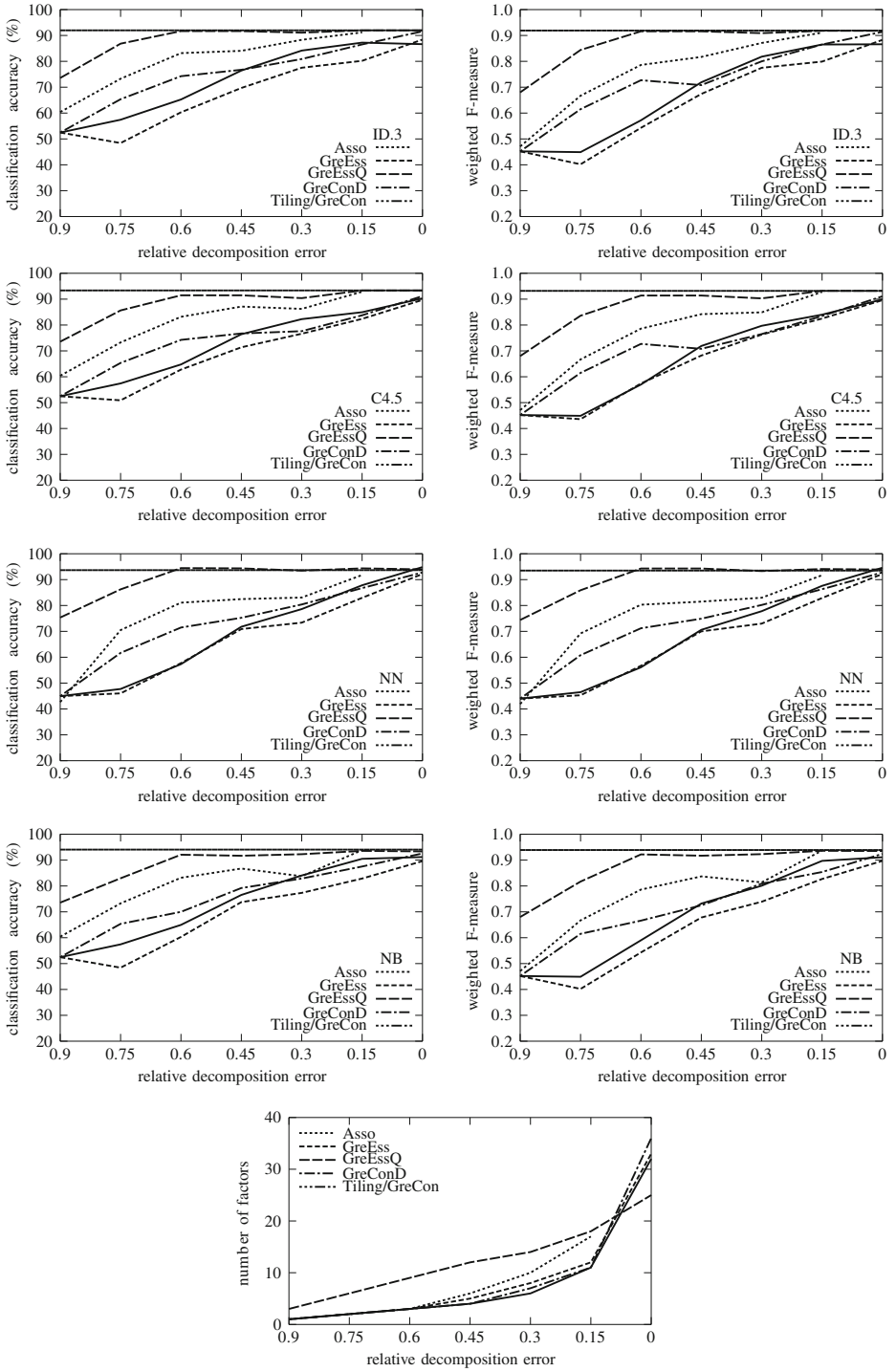

Fig. 8 *Zoo dataset*: classification accuracy (*left column*), weighted F-measure (*right column*) and number of factors (*bottom figure*)

We can clearly see from the graphs that for almost all datasets the best results (in terms of both classification accuracy and weighted F-measure) are obtained for preprocessed data, for all ML algorithms used, by the GreEssQ algorithm, outperforming all other algorithms. The reason seems to be that this algorithm is designed to utilize as factors the so-called attribute concepts (formal concepts generated by a single attribute) and to cover by factors primarily the areas of 1s in input matrix I which are covered by a small number of factors, i.e. factors that may be regarded, in a sense, quasi-mandatory [2, 3] (recall from [2] that a mandatory factor of a dataset is one that needs to be present in every exact decomposition of the dataset). This approach tends to result in a non-redundant set of factors. Such factors appear to be good new attributes for classification.

