Chapter 5
Roles Played by Bayesian Networks in Machine Learning: An Empirical Investigation

Estevam R. Hruschka Jr. and Maria do Carmo Nicoletti

Abstract. Bayesian networks (BN) and Bayesian classifiers (BC) are traditional probabilistic techniques that have been successfully used by various machine learning methods to help solving a variety of problems in many different domains. BNs (and BCs) can be considered a probabilistic graphical language suitable for inducing models from data aiming at knowledge representation and reasoning about data domains. The main goal of this chapter is the empirical investigation of a few roles played by BCs in machine learning related processes namely (i) data pre-processing (feature selection and imputation), (ii) learning and (iii) post-processing (rule generation). By doing so the chapter contributes with organizing, specifying and discussing the many different ways Bayes-based concepts can successfully be employed in automatic learning.

5.1 Introduction

Since the beginning of the past decade Bayesian networks (BNs) (also known as belief networks or directed probabilistic graphical models) have been attracting a great deal of attention and have been successfully applied to solve a variety of problems in many different domains, most of them related to modeling and decision under uncertainty. They have been used in domains such as medicine (Díez *et al.* 1997) (Husmeier *et al.* 2005), molecular biology (Friedman 2004) (Sachs *et al.* 2005), genomics (Sebastiani *et al.* 2003) (Friedman *et al.* 2000) (Jansen *et al.* 2003), agricultural (Bressan *et al.* 2009) and many others. An overview of the main applications involving BNs can be seen in (Lauritzen 2003) and more recently in (Pourret *et al.* 2008).

Estevam R. Hruschka Jr. · Maria do Carmo Nicoletti
Computer Science Department, UFSCar, S. Carlos, SP, Brazil
e-mail: `estevam@dc.ufscar.br`

Maria do Carmo Nicoletti
FACCAMP, C. L. Paulista, SP, Brazil
e-mail: `carmo@dc.ufscar.br`

BNs can be considered a probabilistic graphical language for knowledge representation and reasoning. A BN (Pearl 1988) has a DAG (directed acyclic graph) structure. Each node in the graph corresponds to a discrete random variable in the domain. Edges represent conditional dependencies; an edge $Y \rightarrow X$ describes a parent-child relation, where Y is the parent and X is the child. Nodes that are not connected represent variables that are conditionally independent of each other. Each node of the BN structure is associated with a conditional probability table (CPTable) specifying the probability of each possible state of the node, given each possible combination of states of its parents.

A Bayesian classifier (BC) is a particular type of BN that aims at correctly predicting the value of a discrete class variable, given a vector of feature values. As pointed out in (Heckerman *et al.* 2000), BNs and BCs are usually employed in data mining tasks mainly because they (i) may deal with incomplete datasets straightforwardly; (ii) can learn causal relationships; (iii) may combine prior knowledge with patterns learnt from data and (iv) can help to avoid overfitting. Since Bayesian classifiers are a particular type of Bayesian networks, most of the related concepts and results are valid for both.

The main goal of this chapter is to empirically investigate possible roles played by Bayesian classifiers in three main subprocesses of machine learning processes namely: (1) data pre-processing (imputation and feature selection), (2) learning and (3) post-processing (rule generation and pruning). Although the natural order to approach machine learning (ML) subprocesses is the sequential order as stated above, in this chapter the learning of BNs and BCs will be discussed first since algorithms used for learning can be used for pre-processing as well as post-processing the data.

Besides the Introduction, the chapter is organized in six more sections. Section 2 introduces several of the underlying concepts involved in BNs and BCs, focusing on those that are relevant to some of the roles played by Bayesian models discussed in the chapter. Section 3 approaches BNs and BCs as knowledge representations and briefly presents the main ideas of three important algorithms: the Naïve Bayes (Duda and Hart 1973), PC (Spirtes *et al.* 1993) and K2 (Cooper and Herskovits 1992), used for learning BNs and BCs. Sections 4 and 5 address the use of Bayesian Classifiers for modifying the original training data available, aiming at improving its quality. The help provided by BN-based methods will specifically be investigated when the original training data patterns (a) are described by features that might be irrelevant (or superfluous) for the purpose of the learning task at hand (Section 4) and/or (b) have missing feature values, a recurrent and commonly problem found in collected data (Section 5). Section 6 describes in detail how BNs can be post-processed in order to create a set of rules (Hruschka Jr. *et al.* 2008). In Section 7 the main conclusions of the work described in the chapter are summarized.

