J Intell Inf Syst (2007) 29:231–252
DOI 10.1007/s10844-006-0016-x

Bayesian networks for imputation in classification problems

Estevam R. Hruschka Jr. · Eduardo R. Hruschka · Nelson F. F. Ebecken

Received: 16 August 2004 / Revised: 28 November 2005 / Accepted: 7 December 2005 /
Published online: 24 January 2007

Abstract Missing values are an important problem in data mining. In order to tackle this problem in classification tasks, we propose two imputation methods based on Bayesian networks. These methods are evaluated in the context of both prediction and classification tasks. We compare the obtained results with those achieved by classical imputation methods (Expectation–Maximization, Data Augmentation, Decision Trees, and Mean/Mode). Our simulations were performed by means of four datasets (Congressional Voting Records, Mushroom, Wisconsin Breast Cancer and Adult), which are benchmarks for data mining methods. Missing values were simulated in these datasets by means of the elimination of some known values. Thus, it is possible to assess the prediction capability of an imputation method, comparing the original values with the imputed ones. In addition, we propose a methodology to estimate the bias inserted by imputation methods in classification tasks. In this sense, we use four classifiers (One Rule, Naïve Bayes, J4.8 Decision Tree and PART) to evaluate the employed imputation methods in classification scenarios. Computing times consumed to perform imputations are also reported. Simulation results in terms of prediction, classification, and computing times allow us performing several analyses, leading to interesting conclusions. Bayesian networks have shown to be competitive with classical imputation methods.

Keywords Missing values · Bayesian networks · Data mining

E. R. Hruschka Jr. (✉)
UFSCar/Federal University of São Carlos, São Carlos, Brazil
e-mail: estevam@dc.ufscar.br

E. R. Hruschka
Catholic University of Santos (UniSantos), Santos, Brazil

N. F. F. Ebecken
COPPE/Federal University of Rio de Janeiro, Rio de Janeiro, Brazil

1 Introduction

In real-life databases, some observations of one or more attributes (variables[1]) are typically missing. These missing values are a critical problem for data mining methods, which are usually not able to cope with them in an *automatic fashion*, i.e., without data preparation. In general, there are many approaches to deal with the problem of missing values (Han & Kamber, 2001): (a) Ignore the objects containing missing values; (b) Fill the gaps manually; (c) Substitute the missing values by a constant; (d) Use the mean or the mode of the objects as a substitution value; and (e) Get the most probable value to fill the missing ones.

The first approach involves removing the examples and/or attributes with missing values. However, the waste of data may be considerable and incomplete datasets may lead to biased statistical analyses. The second approach is unfeasible in data mining tasks, because of the huge dimensionality of the employed datasets. The third approach assumes that all missing values represent the same value, probably leading to considerable distortions. The substitution by the mean/mode value is a common practice and sometimes can even lead to reasonable results. However, it does not take into consideration the between-attribute relationships, which are useful to the process of missing values substitution. Therefore, we believe that the best approach involves trying to fill the missing values with the most probable ones.

The substitution of missing values, also called *imputation*, should not change important characteristics of the dataset. In this sense, it is necessary to define the important characteristics to be maintained. Data mining methods usually explore relationships between attributes and, thus, it is critical to preserve them, as far as possible, when replacing missing values (Pyle, 1999). In other words, the imputation goal is to carefully *substitute* missing values, trying to avoid the imputation of bias in the dataset. When the imputation is performed in a suitable way, higher quality data are produced, and the data mining outcomes can even be improved.

This work describes and evaluates two imputation methods based on Bayesian Networks (BNs). Following Schafer and Graham (2002), maximum likelihood methods (e.g., EM and Bayesian algorithms) represent the state of art for imputation. Considering high dimensional datasets, BNs are usually more efficient than methods based on the EM algorithm (Di Zio, Scanu, Coppola, Luzi, & Ponti, 2004). Since BNs may be seen as classification/regression models (Heckerman, 1995), the imputation of missing values may be seen as a prediction task (Hruschka Jr. & Ebecken, 2002). Thus, when having prior information about the data (i.e., dataset), BNs can be automatically generated and efficiently used to predict the most suitable values to *substitute* the missing ones.

The remainder of this paper is organized as follows. The next section approaches related works found in the literature. Section 3 describes our Bayesian imputation methods, which are evaluated in classification problems. The achieved results are compared with those obtained by some classical imputation methods—EM (Expectation–Maximization), Decision Trees, Data Augmentation and Mean/Mode imputation. Section 4 describes a methodology to estimate the bias inserted by imputation methods in the context of classification problems, as well as it reports our simulation results in four datasets that are benchmarks for data mining methods. Finally, Section 5 describes the conclusions and points out some future work.

[1]In this work the terms *attribute* and *variable* will be employed interchangeably.

2 Related work

Techniques to deal with missing values are not recent (Anderson, 1946; Dempster, Laird, & Rubin, 1977; Preece, 1971; Rubin, 1976) and many methods (most of them to survey data analysis) are described in the literature (Little & Rubin, 1987). However, considering data mining applications, missing values imputation has become an effervescent research area only in the last years.

When working with decision trees, some practical results about ignoring instances with missing values can be found in (Quinlan, 1986; White, 1987). Another approach involves replacing missing values by the most frequent value (Kononenko, Bratko, & Roskar, 1984) or by a nearest-neighbor based approach (Batista & Monard, 2003). In the probability method (Lobo & Noneao, 2000; Quinlan, 1989), a decision tree is constructed to determine the missing values of each attribute, using the information contained in other attributes and ignoring the class. The dynamic path generation (Quinlan, 1986) and the Lazy decision tree approach (Friedman, Kohavi, & Yun, 1996) do not generate the whole tree, but only the most promising path instead.

Missing data may distort the statistical pattern of the sample, thus statistical learning methods may not bring suitable results when employed in datasets containing missing values. More on the influence of missing data to representative samples can be found in (Gelman, Carlin, Stern, & Rubin, 1995; Little & Rubin, 1987; Rubin, 1976). Considering the application of Bayesian methods to randomly distributed missing data (Han & Kamber, 2001; Rubin, 1976), a common way to find the posterior distribution of the joint probabilities and the marginal probability of the variable containing missing values involves treating missing values as unknown parameters and applying a MCMC (Monte Carlo Markov Chain) method (Gilks, Richardson, & Spiegelhalter, 1996; Gilks & Roberts, 1996; Spiegelhalter, Thomas, & Best, 1996, 1999). The Bayesian *bound and collapse* algorithm (Sebastiani & Ramoni, 1997) works in two phases: *bounding* samples that have information about the missing values mechanism and encoding the other ones in a probabilistic model of non-response. Afterwards, the *collapse phase* generates a single value to substitute the missing ones.

In multivariate analysis, some works apply the Multiple Imputation (MI) (Rubin, 1977, 1987) method to handle missing data. MI consists in using more than one value to fill the gaps in the sample (for example the mean/mode of the more probable ones). The MI method can provide good estimations of the sample standard errors and any kind of analysis can be applied, but the computational cost is higher than in single imputation.

The EM (Expectation–Maximization) algorithm (Dempster et al., 1977) is a classical imputation method. It assumes that missing data (Y) are governed by a distribution $f(Y|X,\theta)$ where X (data without missing values) and θ (i.e., mean and variance) are fixed. In a nutshell, the EM algorithm consists in (Little & Rubin, 1987): (a) replace missing values by estimated values; (b) estimate parameters θ; (c) re-estimate the missing values assuming the new parameter estimates are correct; (d) re-estimate parameters, and so forth, iterating until convergence. In other words, the EM algorithm is based on the likelihood function, and it fills in the missing data based on a initial estimate of θ; re-estimates θ based on the complete and filled data, iterating until the estimates converge. Depending on the complexity of the density function that describes the dataset, the convergence may be slow. More on the convergence rate can be found in (Friedman, 1997; Jordan & Xu, 1996; Lam & Bacchus, 1994; Redner & Walker, 1984; Schwartz, 1978; Wu, 1983; Xu & Jordan, 1996). In addition, the computations necessary to the EM implementation are dependent on the particular application (density function and parameters). More details on the EM algorithm can be found in (Bilmes, 1997; Dempster et al., 1977; Ghahramami & Jordan, 1995; Jordan & Jacobs, 1994; Redner & Walker, 1984; Wu, 1983).