Poor classification performance of Asso algorithm on some of the datasets (breast-cancer, kr-vs-kp, mushroom), particularly for a relatively large decomposition error, i.e. a small number of factors, is likely due to a different nature of factors used by the algorithm. Namely, the factors used by Asso need not be formal concepts, i.e. need not be rectangles full of 1s. Instead, the rectangles corresponding to the factors computed by Asso may contain 0s. Such factors used as new Boolean attributes tend to result in worse classification performance.

GreConD and GreEss algorithms have quite similar performance. This is not surprising due to a similar strategy of searching for factors. However, for relatively higher number of factors, GreEss has worse performance. Tiling/GreCon algorithm performs similarly to GreConD, on some datasets little bit worse, which is not surprising in view of the fact that, as shown in [2], Tiling/GreCon and GreConD deliver factorizations of similar quality (in terms of the number and size of factors).

Note that for most of the combinations of ML algorithm and dataset, with the exception of the breast-cancer, car and tic-tac-toe datasets, the classification performance is better for the original data than for the data preprocessed by Boolean matrix factorization (also with exact or nearly exact matrix decomposition, i.e. the relative decomposition error equal to 0 or close to 0). However, as we can see from the graphs, sometimes the preprocessed data lead to a better classification accuracy (and the weighted F-measure as well) than the original data and this is true even for decomposition with a few factors covering less than 100 % of input data. This can be seen for instance for the kr-vs-kp, solar-flare_2, vote and zoo datasets and Nearest Neighbor ML algorithm or the mushroom dataset and Naive Bayes ML algorithm. See [12, 13] for indications of when, i.e. for which datasets and ML algorithms, the data with original attributes replaced by factors (computed by GreConD algorithm) covering 100 % of input data leads to a better classification accuracy compared to the original data. Furthermore, the next section discusses a significant improvement of the classification quality with the preprocessed data.

The results for the breast-cancer, car and tic-tac-toe datasets are particularly interesting. First, for the car and tic-tac-toe datasets, all the Boolean matrix factorization algorithms have exactly the same performance, which is due to the very natural factors contained in the datasets successfully found by all the BMF algorithms. And second, more importantly, the preprocessed data, for all ML algorithms, lead to a better classification accuracy than the original data, with a few factors covering less than 100 % of input data. See, in particular, the graphs for ID.3 and Nearest Neighbor ML algorithms. For the breast-cancer dataset, this is true for ID.3 and C4.5, i.e. the decision tree algorithms. Furthermore, the number of factors leading to the best average percentage rates of correct classifications is such that the factors cover just 40 % (which corresponds to 0.6 on the x-axis) of input data. This indicates either many superfluous attributes or large noise in the input data that is overcome by using the factors.