5.2 Relevant Concepts Related to Bayesian Networks and Bayesian Classifiers

The issues discussed in this chapter are dependent on several concepts used in Bayes theory which, in turn, are heavily dependent on the probability theory. A brief review of the main relevant concepts is presented next. Most of the concepts are defined using the notation borrowed from (Friedman *et al.* 1997) and many can be revisited in Moore´s tutorials (Moore 2011). Consider:

- Lowercase letters denote specific values taken by those variables (e.g. x, y, z)
- Boldface capital letters denote sets of variables (e.g **X**, **Y**, **Z**)
- Boldface lowercase letters (e.g. **x**, **y**, **z**) denote assignments of values to the variables in sets **X**, **Y**, **Z** respectively. (Val(**X**) is used in the obvious way)
- A finite set of discrete random variables $\psi = \{X_1, X_2, \ldots, X_n\}$
- Each variable X_i may take on values from a finite set, denoted by $Val(X_i)$, $i=1,\ldots,n$.
- Capital letters will be used for variable names (e.g. X, Y, Z)

Definition 1. The probability that variable X takes the value x will be denoted $P(X = x)$ (or P(x) when there is no risk of ambiguity). The *joint probability distribution* (JPD) over n random variables $X_1, X_2, \ldots, X_n$ encodes the probability of a particular assignment to all the variables i.e. $P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$ or simply $P(x_1, x_2, \ldots, x_n)$ ♦.

Definition 2. The *conditional probability* that a random variable X takes on the value x given some other random variable Y takes on the value y is written P(x|y) and is defined by eq. (1) provided that $P(y) > 0$ ♦.

$$P(x|y) = \frac{P(x,y)}{P(y)} \tag{1}$$

Eq. (1) can be generalized for a set of random variables $X_1, X_2, \ldots, X_n$ and $Y_1, Y_2, \ldots, Y_m$ as eq. (2), provided that $P(y_1,y_2,\ldots,y_m) > 0$.

$$P(x_1, x_2,\ldots, x_n|y_1, y_2, \ldots, y_m) = \frac{P(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m)}{P(y_1, y_2, \ldots, y_m)} \tag{2}$$

Using the notation described at the beginning of this section, eq. (2) can be rewritten as eq. (3).

$$P(\mathbf{x}|\mathbf{y}) = \frac{P(\mathbf{x},\mathbf{y})}{P(\mathbf{y})} \tag{3}$$

Two sets of random variables being conditionally independent of a third set is a fundamental concept for establishing a few others concepts as well as a few procedures in a learning environment based on Bayesianism. The concept is formalized in *Definition 3.*

Definition 3. Let P be a joint probability distribution over the variables in ψ and let **X**, **Y**, **Z** be subsets of U. **X** and **Y** are said to be *conditionally independent* given **Z** noted as $I(\mathbf{X},\mathbf{Y}|\mathbf{Z})$ if for all $\mathbf{x} \in Val(\mathbf{X})$, $\mathbf{y} \in Val(\mathbf{Y})$, $\mathbf{z} \in Val(\mathbf{Z})$, $P(\mathbf{x}|\mathbf{z},\mathbf{y}) = P(\mathbf{x}|\mathbf{z})$ whenever $P(\mathbf{y},\mathbf{z}) > 0$. ♦

Bayesian networks (BN) belong to the family of probabilistic graphical models (GMs); more specifically, they are represented by a directed acyclic graph (DAG). As mentioned in (Murphy 1998), BNs enable an effective representation and computation of the JPD over a set of random variables. In a DAG the set of parents of a node X is represented by $\pi(X)$. By having the structure of an acyclic graph, it can be guaranteed that there is no node in the BN that can be its own ancestor or its own descendent. Such a condition, as mentioned in (Ben-Gal 2007), is of vital importance to the factorization of the joint probability of a collection of variables (nodes).