Following Tanner and Wong (1987), the Data Augmentation (DA) method may be informally described as the process in which observed data Y (whose distribution depends on parameters θ) is augmented by the quantity Z (using a Monte Carlo sampling strategy). Based on the multiple imputation idea, one can generate multiple values of Z using the $p(Z|Y)$ distribution and then obtain $p(\theta|Y)$ as the average of $p(\theta|Y,Z)$ over the imputed Z's. In theory, this method provides a way to improve the inference in small samples, in which the EM method has some pitfalls.

Di Zio et al. (2004) describe the use of Bayesian Networks (BNs) for imputing missing values, arguing that one of the main advantages of using BNs as imputation models is the possibility of preserving statistical relationships between variables, as well as dealing with high dimensional datasets. The proposed method uses the implementation of the PC algorithm (Spirtes, Glymour, & Scheines, 1993), available in the Hugin Software (Madsen et al., 2003), to learn a BN from data. Di Zio et al. (2004) have also proposed some adjustments to improve the network structure. These adjustments are aimed at defining an appropriate variable ordering, which is based on the concept of reliability. Further details concerning the differences between our method and the one proposed by Di Zio et al. (2004) are provided in Section 3. Finally, in the simulations to be described in Section 4 we compare our Bayesian methods with the following classical imputation methods:

(a) Mean/Mode (MM): mean imputation for ordinal attributes and Mode imputation for nominal attributes;
(b) Decision Trees (DT): the J4.8 algorithm, to be described in Section 4.4, was employed to construct decision trees to determine the missing values of each attribute, using the information contained in other attributes and ignoring the class.
(c) Expectation–Maximization (EM): we employ the implementation provided by the Norm Software (Schafer, 2000), which treats variables as if they were jointly normal. The two steps of EM are: (i) expectation (E)—missing sufficient statistics are replaced by their expected values given the observed data, using estimated values for the parameters; and (ii) maximization (M)—parameters are updated by their maximum-likelihood estimates, given the sufficient statistics obtained from the E step. The procedure is run until convergence is met.
(d) Data Augmentation (DA): we also employ the DA implemented at the Norm Software (Schafer, 2000) as an iterative simulation technique, i.e., a kind of Monte Carlo Markov Chain (MCMC) in which there are three types of quantities: observed data, missing data, and parameters. The missing data and the parameters are unknown. The implementation alternately performs two steps, under an assumption of joint normality. In the I-step, it imputes the missing data by drawing them from their conditional distribution given both the observed data and assumed values for the parameters; In the P-step, it simulates new values for the parameters by drawing them from a Bayesian posterior distribution given the observed data and the most recently imputed values for the missing data. Alternating between these two steps the Norm software sets up a Markov chain that converges to a stationary distribution, which is the joint distribution of the missing data and parameters given the observed data. DA may be regarded as a stochastic version of EM.

3 Bayesian imputation methods

A Bayesian Network (BN) is a directed acyclic graph in which the nodes represent the variables and the arcs represent a causal relationship among the connected variables. A conditional probability table gives the strength of such relationship. The variables that are not connected by an arc can be considered as having no direct causal influence on them. The strength of the causal influence is given by the conditional probability $P(x_i|\pi_{xi})$, in which x_i is the i-th variable, and π_{xi} is the set of parents of x_i. As described by Pearl (1988), the conditional independence assumption (Markov condition) allows one to calculate the joint distribution of all variables as:

$$P(x_1, x_2, \ldots, x_n|BK) = \prod_{i=1}^{n} P(x_i|\pi_{xi}, BK) \tag{1}$$

where BK represents Background Knowledge. Therefore, a BN can be used as a knowledge representation that allows inferences. Learning BNs from data became an effervescent research topic in the last decade (Gelman et al., 1995), and there are two main classes of methods to perform this task: methods based on heuristic search and methods based on the conditional independence (CI) definition to generate the network. There are also methods that combine these two strategies.

In a process of learning a BN from data, the variables of the BN represent the dataset attributes. When using algorithms based on heuristic search, the initial order of the dataset attributes is considered an important issue. Some of these algorithms depend on this ordering to determine the arcs direction such that an earlier attribute (in an ordered list) is a possible parent only of the later ones. On the other hand, conditional independence methods try to find the arcs direction without the need of ordering the attributes. However, even for the CI methods, when the ordering is known the algorithms can be improved (Spirtes et al., 1993). In our work, we apply a method (called K2χ^2), originated from a heuristic search based learning algorithm (K2), that applies the statistical χ^2 test to determine an initial order of attributes to improve the learning process.

3.1 K2 algorithm

The K2 algorithm (Cooper & Herskovits, 1992) constructs a BN from data and uses a heuristic search for doing so. It assumes that the attributes are discrete; the dataset is complete and has only independent cases; and all the attributes must be preordered. Considering these assumptions, K2 searches for a Bayesian structure that best represents the dataset.

The *attributes preorder* assumption is an important aspect of K2. It uses an ordered list (containing all the attributes including the class), which asserts that only the attributes positioned before a given attribute A may be parents of A. Hence, the first attribute in the list has no parent, i.e., it is a root node in the BN. In our work, the *class attribute* is always put as the first one in the ordered list.

The network construction uses a greedy method to search for the best structure. It begins as if every node had no parent. Then, beginning with the second attribute from the ordered list (the first one is the class), the possible parents are tested and those that maximize the whole probability structure are added to the network. This process is repeated to all attributes in order to get the best possible structure. Formally, the results of the following equation are maximized:

$$P(B_s, D) = c\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}N_{ijk}! \tag{2}$$

where each attribute x_i (i=1,...,n) has r_i possible values $\{v_{i1}, v_{i2}, \ldots, v_{ir_i}\}$. D is a dataset with m objects and B_s is a structure containing the attributes to be represented. Each attribute x_i within B_s has a set of parents π_{xi}, and q_i is the number of instantiations of π_{xi}. N_{ijk} is the number of objects in D, in which x_i has value v_{ik} and π_{xi} is instantiated as w_{ij} (w_{ij} represents the j-th instantiation relative to D of π_{xi}). Finally, N_{ij}=$\sum N_{ijk}$ and c is the a priori probability constant $P(B_s)$ of each B_s.

With the best structure already defined, the network conditional probabilities are determined. It is done using a Bayesian estimation of the (predefined) network structure probability. The Bayesian estimation is adopted in other learning Bayesian methods as in Spiegelhalter and Lauritzen (1990), but there are also other ways to compute this probability like, for instance, by variance analysis (Cooper & Herskovits, 1992).

3.2 K2χ^2 algorithm

The use of K2 for learning BNs from data is often motivated by its ability to find the network structure efficiently (Cano, Sordo, & Gutiérrez, 2004; Hsu, 2004) given that a reasonable variable ordering (VO) is provided. If the VO is not adequate, the quality of the learned structure may be low. Thus, searching for a good VO is relevant to the K2 learning process and it can be performed by: (a) using the knowledge of a human expert to help the definition of such an ordering; (b) performing an exhaustive search; (c) using heuristic search methods.

Di Zio et al. (2004) propose to order the variables according to their reliability. They consider more reliable those variables with a lower percentage of missing items, higher accuracy, and availability of external sources. The authors observe that reliable variables are frequently available when samples are drawn from a sampling frame where social or demographic variables are already known. Taking into account such domain knowledge, a single BN is constructed for imputation in all attributes with missing values. However, such a scenario is not common in data mining applications, and the assessment of the reliability of the variables may be hard to perform automatically. Actually, in data mining applications human experts are not always available to define the variable ordering, and the exhaustive search approach is usually not computationally feasible. Therefore, the adopted strategy usually involves performing a heuristic search.

It is particularly desirable to employ an efficient heuristic search for the K2 algorithm. Although it is known that the search for an adequate ordering is computationally less expensive than the search for the BN structure (Friedman, Linial, Nachman, & Pe'er, 2000), if the complexity of the VO search procedure is too high, one of the main characteristics of K2 (i.e., its fast performance in the learning process) may be affected and, in this case, other BN learning algorithms may be more suitable. For this reason, in this work we apply a heuristic based on the classical χ^2 statistical test (DeGroot, 1970) to define an appropriate VO before using K2.