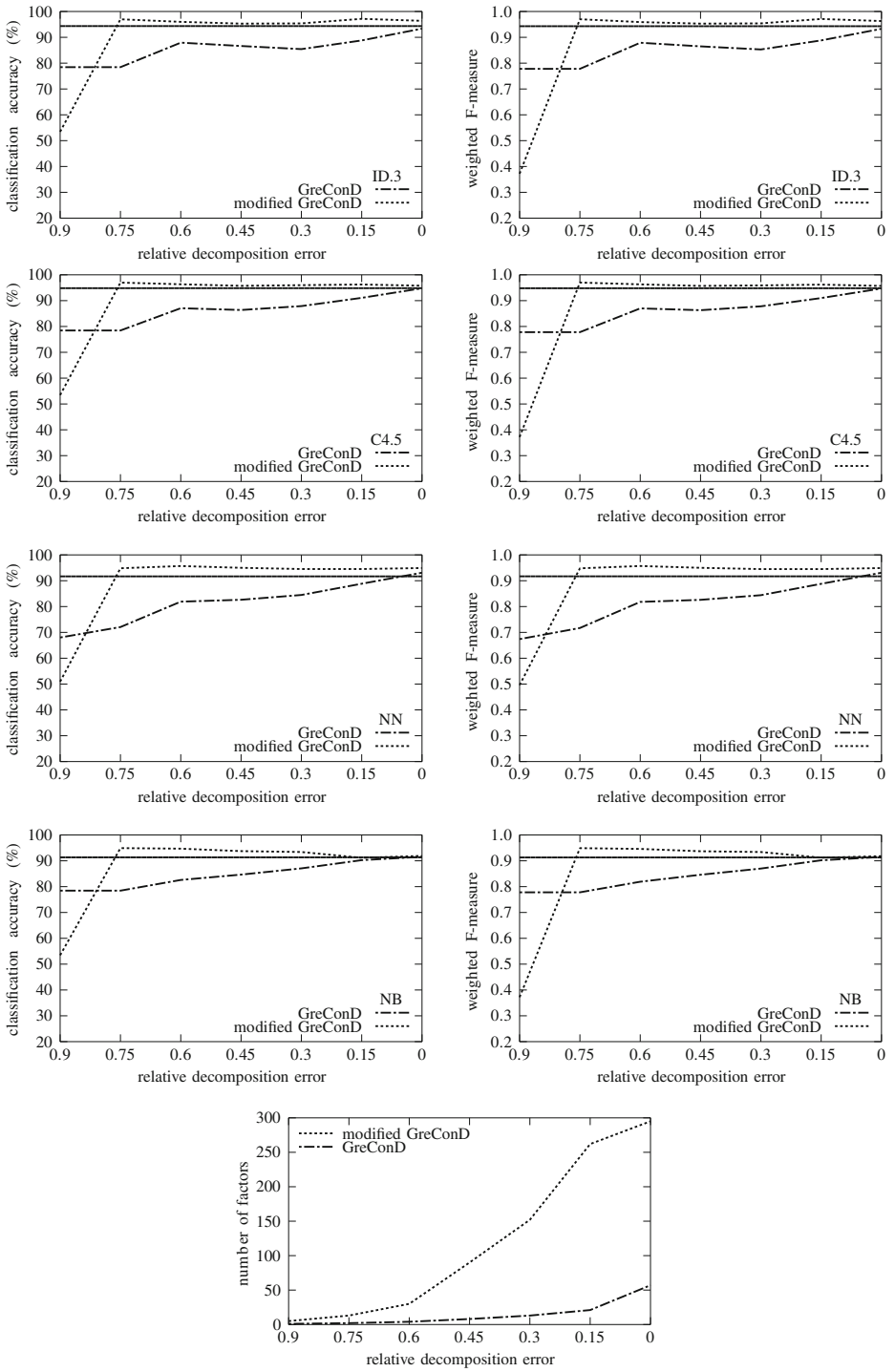

Fig. 9 *Modified* GRECOND *on vote dataset*: classification accuracy (*left column*), weighted F-measure (*right column*) and number of factors (*bottom figure*)

3.3 Further improvement of classification quality

In this section we briefly demonstrate further improvement of classification quality when using for data preprocessing a modified version of GRECOND algorithm from [2] proposed in [12, 13] (see the introduction). Recall that the modification consists in employing the entropy of class labels assigned to objects in the greedy selection of factors. Such factors are, as new Boolean attributes, more preferable to the subsequent classification which gains higher quality. We refer to [12, 13] for details on the modification and entropy employment.

The experimental setting remains the same as for the experiments above, we evaluate the impact of the modified GRECOND algorithm on classification of Boolean data and compare it to the unmodified GRECOND. The results for the vote dataset are depicted in Fig. 9. The figure has the same content as Fig. 1, 2, 3, 4, 5, 6, 7, and 8, only the Boolean matrix factorization methods are now the modified and unmodified GRECOND. The classification measure and the number of factors rates of the modified GRECOND are depicted by the dashed line while the rates of the unmodified GRECOND remain depicted by the dot-and-dashed line.

First, we can see in the figure that the data preprocessed by the modified GRECOND with factors selected based on entropy is classified significantly better than the data preprocessed by the unmodified GRECOND. Second, the data preprocessed by the modified GRECOND lead to a better classification than the original data, for all ML algorithms used, and this happens already from a few factors covering just 25 % (0.75 on the x-axis) of input data. At that point, the number of the factors is considerably lower than the number of factors computed by the unmodified GRECOND that cover 100 % of the input data and even lower (below half for the particular dataset) than the number of original Boolean attributes. Similar results can be seen for several other datasets used in the previous experiments. It is likely that starting from a certain (small) relative number of factors selected based on entropy of class labels assigned to objects the factors appear to be very good new attributes for classification.