The specification of its DAG structure is considered the "qualitative" aspect of a BN. As it will be seen later in the chapter (Section 3.2), the concept of skeleton of a DAG (Definition 4) can be used during its construction. The specification of its "quantitative" aspect is done by specifying the conditional probability distribution at each node i.e., specifying the probability of each possible state of the node, given each possible combination of states of its parents. As pointed out in (Ben-Gal 2007), for discrete random variables, the conditional probability distribution is often represented by a table listing the local probability that the corresponding child node takes on each of its feasible values, for each combination of values of its parents. The joint distribution of a collection of variables can be determined uniquely by these local conditional probability tables (CPTables). Definition 5 gives a formal definition of BN based on the one proposed in (Friedman *et al.* 1997).

Definition 4. Let G be a DAG (directed acyclic graph). The *skeleton* of G is the undirected graph obtained from G by replacing its arcs with undirected edges ♦.

Definition 5. Consider the finite set of discrete random variables $\psi = \{X_1, X_2, \ldots, X_n\}$ where each variable X_i may take on values from a finite set. A Bayesian network for ψ is a pair $B = <G,\Theta>$. G is a directed acyclic graph (DAG) whose vertices correspond to the random variables $X_1, X_2, \ldots, X_n$ and whose arcs represent direct dependencies between the variables. A conditional dependency (which can be seen as a causal relationship) between two variables X_i and X_k defines an arc. The arc $X_k \rightarrow X_i$ describes a parent-child relation, where X_k is the parent and X_i is the child. Nodes that are not connected represent variables that are conditionally independent of each other. The graph G encodes independence assumptions: each variable X_i is independent of its nondescendants given its parents in G.

The second component of the pair i.e. Θ, represents the set of parameters that quantifies the network. It contains a parameter $\theta_{x_i|\pi_{x_i}} = P_B(x_i|\pi_{X_i})$ for each possible value x_i of X_i, and π_{x_i} of π_{X_i}, where π_{X_i} denotes the set of parents of X_i in G. A Bayesian network B defines a unique *joint probability distribution* over ψ given by eq. (4).

$$P_B(X_1,X_2,\ldots,X_n) = \prod_{i=1}^{n} P_B(X_i \mid \pi_{X_i}) = \prod_{i=1}^{n} \theta_{X_i \mid \pi_{X_i}} \qquad ♦ \text{ (4)}$$

If a variable X_i has no parents its local probability distribution is referred to as *unconditional*, otherwise it is *conditional*. Also, if the variable represented by a node is observed, the node is said to be an *evidence node* otherwise it is said to be a *hidden node*.

A Bayesian classifier (BC) is a particular kind of BN that aims at correctly predicting the value of a discrete class variable, given the value of a vector of feature variables.

As proved in (Pearl 1988) the only nodes that have influence on the conditional probability distribution of a given node X (given the state of all the remaining nodes) are the nodes that belong to the Markov Blanket of X, an important concept formalized in Definition 6.

Definition 6. In a Bayesian network structure let λ_X represent the set of children of node X and π_X represent the set of parents of node X. The subset of nodes containing λ_X, π_X and any other parents of λ_X is called *Markov Blanket* (MB) of X. ♦

Fig. 1 shows a pictorial representation of the Markov Blanket of a variable X in a given Bayesian network.