As stated in (Cheng & Greine, 1999), the Chi-squared (χ^2) statistical test allows finding conditional independence relationships among attributes. Such relationships can be used as constraints to construct a BN. The PC algorithm (Spirtes et al., 1993), which is a classical

Table 1 Summary of the ALARM network structures achieved by K2 and K2χ^2 in relation to the original network structure described by Beinlinch et al. (1989)

Variable ordering (VO)	Correct arcs		Extra arcs	Missing arcs	ACCR (%)
	Correct direction	Wrong direction			
K2 (Cooper & Herskovits, 1992)	45	0	1	1	99.08
K2χ^2	16	27	39	3	98.99

ACCR = Average Correct Classification Rate

BN learning algorithm that does not depend on an initial VO, uses the χ^2 statistical test to measure the effectiveness of the relationships between pairs of attributes. Following this concept, we have applied the χ^2 statistical test to optimize the attribute ordering in the structure learning phase of K2. Therefore, our BN learning method is called K2χ^2. In order to define the VO, the χ^2 statistical test is performed in each attribute jointly with the class attribute. Thus, the strength of the dependence relationship between each attribute and the class attribute can be measured. Subsequently, the attributes are decreasingly ordered according to the χ^2 scores. The first attribute in the ordered list has the highest χ^2 score, i.e., it is the most dependent upon the class attribute. Obviously, the relation between the χ^2 statistical test and the best VO may not hold strictly, but some previous works (Hruschka Jr. & Ebecken, 2003; Hruschka Jr., Hruschka, & Ebecken, 2004), as well as our current paper, suggest that good results can be achieved.

Before providing a detailed description of our imputation methods, we illustrate the efficacy of the K2χ^2 learning algorithm when compared with the K2 originally proposed by Cooper and Herskovits (1992). To do so, we apply the K2χ^2 to construct a Bayesian Network (BN) from 10,000 cases generated from the classic ALARM BN (Beinlinch et al., 1989). The GENIE data generator (Druzdzel, 1999) was employed to obtain such cases. In (Cooper & Herskovits, 1992), the ALARM network is retrieved (reconstructed from data) assuming a given VO that is defined a priori, taking into account the observation of the original network. In this sense, a node X is added to the node-order list only after its parents have been inserted in the list. Using this approach to define the VO, Cooper & Herskovits got a network structure very similar to the original one, as indicated by the results reported in Table 1. However, the approach employed by Cooper and Herskovits (1992) to define the VO can only be applied when the network structure is known beforehand, or when domain knowledge is available. In our work, we assume that there is no *a priori* information concerning the VO. More precisely, our underlying assumption is that the VO estimated by means of the χ^2 test allows achieving accurate BNs. To apply the χ^2 test, information about a *class variable* is required. The ALARM network represents 8 diagnostic problems, 16 findings and 13 intermediate variables. In our experiments, we have chosen the variable 12 to be the *class variable*. This choice allows performing an interesting comparison between the network structures achieved by K2 and K2χ^2 (Table 1), provided that in the work of Cooper and Herskovits (1992) such variable is the first one in the VO. Since our work addresses the application of K2χ^2 for imputation in classification problems, it is particularly important to compare the results obtained by K2 and K2χ^2 in a classification task. From this standpoint, in Table 1 we also report the average correct classification rates (ACCRs), achieved by means of a 10-fold cross-validation strategy (Witten & Frank, 2000), for BNs constructed by K2 and K2χ^2. In summary, one can observe that, even with a network structure with some extra and missing arcs, the results obtained by K2χ^2 are very similar to the ones achieved by K2 using the VO taken from the original network. Also, good classification results have been reported for K2χ^2 in (Hruschka Jr. & Ebecken, 2003; Hruschka Jr. et al., 2004).

3.3 Imputation methods (BN-K2Iχ^2 and 1BN-K2Iχ^2)

We propose two imputation methods that rely on the construction of BNs to infer the most suitable values to fill in the gaps produced by missing values. For both methods, the K2χ^2 learning algorithm is applied to construct BNs to be used as prediction models to substitute the missing values. Particularly, we propose two variants for a method that we have called K2Iχ^2, which stands for K2 Imputation using the χ^2 statistical test, namely BN-K2Iχ^2 and 1BN-K2Iχ^2. Fundamentally, the former constructs one Bayesian network for each attribute with missing values, whereas the latter uses a single Bayesian network for imputation in all attributes with missing values. For both BN-K2Iχ^2 and 1BN-K2Iχ^2, the imputation process starts with the selection (from the original sample) of all sample objects that do not have missing data. These objects form a *clean sample,* which is then used as a *clean training dataset* to the construction of BNs.

BN-K2Iχ^2 identifies all attributes with missing values and for each one—now considered as a class attribute—a Bayesian network is generated. It means that if there are *n* attributes with missing values, *n* training *clean* samples are generated. More specifically, the imputation process performed by BN-K2Iχ^2 can be summarized as follows: (a) identify the attributes with missing values; (b) generate a *clean training* dataset for each class attribute identified in the previous step; (c) construct a BN for each *clean training* dataset; and (d) infer the best values to substitute the missing ones. In summary, a BN is generated for each attribute with missing values (here considered as the class attribute). As detailed in Section 3.1, the *class variable* is used as the first one in the VO. Thus, in principle our approach could be considered unsuitable for learning tasks in which a decision attribute (*class variable*) is not available in the dataset. To circumvent this problem, a hidden variable can be inserted in the network structure (Cooper & Herskovits, 1992) or other unsupervised or semi-supervised methods (Nigam, 2001) can be applied. However, as far as imputation problems are concerned, there will be no difficulties for the application of our approach even when a decision attribute is not available (e.g., unsupervised learning tasks). Since the steps (a) and (b) described above generate a dataset for each variable with missing values, such variable can be always considered as a *decision attribute* and, consequently, be accordingly positioned as the first one in the VO. Therefore, the absence of a decision attribute in the original sample will not impair the application of our imputation method. In a causal interpretation, putting the variable with missing values as the class attribute corresponds to assume that it may cause all the other variables.

Instead of generating one BN for each attribute with missing values, 1BN-K2Iχ^2 constructs a single BN to infer the best values to substitute the missing ones in all attributes. Thus, the imputation process performed by 1BN-K2Iχ^2 can be summarized in the following steps: (a) generate a single *clean training* dataset; (b) construct a BN using the single *clean training* dataset generated in the previous step; and (c) use the BN generated in step (b) to infer the best values to substitute the missing ones. Therefore, differently from BN-K2Iχ^2, 1BN-K2Iχ^2 is not suitable for unsupervised learning (clustering) tasks and, to evade this limitation, an unsupervised learning algorithm must be used instead of K2χ^2. However, as our work focuses on classification problems, this possibility will not be further explored in this paper.

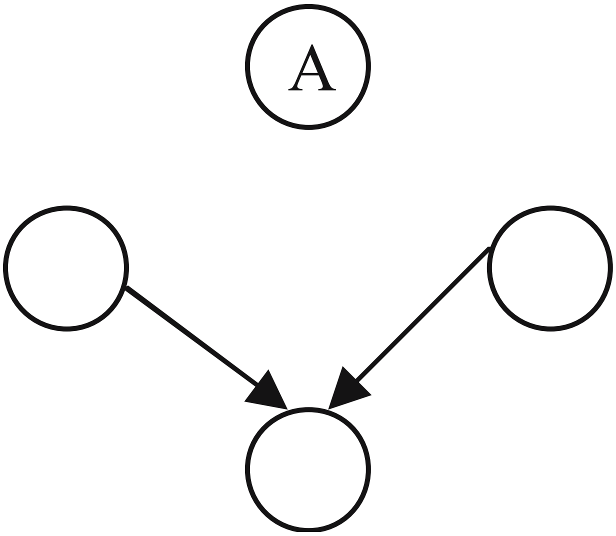

Fig. 1 Bayesian network without connection between the target attribute *A* and the other attributes

Having a VO focusing on each attribute tends to produce more accurate Bayesian networks for imputation. In this sense, the motivation for generating a Bayesian network for each attribute with missing values (BN-K2Iχ^2) relies on the optimization of the imputation process. On the other hand, generating a single Bayesian network (1BN-K2Iχ^2) tends to require less computation effort. These aspects are experimentally assessed in Section 4.

Depending on the proportion of missing values for a given attribute, substitution methods may have problems to infer suitable values to be imputed. From a general standpoint, there is a relationship between the quality of imputations and the availability of clean (without missing values) training examples. The more clean training examples, the better the imputations tend to be, because there is more information available. Thus, high proportions of missing values tend to degrade the results of imputation methods.

The K2Iχ^2 uses an inductive learning process (Mitchell, 1997) to generate the imputation model. Thus, similarly to several inductive learning methods, it assumes that the dataset attributes are related to each other. However, whenever a attribute *A* is independent from the other attributes, the BN structure will not connect attribute *A* with the other ones (for instance like illustrated in Fig. 1). In this case, since the K2Iχ^2 generates a probabilistic model, the imputation (or prediction) will be based only on the observed marginal distribution of attribute *A*. Putting it differently, if a specific attribute *A* has no relationship with the other ones, the K2Iχ^2 imputation efficacy would be based only on this attribute (*A*) marginal distribution (a priori knowledge) obtained from the objects without missing values. Regardless of some theoretical limitations, the next section shows that our method provides good results, mainly because attribute interactions wildly exist in many datasets.