4 Conclusions

We presented an experimental study which shows that when Boolean matrix factorization is used as a preprocessing technique in Boolean data classification in the scenario proposed in [12, 13], the particular factorization algorithms impact in a significant way the accuracy of classification. For this purpose, we compared five such algorithms from the literature. In addition to demonstrating further the usefulness of Boolean matrix factorization for classification of Boolean data, the paper emphasizes Boolean factorization as a data dimensionality reduction technique that may be utilized in a similar way as the matrix-decomposition-based methods designed for real-valued data.

In the future research, we intend to further investigate, possibly by theoretical analysis, the role of Boolean factorization on classification accuracy in the presented scenario. The aim of such research is to help us understand the experimental results reported in this paper to help design new Boolean factorization algorithms which perform well in data preprocessing for classification. In particular, we intend to utilize entropy (or other measure of the distribution) of class labels among objects to which a factor applies and modify the present algorithms by employment of such entropy to improve the quality of classification, which has been done for GRECOND in [12, 13]. Furthermore, we intend to investigate and utilize further appropriate transformation functions between the attribute and the factor spaces, in particular those suitable for approximate factorizations. A comparison with other data

dimensionality techniques, see e.g. the references in [17], in the presented scenario is also an important topic for future research. In this respect, both the impact on the classification accuracy as well as the transparency of the resulting classification model are important aspects to be evaluated in such a comparison. In a broader perspective, our paper shall help stimulate further research in utilization of Boolean matrix factorization in machine learning and data mining that involves Boolean data.

Acknowledgments We acknowledge support by the ESF project No. CZ.1.07/2.3.00/20.0059, the project is co-financed by the European Social Fund and the state budget of the Czech Republic (R. Belohlavek); Grant No. 202/10/P360 of the Czech Science Foundation (J. Outrata); and by the IGA of Palacky University, No. PrF_2012_029 (M. Trnecka).

References

1. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository University of California, Irvine, School of Information and Computer Sciences (2007). http://www.ics.uci.edu/mlearn/MLRepository.html
2. Belohlavek, R., Vychodil, V.: Discovery of optimal factors in binary data via a novel method of matrix decomposition. J. Comput. Sys. Sci. **76**(1), 3–20 (2010)
3. Belohlavek, R., Trnecka, M.: From-below approximations in Boolean matrix factorization: geometry and new algorithm (submitted, available at arXiv:1306.4905 [cs.NA])
4. Chung, Y., Lee, S.Y., Elston, R.C., Park, T.: Odds ratio based multifactor-dimensionality reduction method for detecting genegene interactions. Bioinformatics **23**(1), 71–76 (2007)
5. Ganter, B., Glodeanu, C.V.: Ordinal factor analysis. Lect. Notes Comput. Sci. **7278**, 128–139 (2012)
6. Ganter, B., Wille, R.: Formal Concept Analysis. Mathematical Foundations. Springer, Berlin (1999)
7. Geerts, F., Goethals, B., Mielikäinen, T.: Tiling, Databases. In: Proceedings DS 2004, LNCS, vol. 3245, pp. 278–289 (2004)
8. Kim, K.H. In: Dekker, M. (ed.): Boolean Matrix Theory and Applications (1982)
9. Kohavi, R.: A study on cross-validation and bootstrap for accuracy estimation and Model Selection. Proc. IJCAI, 1137–1145 (1995)
10. Mitchell, T.M.: Machine Learning. McGraw-Hill (1997)
11. Miettinen, P., Mielikäinen, T., Gionis, A., Das, G., Mannila, H.: The discrete basis problem. IEEE Trans. Knowl. Data Eng. **20**(10), 1348–1362 (2008). (preliminary version in PKDD 2006, pp. 335–346.)
12. Outrata, J. Preprocessing input data for machine learning by FCA. In: Proceedings CLA 2010, pp. 187–198, Sevilla, Spain
13. Outrata, J. Boolean factor analysis for data preprocessing in machine learning. In: Proceedings ICMLA 2010, pp. 899-902, Washington, D.C., USA
14. Outrata, J., Vychodil, V.: Fast algorithm for computing fixpoints of galois connections induced by object-attribute relational data. Inf. Sci. **185**(1) (114127)
15. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
16. Ritchie, M.D., et al.: Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. Am. J. Hum. Genet **69**, 138–147 (2001)
17. Tatti, N., Mielikäinen, T., Gionis, A., Mannila, H.: What is the dimension of your binary data? Proc. ICDM, 603–612 (2006)