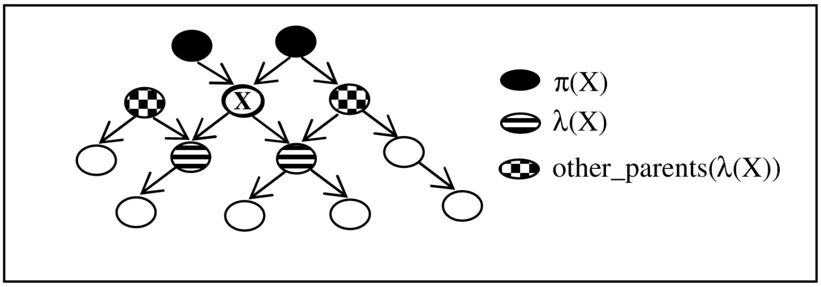

Fig. 1 MB(X) = { Z | Z ∈ π(X) or Z ∈ λ(X) or Z ∈ other_parents(λ(X))}

MB(X) contains all the nodes that shield X from the rest of the BN, i.e., the MB(X) is the only knowledge needed to predict the value of X. The concept of *moral graph* presented in Definition 7 is employed in the junction tree algorithm (Pearl 1988) which is used in belief propagation on graphical models.

Definition 7. Let G=<N1,A> be a DAG. Its counterpart *moralized graph*, G1=<N2,E> is a graph such that N1=N2 and E = {e | e = undirected(a), for all a ∈ A} ∪ {e_new | e_new = (n1,n2), n1≠n2, | ∃ <n1,nk> ∈ A ∧ ∃ <n2,nk> ∈ A} ♦.

The corresponding moral graph of the DAG shown in Fig. 1 is shown in Fig. 2, where the three new added arcs are shown in thicker lines. A BN represents the conditional independence of a node and its predecessors, given its parents; the conditional independence test can be used for directing the construction of BNs. The concept of direction-dependent separation (*d-separation*), formally introduced in Definition 9 can be used to identify d-separated nodes in a BN.

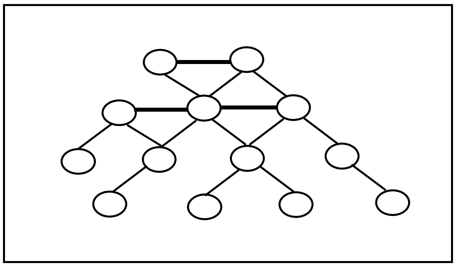

Fig. 2 Moral graph from the DAG shown in Fig. 1

Let G=<N,A> be a BN and let X ⊆ N, Y ⊆ N and E ⊆ N be three subsets of nodes. It can be proved that if every undirected path from a node in X to a node in Y is d-separated by E, then X and Y are *conditionally independent* given E. The proof that d-separated nodes are conditionally independent is elaborated and can be found in (Pearl 1988).

Definition 8. Let G=<N,A> be a BN. An *undirected path* in G is a path that does not take into account the directions of the arcs ♦.

Definition 9. (Russell and Norvig 1995) A set of nodes E *d-separates* two sets of nodes X and Y if every undirected path from a node in X to a node in Y is *blocked* given E. A path is blocked given a set of nodes E if there is a node Z on the path for which one of three conditions holds:

(1) Z is in E and Z has one arrow on the path leading in and one arrow out (chain).
(2) Z is in E and Z has both path arrows leading out.
(3) Neither Z nor any descendant of Z is in E, and both path arrows lead in to Z♦.

Fig. 3 based on (Russell and Norvig 1995) shows a pictorial representation of situations (1), (2) and (3) of Definition 9.