4 Simulations

Classifications problems emerge in several data mining applications (e.g., in bioinformatics, business, text classification, etc.). This fact has motivated the evaluation of our imputation method in the context of classification tasks.

Fig. 2 Dataset D (Original)

Example / Attribute	a_1	a_2	a_3	a_4	Class
e_1	k_{11}	k_{12}	k_{13}	k_{14}	A
e_2	k_{21}	?	k_{23}	?	B
e_3	k_{31}	k_{32}	k_{33}	k_{34}	A
e_4	k_{41}	k_{42}	?	k_{44}	B
e_5	k_{51}	k_{52}	k_{53}	k_{54}	A
e_6	k_{61}	k_{62}	k_{63}	k_{64}	B
e_7	?	k_{72}	k_{73}	k_{74}	A
e_8	k_{81}	k_{82}	k_{83}	k_{84}	A
e_9	k_{91}	k_{92}	k_{93}	k_{94}	B

4.1 Theoretical aspects

In general, a dataset D is formed by examples with and without missing values. Let us call *C* (complete) the subset of examples of *D* that do not have missing values, and *M* (missing) the subset of examples of *D* with at least one missing value. In this context, imputation methods should carefully *complete* the missing values of *M*, originating a filled dataset *F*. Figures 2, 3, 4, and 5 illustrate these concepts considering an hypothetical dataset *D*, where a_j*s* are the attributes, e_i*s* are the examples, k_{ij} are known values (*i*-th example and *j*-th attribute), and the symbol "?" represents a missing value. In an ideal situation, the imputation method would fill in the missing values, originating filled values f_{ij}, without inserting bias in the dataset. In a more realistic view, one tries to decrease the amount of inserted bias to acceptable levels, in a way that a dataset $D' = \{C + F\}$, probably containing more information than *C* (in the sense that the attributes without missing values in *M* may contain important information), can be used for data mining (e.g., considering issues such as attribute selection, combining multiple models, and so on).

There are two ways of evaluating the *bias* inserted by an imputation method: in a prediction task and in a classification task. In a prediction task, one simulates missing values in C. Some known values are removed and then imputed. In this way, it is possible to evaluate how similar the imputed values are to the real, known values. The more similar the imputed value to the real one, the better the imputation method is. This alternative is efficient to compare different imputation methods, because it requires few computations after imputing values. However, it does not allow estimating the efficacy of the classifier in D′. In other words, although this procedure is valid, the prediction results are not the most important issue to be analyzed, but they are here reported for illustrative purposes. In reality, the substitution process must generate values that least distort the original

Fig. 3 Dataset C (Complete)

Example / Attribute	a_1	a_2	a_3	a_4	Class
e_1	k_{11}	k_{12}	k_{13}	k_{14}	A
e_3	k_{31}	k_{32}	k_{33}	k_{34}	A
e_5	k_{51}	k_{52}	k_{53}	k_{54}	A
e_6	k_{61}	k_{62}	k_{63}	k_{64}	B
e_8	k_{81}	k_{82}	k_{83}	k_{84}	A
e_9	k_{91}	k_{92}	k_{93}	k_{94}	B

Fig. 4 Dataset M (Missing Values)

Example / Attribute	a_1	a_2	a_3	a_4	Class
e_2	k_{21}	?	k_{23}	?	B
e_4	k_{41}	k_{42}	?	k_{44}	B
e_7	?	k_{72}	k_{73}	k_{74}	A

characteristics of D, which are given by the between-attribute relationships, defined by each particular classification algorithm. In a more practical view, the known values in M may contain important information, which, in turn, would be lost if their corresponding examples were discarded.

Several authors have also argued that it is more important to preserve the relationships between attributes than to get more accurate predictions. For instance, Pyle (1999) explains that although the imputed values are predictions, it is not the accuracy of these predictions that is of most importance when replacing missing values. It is more important that such predictions produce a workable estimate that least distorts the values that are actually present. In other words, the main purpose of imputation is not to use the values themselves, but to make available to the modeling tools the information contained in the other variables' values that are present. Di Zio et al. (2004) also argue that the imputation should preserve the between-attributes relationships as much as possible. To evaluate it in the context of sample surveys, they employ a probabilistic approach, analyzing the preservation of the joint distribution—statistical consistency. Schafer and Graham (2002), by their turn, observe that a missing value treatment cannot be properly evaluated apart from the modeling, estimation, or testing procedure in which it is embedded. Therefore, in our experiments, we have focused on the inserted biases in terms of classification results, which somehow allow evaluating to what extent the relationships between attributes are being maintained. Di Zio et al. also observe that "obtaining high values for the proportion of correct imputations is very difficult, emphasizing that it may be unnecessarily difficult." Finally, one must remember that in real-world applications the imputed values can not be compared with any value. Thus, since the evaluation of the prediction capability must be performed in a complete dataset, one could alternatively employ a sample of C to perform such evaluation. However, this procedure has the undesirable effect of lessening the information available for the evaluation of candidate imputation methods. In summary, the maintenance of the between-attributes relationships is the most important aspect to be evaluated in the imputation process. Under this perspective, we define the inserted bias as follows.

Definition 1 *The inserted bias is the magnitude of the change in the between-attribute relationships caused by patterns introduced by an imputation process.*

The problem is that the relationships between attributes are hardly known a priori, i.e., before the data mining process is performed. Therefore, usually the inserted bias can not be directly measured, but it can be estimated. In classification problems, the underlying assumption is that the between-attribute relationships are induced by a particular classifier.

Fig. 5 Dataset F (Filled)

Example / Attribute	a_1	a_2	a_3	a_4	Class
e_2	k_{21}	f_{22}	k_{23}	f_{24}	B
e_4	k_{41}	k_{42}	f_{43}	k_{44}	B
e_7	f_{71}	k_{72}	k_{73}	k_{74}	A

1) Evaluate the classifier's ACCR in a cross-validation process, using C;

2) Evaluate the classifier's ACCR in F (test set) considering that C is the training set;

3) The estimated inserted bias is the difference between the results achieved in 2) and 1).

Fig. 6 Estimating the inserted bias in a classification context

Consequently, the quality of such *discovered* relationships can be indirectly measured (estimated) by classification measures like, for instance, the Average Correct Classification Rate (ACCR) criterion. In this sense, we propose a methodology to estimate the inserted bias as follows.

In data mining applications, one usually employs different classifiers, choosing the best one according to some criterion of model quality (e.g., the ACCR). The underlying assumption is that the best classifier (BC)—in relation to C and to the available classifiers—should provide a suitable model for classifying examples of D. Thus, it is important to assess if the imputed values adjust themselves to the BC model. It is a common practice to evaluate classifier performance in a test set. The same concept can be adapted to evaluate imputations, considering C as the training set and F as the test set. In summary, we propose to estimate the *inserted bias* by the procedure described in Fig. 6:

According to Fig. 6, a *positive bias* is achieved when the Average Correct Classification Rate (ACCR) in F (step 2) is greater than in the cross validation process in C (step 1), i. e., the imputed values are likely to improve the classifier's ACCR in D′. By the same token, a *negative bias* is inserted when the imputed values are likely to worsen the classifier's ACCR. Finally, no bias is likely inserted when the accuracies in F and in the cross-validation process are equal, and this is the ideal situation. The imputation process should not introduce patterns into the data that are not present in the known values (Pyle, 1999), because these patterns may be later discovered during data mining in D′. Indeed, these artificial patterns are simply an artifact of the imputation process.

One could argue that a cross-validation process should be performed in both datasets (C and F), comparing the corresponding results. However, it is possible to get different models in C and F. For instance, different decision trees (e.g., in terms of selected attributes), but with similar accuracies in a cross-validation process, may be obtained in C and F. In this case, it is unlikely that a similar ACCR would be get in D′. In addition, we are interested in evaluating if the imputation process preserves the between-attribute relationships, assuming that these are defined by each particular classification model. In this sense, the complete dataset (C) is more reliable and, consequently, is its associated model. In other words, verifying how the data in F adjust themselves to the model obtained in C allows estimating the classifier ACCR in D′. In addition, performing cross-validation in both datasets (C and F) is computationally more expensive, because it is necessary to get and evaluate the classifier several times for both C and F.

4.2 Methodology

Our imputation methods were evaluated both in prediction and classification tasks, making possible to investigate relationships between these two aspects. To do so, we simulate missing values in complete datasets, eliminating some values that are a priori known. In this context, *artificial* missing values are imputed and compared with the original ones. We also

Table 2 Summary of dataset characteristics

Dataset	# examples	# classes	# attributes	Attribute type
Congressional Voting Records	232	2	16	Binary
Mushroom	5644	2	22	Nominal
Wisconsin Breast Cancer	683	2	9	Ordinal
Adult	45,222	2	14	8 Nominal/6 Ordinal

compare imputations performed by BN-K2Iχ^2 and 1BN-K2Iχ^2 with those obtained by Expectation—Maximization (EM), Data Augmentation (DA), Decision Trees (DT), and Mean/Mode (MM). For the sake of simplicity, from now on BN-K2Iχ^2 and 1BN-K2Iχ^2 will be called BN and 1BN respectively.

In our simulations, we employed four datasets that are data mining benchmarks: Congressional Voting Records, Mushroom, Wisconsin Breast Cancer, and Adult. These datasets are available at the UCI Machine Learning Repository (Merz & Murphy, 1997) and allow showing the applicability of our method in datasets formed by several types of attributes. Table 2 summarizes the main features of each employed dataset. Our Bayesian imputation methods were not used in continuous datasets, mainly because they require discrete valued datasets. However, this is not a severe limitation of BN and 1BN, because attributes formed by continuous values can be discretized. In this sense, the Wisconsin Breast Cancer can be viewed as an example of such discretized dataset.

Let us first consider how the datasets employed in our simulations were formed. As previously mentioned, some values from the original dataset were randomly eliminated in order to simulate the missing ones. It was performed by removing, independently, 30% of the values from each attribute. According to Rubin's definitions (1976), we are assuming a probability distribution of missingness called MCAR (missing completely at random). We have used stratified datasets, i.e., the class proportion was maintained in each dataset. Thus, according to Section 4.1, C and M are formed by 70% and 30% of D, respectively. In another perspective, we assume a missing value pattern that involves missing values in one attribute at a time. In this sense, for each dataset described in Table 2 (formed by n attributes), n simulated datasets are generated by eliminating the values of a particular attribute, i.e., the remaining $(n-1)$ attributes maintain their original values. Then, imputation methods (BN, 1BN, EM, DT, DA, and MM) are used to substitute the missing values in M. After the substitution process, all the imputed values are combined into a single dataset, which contains only imputed values, representing what we have called F in Section 4.1. In this way, we can better evaluate the inserted bias (Fig. 6) of each imputation method, because the classification results in F are not influenced by original values. Since we have employed six imputation methods for each dataset with missing values, six datasets with imputed values are generated—these could be called F1 (BN), F2 (1BN), F3 (EM), F4 (DT), F5 (DA), and F6 (MM).

Table 3 Typical execution times (in seconds) to impute missing values in one attribute

Dataset	BN	1BN	EM	DT	DA	MM
Congressional Voting	16	1	48	16	64	≅0
Mushroom	264	12	352	88	396	≅0
Wisconsin Breast Cancer	9	1	27	9	36	≅0
Adult	798	57	1,190	350	1,246	≅0

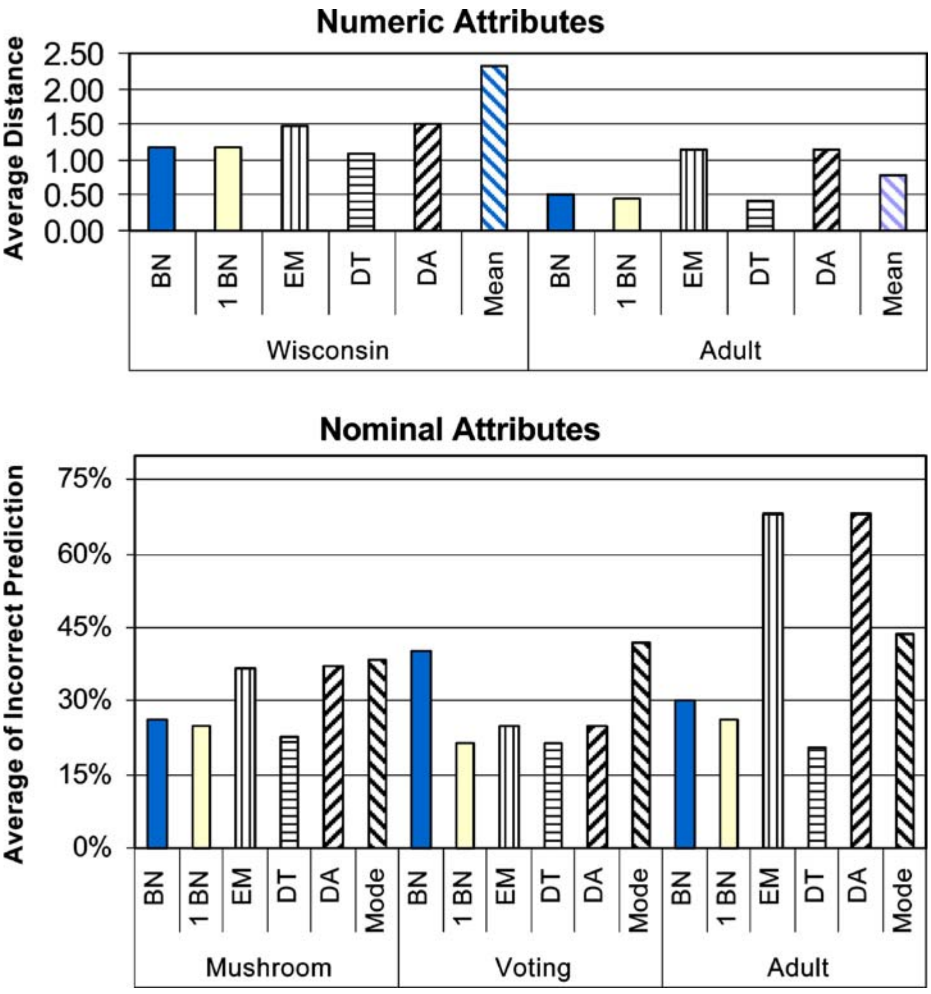

Fig. 7 Prediction results

In order to illustrate the computational efficiency of each imputation method, we report results of typical execution times necessary to impute missing values in each dataset. To do so, we employed a PC AMD 2.0 GHz, 512 MB of RAM, running on Windows XP and the achieved results are described in Table 3. In Section 4.5, we analyze these results in the light of both prediction and classification results, which are detailed in Sections 4.3 and 4.4 respectively.

4.3 Evaluating imputation as a prediction task

In this section, we compare the original values with the imputed ones. For nominal attributes, we report the average proportion (%) of incorrect imputations, that is the fraction (in relation to the total number of imputations) of incorrectly imputed values. For numerical attributes, we describe the average Euclidean distances between original and imputed values. The prediction results are illustrated in Fig. 7, where one can see that imputations

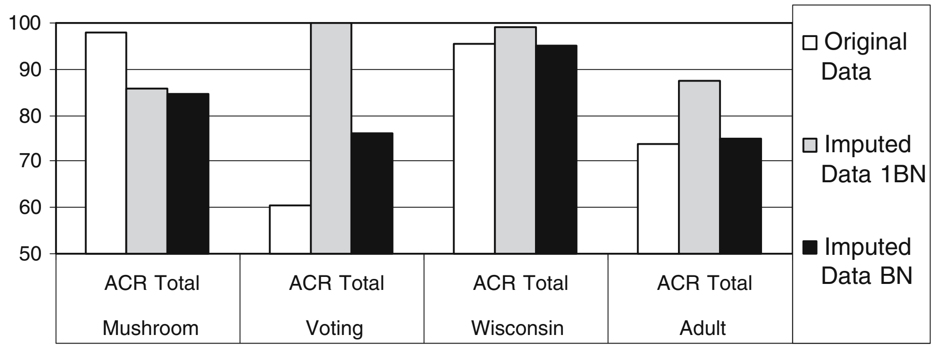

Fig. 8 ACCRs of BN in original (C) and imputed datasets (F)

performed by Bayesian networks (BN and 1BN) and DT (Decision Trees) have, in general, provided superior results in our simulations. In fact, imputations performed by DTs have slightly surpassed those obtained by the other employed imputation methods. In Fig. 8, we illustrate the influence of imputed values in classifications performed by $K2\chi^2$. In most of the datasets, the classification bias inserted by 1BN is greater than the one inserted by BN. This is the expected result, because the same Bayesian network is used for both imputation and classification by 1BN. However, in this work we are particularly interested in evaluating the efficacy (in terms of inserted bias) of BN and 1BN as data preparation tools for other classifiers. This aspect is addressed in the next section.