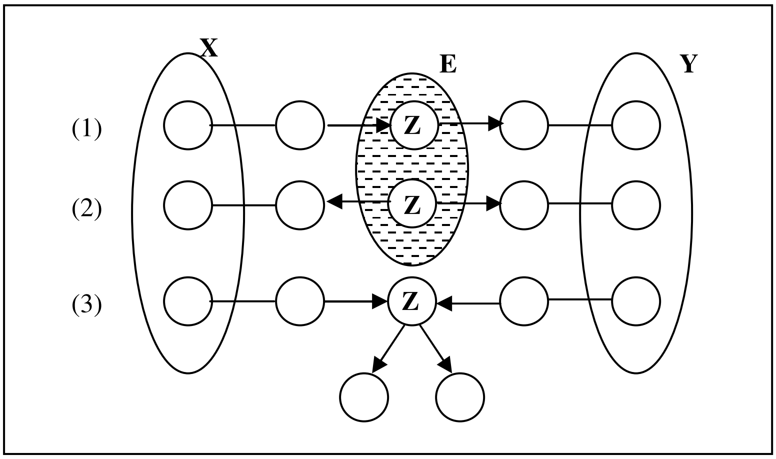

Fig. 3 Pictorial representation of the three situations a path from a node in X to a node in Y can be blocked, given the evidence E. If every path from X to Y is blocked, E d-separates X and Y

5.3 Learning Bayesian Networks and Bayesian Classifiers from Data

This section discusses the learning of Bayesian networks and Bayesian classifiers from data. Originally BNs were manually constructed by taking into account the variables involved in the problem and the causal dependencies among them. In the last years, however, with the advances of machine learning (ML) and ML techniques, several algorithms for inducing BNs from data have been proposed. Since in many practical situations all it is available is data, inductive learning algorithms play an important role in constructing BNs. As discussed in (Chickering 1996), learning Bayesian networks is NP-complete. BN learning can be divided into qualitative learning, focused on learning the DAG and quantitative learning, focused on the learning of conditional probabilities. Learning the BN structure is considered to be a more difficult problem than learning the BN parameters, unless the naïve Bayes method is employed, as discussed in Subsection 3.1.

As suggested in (Ben-Gal 2007), the BN learning problem can be stated informally as follows: given training data and prior information (e.g, expert knowledge, casual relationships), estimate the graph topology (network topology) and the parameters of the JPD in the BN (CPTables). One possible approach to the problem of inducing a BN from a training set is to use a scoring function to direct the search for an optimal BN in the space of possible BNs. Usual scoring functions are Bayesian scoring functions such as the one used in the K2 algorithm (Cooper and Herskovits 1992) presented in the Subsection 3.3 and others presented in (Heckerman *et al.* 1995). The function based on the minimal description length (MDL) principle (Lam and Bacchus 1993) (Suzuki 1993) is also commonly used.

Besides methods based on search-and-score, another approach, which conforms to constraint based learning, is based on conducting independence tests on the training data and construct the BN based on their results. Its main representative is the PC algorithm (Spirtes *et al.* 1993), discussed in Subsection 3.2.

5.3.1 The Naïve Bayes Classifier

In spite of its naivety, simplicity (its general DAG is always as displayed in Fig. 4) and relying on strong assumptions, the so called naïve Bayes classifier (NBC) is considered one of the most effective classifiers (see (Friedman *et al.* 1997 pg. 131) (Kohavi *et al.* 1997 pg. 79)). Langley and co-workers in (Langley *et al.* 1992) have shown that the NBC is competitive with one of the most successful ML system, the decision-tree inducer C4.5 (Quinlan 1993). The NBC assumes that:

- All other variables are conditionally independent of each other given the class variable.
- All other variables are directly dependent on the classification variable.

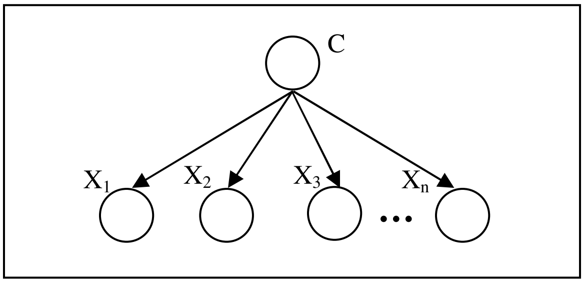

Fig. 4 The general DAG of a naïve Bayes network (classifier) where X_i's are features and C represents a class

Several NBC-based proposals attempt to achieve better performance than NBC by rewriting the assumptions. This is the case, for instance, of TAN (Tree Augmented Naïve Bayes) (Friedman and Goldszmidt 1996), SNB (Selective Naïve Bayes) (Langley and Sage 1994), BAN (Bayesian Network Augmented Naïve Bayes) (Cheng and Greiner 1999) and GBN (General Bayesian Network) (Cheng and Greiner 2001).