4.4 Evaluating imputation in classification tasks

This section reports results of simulations performed to evaluate the influence of imputed values in classification tasks. To do so, we employ the concepts described in Section 4.1, using four classifiers: One Rule, Naïve Bayes, J4.8 Decision Tree and PART. These classifiers are popular in the data mining community, and make part of the WEKA System (Witten & Frank, 2000), which was used to perform our simulations, using its default parameters. These classifiers can be briefly described as:

(a) One Rule (1R): it is an efficient and simple method that often produces good rules for characterizing the structure in the data. It generates a one-level decision tree, which is expressed in the form of a set of rules that test just one selected attribute.
(b) Naïve Bayes: it uses all attributes and allows them to make contributions to the decision that are equally important, and independent of one another given the class, leading to a simple scheme that works well in practice. Naïve Bayes has an inductive bias similar to the one of $K2\chi^2$. However, $K2\chi^2$ induces a network structure taking into account the possible dependences among the set of attributes, instead of using the naïve assumption, for which all the variables are connected directly to the class.
(c) J4.8: it is the Weka's implementation of the popular C4.5 decision tree learner (Quinlan, 1986). In fact, J4.8 is the later and slightly improved version, called C4.5 revision 8, which was the last public version of this family of algorithms before the commercial C5.0 was released (Quinlan, 1989).
(d) PART: this method provides rules from pruned partial decision trees built using C4.5. It combines the divide-and-conquer strategy of decision trees learning with the separate-and-conquer one for rule learning (Witten & Frank, 2000). In essence, to

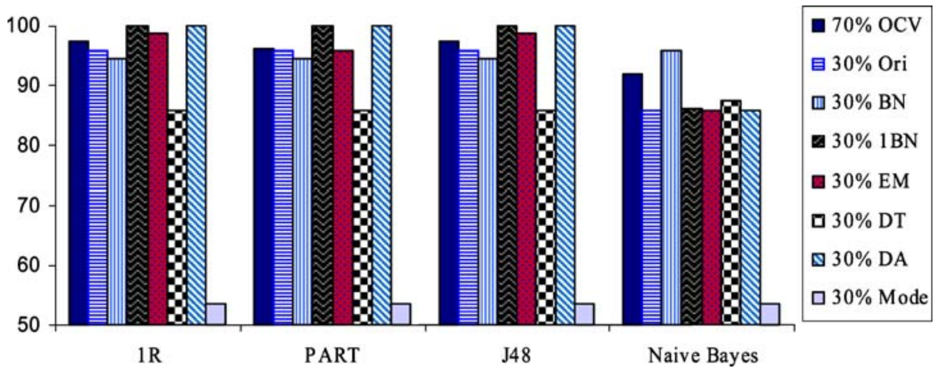

Fig. 9 Classification results—Congressional Voting Records Dataset

make a single rule, a pruned decision tree is built for the current set of examples. Then, the leaf with the largest coverage is made into a rule and the tree is discarded.

In order to estimate the classifier ACCR in the set C, we perform a ten-fold-cross validation process. In practical data mining applications, this estimate is usually considered a suitable criterion for choosing the best classifier, considering the available ones. Under this perspective, it is important to estimate the inserted bias (Fig. 6) considering the best classifier in each dataset.

Figures 9, 10, 11, and 12 show the simulation results in each dataset, considering the employed classifiers. In these figures, *70% OCV* stands for *70% 10-fold Cross-Validation in C,* whereas *30% Ori* corresponds to the original data used to simulate the missing values, originating the missing dataset (M), which was then filled by each imputation method. Consequently, the ACCRs in filled datasets are represented by 30%BN, 30%1BN, 30%EM, 30%DT, 30%DA, and 30%Mean/Mode, respectively for imputations by K2Iχ^2 (BN and 1BN), Expectation Maximization, Decision Trees, Data Augmentation and Mean/Mode. The results in *30% Ori* are reported only to give an idea of how the employed classifiers perform on the corresponding original data, i.e., they are useful for illustrative purposes only, because in practice these data would not be available.

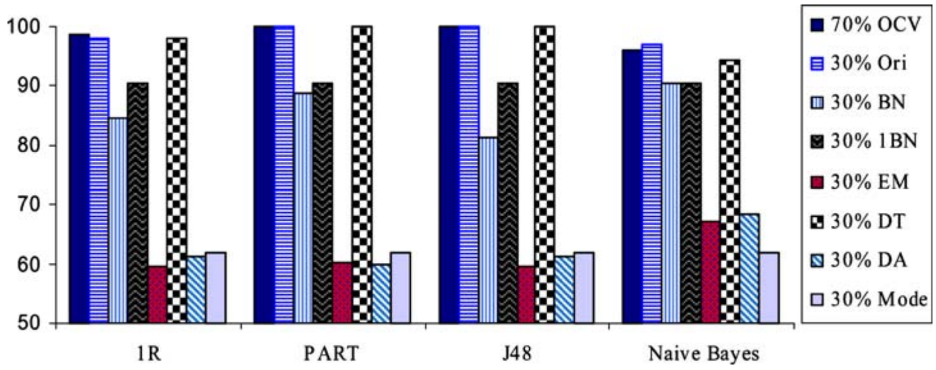

Fig. 10 Classification results—Mushroom Dataset

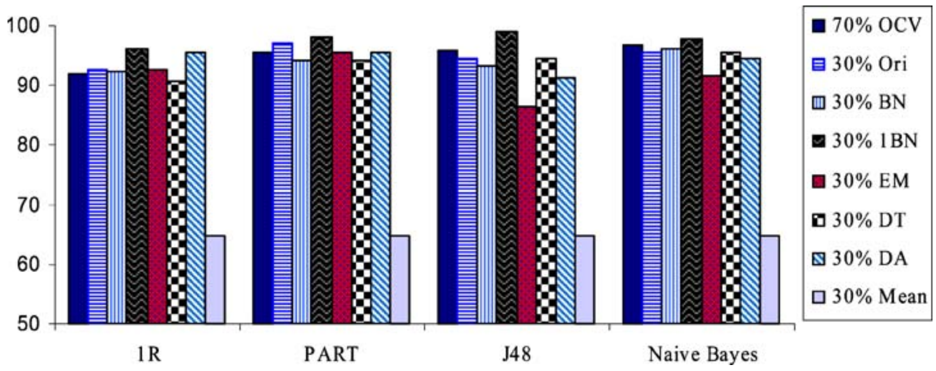

Fig. 11 Classification results—Wisconsin breast cancer dataset

Considering the estimated inserted bias (Fig. 6), in general imputations by K2Iχ^2 (BN and 1BN) provided better results in two datasets (Wisconsin Breast Cancer and Adult), whereas Decision Trees (DT) achieved better results in Mushroom, and Expectation–Maximization (EM) presented superior results in Congressional Voting Records. The assignment of constant values to substitute missing values (in our simulations the imputations performed by the Mean/Mode may be seen as a special case of such procedure) inserted the most important biases, and further important results obtained in each dataset are described in the sequel.

Figure 9 reports the results obtained in the Congressional Voting Records dataset. Two classifiers (1R and J4.8) provided the best results, in terms of the Average Correct Classification Rate (ACCR) in the complete dataset (70% OCV). In fact, 1R and J4.8 selected the same attribute for classification purposes. Considering the classifiers 1R and J4.8, the values imputed by EM inserted less bias (+1.07%) than other imputation methods. It is interesting to observe that, although the ACCRs in the data filled by DA and 1BN are equal to 100% (i.e., better than those provided by EM—98.59%) their corresponding inserted bias are equal to + 2.48%. In this sense, DA and 1BN may have inserted additional information in the dataset. As previously addressed in Section 4.1, this is an undesirable characteristic of an imputation method, which should permit the original information

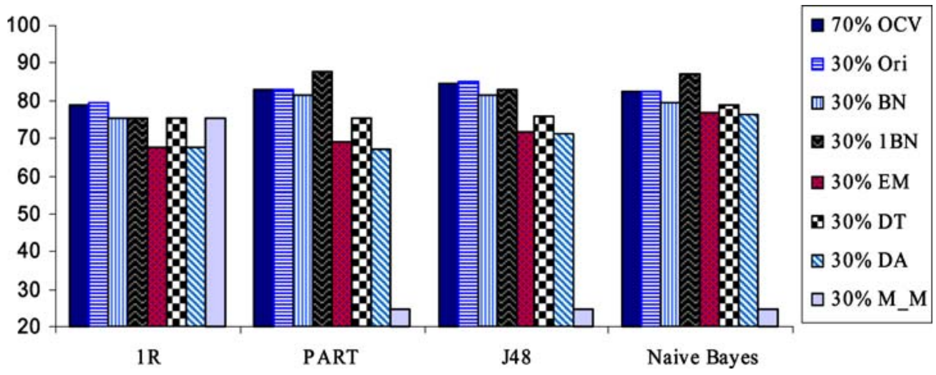

Fig. 12 Classification results—Adult Dataset

Table 4 Less inserted bias: italic type refers to the best classifier(s) in each dataset

Dataset/Classifier	1R	PART	J48	Naive Bayes
Congressional Voting Records	*EM*	EM	*EM*	BN
Mushroom	DT	*DT*	*DT*	DT
Wisconsin Breast Cancer	BN	EM/DA	DT	*BN*
Adult	BN/1BN/DT/Mean	BN	*1BN*	BN

(present in non-missing values) to be used by the employed classifier, ideally adding as little distortion to the dataset as possible.

In the Mushroom dataset (Fig. 10), classifiers PART and J4.8 have provided the best results in terms of ACCRs achieved in the complete dataset (70%OCV). In this context, imputations by decision trees (DT) have inserted no bias and this is the ideal result.

Figure 11 depicts simulation results in the Wisconsin Breast Cancer dataset, in which Naïve Bayes achieved the best classification results, and imputations by BN inserted less bias (−0.76%) than other methods.

Simulation results achieved in the Adult dataset are reported in Fig. 12. The classifier J4.8 provided the best results and 1BN inserted less bias (−1.44%) than other imputation methods.

Under a data mining perspective, simulation outcomes concerning the inserted bias in relation to the best classifier for each dataset may be regarded as the most important aspect to be observed. However, we believe that it is also interesting to provide details about what imputation method inserted less bias in each specific scenario (dataset/classifier). Table 4 provides an overview of the performance of the employed imputation methods. As it can be seen, K2Iχ^2 (BN and 1BN) have shown a slight superiority in relation to other imputation methods, providing the best results in 7 out of 16 scenarios. As far as the performance of BN and 1BN are taken into consideration, in 62.5% of our simulations BN has inserted less classification biases than 1BN, which, by its turn, has shown to be more computationally efficient than BN.

4.5 Summary of simulation results

So far, we have assessed our simulation results mainly considering each important aspect (prediction, classification bias and computing times) individually. In this section, we are mainly interested in evaluating relationships between prediction and classification (inserted bias considering the best classifier in each dataset) results, because, in principle, one is

Table 5 Congressional voting records dataset

Imputation method	IP (%)	Bias (%)—1R and J48	Computing times (s)
1BN	21.57	+2.48	1
DT	21.57	−11.60	16
DA	24.82	+2.48	64
EM	24.82	+1.07	48
BN	40.14	−3.16	16
Mode	41.81	−44.00	0

Table 6 Mushroom dataset

Imputation method	IP (%)	Bias (%)—PART	Bias (%)—J48	Computing times (s)
DT	22.76	0.00	0.00	88
1BN	24.81	−9.45	−9.45	12
BN	26.19	−11.16	−18.90	264
EM	36.87	−39.87	−40.40	352
DA	37.10	−40.17	−38.69	396
Mode	38.45	−38.22	−38.22	0

tempted to assume that better prediction results imply in less inserted bias. In order to facilitate the understanding of such relationships, our tables report the prediction results (Fig. 7) in an ascending order, i.e., from the best obtained result to the worse achieved result in terms of the maximum incorrect prediction (IP) or distance. Also, we here reproduce the computing times previously reported in Table 3.

Table 5 shows the results in the Congressional Voting Records dataset. Although EM provided the less inserted bias (+1.07%), its corresponding incorrect prediction results were not the best ones. Indeed, DT imputations were the best ones according to prediction results. The results obtained by 1BN shown a slightly inferior result in terms of inserted bias (+2.48%), but with associated computational times approximately 50 times faster than EM. Therefore, 1BN also provides very competitive results in this dataset.

The results achieved in the Mushroom dataset (Table 6) indicate that better predictions imply in less inserted bias. As already observed, this situation would be, in principle, the expected one. However, it was only observed in this dataset. Table 6 also shows that imputations by decision trees were superior in terms of inserted biases.

Table 7 describes the most important simulation results obtained in the Wisconsin Breast Cancer dataset, in which BN provided the best result in terms of the estimated inserted bias. It is particularly important to observe that 1BN has also provided very good results in this dataset, mainly if computational efficiency is considered in the analysis. Again, better results in terms of prediction did not necessarily imply in less inserted bias.

Imputations by 1BN inserted less bias in the Adult dataset (Table 8). As far as the computing times are concerned, only Mean/Mode imputations were more efficient than those performed by the 1BN. Since the Adult dataset is formed by nominal and ordinal attributes, we report two columns for prediction results (IP and Distance). One observes that there is a correlation between them. However, better predictions in general did not imply in less inserted bias.

Table 7 Wisconsin breast cancer dataset

Imputation method	Distance	Bias (%)—Naive Bayes	Computing times (s)
DT	1.08	−1.25	9
1BN	1.17	+0.91	1
BN	1.19	−0.76	9
EM	1.47	−5.15	27
DA	1.50	−2.23	36
Mean	2.31	−31.98	0

Table 8 Adult dataset

Imputation method	IP(%)	Distance	Bias (%)—J48	Computing times (s)
DT	20.66	0.41	−8.75	350
1BN	26.26	0.47	−1.44	57
BN	30.15	0.51	−3.09	798
Mean/Mode	43.53	0.77	−59.69	0
EM	68.06	1.13	−12.78	1,190
DA	68.21	1.14	−13.41	1,246

5 Conclusions and future work

In this work, we proposed two Bayesian methods (BN-K2Iχ^2 and 1BN-K2Iχ^2) for imputation, which is an important task in data mining. BN-K2Iχ^2 constructs one Bayesian network for each attribute with missing values, whereas the 1BN-K2Iχ^2 uses a single Bayesian network for imputation in all attributes with missing values. We have also elaborated on the bias inserted by imputation methods. From this standpoint, the imputation goal is to carefully *substitute* missing values trying to avoid the imputation of bias in the dataset. When the imputation is performed in a suitable way, higher quality data are produced, and the data mining outcomes can even be improved. Under this perspective, this paper also described a methodology to estimate the bias inserted by imputation methods in classification tasks.

The performance of BN-K2Iχ^2 and 1BN-K2Iχ^2 was illustrated by means of simulations performed in four datasets that are benchmarks for data mining methods: Mushroom, Congressional Voting Records, Wisconsin Breast Cancer and Adult. We assessed BN-K2Iχ^2 and 1BN-K2Iχ^2 in the context of prediction and classification tasks, using four classifiers (One-rule, J48, PART and Naïve Bayes). The achieved results were compared to those obtained by some classical imputation methods—Expectation–Maximization, Decision Trees, Data Augmentation, and Mean/Mode imputation. Computing times consumed to perform imputations were also reported. The analyses of such simulations lead to interesting conclusions. Considering the estimated inserted bias, imputations by Bayesian networks provided better results in two datasets (Wisconsin Breast Cancer and Adult), whereas Decision Trees achieved better results in the Mushroom dataset and Expectation–Maximization presented superior results in the Congressional Voting Records. The assignment of constant values to substitute the missing ones (in our simulations the imputations performed by the Mean/Mode may be seen as a special case of such procedure) inserted the most important biases. In most of our simulations, better prediction results did not imply in better classification results in terms of inserted bias. Taking into account the proposed imputation methods, BN-K2Iχ^2 tends to provide more accurate results. However, 1BN-K2Iχ^2 showed a better compromise between imputation quality and computational efficiency.

Our future work will concentrate on the evaluation of BN-K2Iχ^2 and 1BN-K2Iχ^2 in real-world data mining applications. We are also going to assess the sensitivity of our imputation methods with respect to other distributions of missingness (Schafer & Graham, 2002), i.e., MAR (missing at random) and MNAR (missing not at random, also called nonignorable missingness mechanism). Another interesting future work involves studying ways of extracting causal rules by means of imputation methods, trying to explain the missing data mechanism in real-world applications.

Acknowledgements The authors acknowledge the Brazilian research agencies CNPq, FAPESP and FAPERJ for their financial support. We are also grateful to Dr. Marek J. Druzdzel for providing us the GeNIe modeling environment developed by the Decision Systems Laboratory of the University of Pittsburgh (http://www.sis.pitt.edu/~dsl), which was very useful to generate the data for the construction of the ALARM network.

References

Anderson, R. L. (1946). Missing plot techniques. *Biometrics*, *2*, 41–47.

Batista, G. E. A. P. A., & Monard, M. C. (2003). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, *17*(5–6), 519–533.

Beinlich, I. A., Suermondt, H. J., Chavez, R. M., & Cooper, G. F. (1989). The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. *Proceedings of the Second European Conference on Artificial Intelligence in Medicine*, 247–256.

Bilmes, J. (1997). *A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models*. Technical Report, University of Berkeley, ICSI-TR-97-021.

Cano, R., Sordo, C., & Gutiérrez, J. M., (2004). Applications of Bayesian networks in meteorology. In J. A. Gámez, et al. (Eds.), *Advances in Bayesian networks* (pp. 309–327). Springer-Verlag.

Cheng, J., & Greine, R. (1999). Comparing Bayesian network classifiers. *Proc. of the fifteenth conference on uncertainty in artificial intelligence (UAI '99)* (pp. 101–108). Sweden.

Cooper, G., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, *9*, 309–347.

DeGroot, M. H. (1970). *Optimal statistical decision*. New York: McGraw-Hill.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*, 1–39.

Di Zio, M., Scanu, M., Coppola, L., Luzi, O., & Ponti, A. (2004). Bayesian networks for imputation. *Journal of the Royal Statistical Society A*, *167*(Part 2), 309–322.

Druzdzel, M. J. (1999). SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: A development environment for graphical decision-theoretic models (Intelligent Systems Demonstration). In *Proceedings of the sixteenth national conference on artificial intelligence (AAAI-99)* (pp. 902–903). Menlo Park, CA: AAAI Press/The MIT Press.

Friedman, N. (1997). Learning belief networks in the presence of missing values and hidden variables. *Proceedings of the 14th International Conference on Machine Learning*.

Friedman, H. F., Kohavi, R., & Yun, Y. (1996). Lazy decision trees. In *Proceedings of the 13th national conference on artificial intelligence* (pp. 717–724). Cambridge, MA: AAAI Press/MIT Press.

Friedman, N., Linial, M., Nachman, I., & Pe'er, D. (2000). Using Bayesian networks to analyze expression data. *Proc. of the fourth international annual conference on computational molecular biology* (pp. 127–135). New York: ACM Press.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (1995). *Bayesian data analysis*. London: Chapman & Hall.

Ghahramami, Z., & Jordan, M. (1995). *Learning from incomplete data* (Tech. Rep. AI Lab Memo No. 1509, CBCL Paper N°. 108). MIT AI Lab.

Gilks W. R., Richardson, S., & Spiegelhalter D. J. (1996). *Markov chain Monte Carlo in practice*. London: Chapman & Hall.

Gilks, W. R., & Roberts, G. O. (1996). Strategies for improving MCMC. In W. R. Gilks, S. Richardson, & D. J. Spiegelhalter (Eds.), *Markov chain Monte Carlo in practice* (pp. 89–114). London: Chapman & Hall.

Han, J., & Kamber, M. (2001). *Data mining: Concepts and techniques.* Morgan Kaufmann.

Heckerman, D. (1995). A tutorial on learning Bayesian networks. *Technical Report MSR-TR-95-06*. Microsoft Research, Advanced Technology Division, Microsoft Corporation.

Hruschka Jr., E. R., & Ebecken, N. F. F. (2002). Missing values prediction with K2. Intelligent Data Analysis *6*(6). (The Netherlands)

Hruschka Jr., E. R., & Ebecken, N. F. F. (2003). Variable ordering for bayesian networks learning from data. In *Proceedings of the international conference on computational intelligence for modeling, control and automation—CIMCA'2003, Vienna*.

Hruschka Jr., E. R., Hruschka, E. R., & Ebecken, N. F. F. (2004). Feature selection by Bayesian networks. *Lecture Notes in Artificial Intelligence*, *3060*, 370–379.

Hsu, W. H. (2004). Genetic wrappers for feature selection in decision tree induction and variable ordering in Bayesian network structure learning. *Information Sciences*, *163*, 103–122.

Jordan, M., & Jacobs, R. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, *6*, 181–214.

Jordan, M., & Xu, L. (1996). Convergence results for the EM approach to mixtures of experts architectures. *Neural Networks*, *8*, 1409–1431.

Kononenko, I., Bratko, I., & Roskar, E. (1984). *Experiments in automatic learning of medical diagnostic rules* (Tech. Rep.). Ljubjana, Yogoslavia: Jozef Stefan Institute.

Lam, W., & Bacchus, F. (1994). Learning Bayesian belief networks, an approach based on the MDL principle. *Computational Intelligence*, *10*, 269–293.

Little, R., & Rubin, D. (1987). *Statistical analysis with missing data*. New York: Wiley.

Lobo, O. O., & Noneao, M. (2000). Ordered estimation of missing values for propositional learning. *Journal of the Japanese Society for Artificial Intelligence*, *15*(1), 499–503.

Madsen, A. L., Lang, M., Kjærulff, U. B., & Jensen, F. (2003). The Hugin Tool for learning Bayesian Networks. *Lecture Notes in Computer Science*, *2711*, 594–605.

Merz, C. J., & Murphy, P. M. (1997). *UCI Repository of Machine Learning Databases*. Retrieved from http://www.ics.uci.edu. Irvine, California: University of California, Department of Information and Computer Science.

Mitchell, T. (1997). *Machine learning*. New York: McGraw-Hill.

Nigam, K. (2001). *Using unlabeled data to improve text classification* (Tech. Rep. CMU-CS-01-126). Doctoral dissertation, Computer Science Department, Carnegie Mellon University.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Mateo, CA: Morgan Kaufmann.

Preece, A. D. (1971). Iterative procedures for missing values in experiments. *Technometrics, 13*, 743–753.

Pyle, D. (1999). *Data preparation for data mining*. San Diego, CA: Academic.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106.

Quinlan, J. R. (1989). Unknown attribute values in induction. *Proceedings of 6th international workshop on machine learning* (pp. 164–168). Ithaca, NY.

Redner, R., & Walker, H. (1984). Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, *26*(2), 152–239.

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, *63*, 581–592.

Rubin, D. B. (1977). Formalizing subjective notion about the effects of nonrespondents in samples surveys. *Journal of the American Statistical Association*, *72*, 538–543.

Rubin, D. B. (1987). *Multiple imputation for non responses in surveys*. New York: Wiley.

Schafer, J. L. (2000). *Analysis of incomplete multivariate data*. London: Chapman & Hall/CRC.

Schafer, J. L., & Graham, J. W. (2002). Missing data: Our view of the state of the art. *Psychological Methods*, *7*(2), 147–177.

Schwartz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, *6*, 461–464.

Sebastiani, P., & Ramoni, M. (1997). *Bayesian inference with missing data using bound and collapse* (Tech. Rep. KMI-TR-58). KMI, Open University.

Spiegelhalter, D. J., & Lauritzen, S. L. (1990). Sequential updating of conditional probability on direct graphical structures. *Networks*, *20*, 576–606.

Spiegelhalter, D. J., Thomas, A., & Best, N. G. (1996). Computation on Bayesian graphical models. *Bayesian Statistics*, *5*, 407–425. Retrieved from http://www.mrc.bsu.cam.ac.uk/bugs.

Spiegelhalter, D. J., Thomas, A., & Best, N. G. (1999). *WINBUGS: Bayesian inference using Gibbs sampling, Version 1.3*. Cambridge, UK: MRC Biostatistics Unit.

Spirtes P., Glymour C., & Scheines R. (1993). *Causation, predication, and search*. New York: Springer-Verlag.

Tanner, M. A., & Wong, W. H. (1987). The calculation of posterior distributions by data augmentation (with discussion). *Journal of the American Statistical Association*, *82*, 528–550.

White, A. P. (1987). Probabilistic induction by dynamic path generation in virtual trees. In M. A. Bramer (Ed.), *Research and development in expert systems III*, (pp. 35–46). Cambridge: Cambridge University Press.

Witten, I. H., & Frank, E. (2000). *Data mining—practical machine learning tools and techniques with java implementations*. USA: Morgan Kaufmann Publishers.

Wu, C. F. J. (1983). On the convergence properties of the EM algorithm. *The Annals of Statistics*, *11*(1), 95–103.

Xu, L., & Jordan, M. (1996). On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, *8*, 129–151.