Since its DAG is always the same (dependent only on the number of features), the learning of a NBC consists purely in inferring, based on a given training data, the CPTables associated to each feature node, given the class label. In the classification phase, the Bayes rule is applied to compute the probability of a class label C_i given a pattern $\mathbf{X} = <X_1, X_2, \ldots, X_n>$, as show eq. (5), (6) and (7).

$$P(C_i|\mathbf{X}) = (P(\mathbf{X}|C_i) \times P(C_i))/P(\mathbf{X}) \quad \text{Bayes rule} \tag{5}$$

$$= P(X_1, X_2, \ldots, X_n|C_i) \times P(C_i) \quad \text{P(}\mathbf{X}\text{) can be removed since it is the same for all class values.} \tag{6}$$

$$= \prod_{k=1}^{n} P(X_k \mid C_i) \times P(C_i) \quad \text{Taking into account the conditional independence assumption.} \tag{7}$$

The probability given by eq. (7) is calculated for each class and the class with the largest posterior probability is assigned to the given pattern.

5.3.2 The PC Algorithm

The PC algorithm (Spirtes *et al.* 1993) starts the learning process from a complete, undirected graph (i.e., for every pair of nodes X and Y, $X \neq Y$, $\exists$ edge(X,Y)) and recursively deletes edges based on conditional independence tests, trying to identify the skeleton of the BN. The resulting structure can then be partially directed and further extended to represent the underlying DAG (Kalisch and Bühlmann 2007).

PC aims at a BN that represents the independence relationship among variables in a dataset and uses the conditional independence criteria $I(X_i,X_j|\mathbf{E})$ where **E** is a subset of variables, X_i and X_j are variables (a particular case of Definition 3, where the two first sets are singletons). If $I(X_i,X_j|\mathbf{E})$ is true, variable X_i is conditionally independent of X_j given **E** (which is verified using the d-separation criterion – see Definition 9). To verify whether X_i and X_j are conditionally independent given **E**, the cross entropy $CE(X_i,X_j|\mathbf{E})$ is computed, where the probabilities are their maximum likelihood estimators extracted from the data (i.e. relative frequencies). Other measures can also be used (Spirtes *et al.* 1993). The main steps of the PC algorithm are summarized in Fig. 5.

1. For each pair of variables, test for their conditional independence.
2. Based on the conditional independence results construct the skeleton (S) of the graph.
3. Identify the orientation of the edges in S.

Fig. 5 A high level description of the PC algorithm

Taking as input a list with all the independencies ($I(X_i,X_j|\mathbf{E})$) and adjacencies of each node (ADJ_{X_i}), PC first finds the graph skeleton (undirected graph) that best represents the d-separations expressed by $I(X_i,X_j|\mathbf{E})$ and then starts establishing the orientation of the edges.

As stated in (Spirtes and Meek 1995) "if the population, from which the sample input was drawn perfectly fits a DAG G, all of whose variables have been measured, and the population distribution P contains no conditional independence except those entailed by the factorization of P according to C, then in the large sample limit the PC algorithm produces the true pattern".

The variable preorder assumption can be used in the PC edge orientation step (step 3 of procedure described in Fig. 5) very successfully. To do that an ordered list (containing all the variable, class included) establishes that only variables that precede a given variable X_i may be parents of X_i. The use of a predefined order among variables can replace the search for the edge orientation.

The impact of variable orderings (VOs) on inducing efficient BCs was investigated in (Santos *et al.* 2007) using a genetic algorithm (GA) articulated to the PC algorithm, in a system named VOGA-PC. The role of the GA in the system was to search for a 'good' ordering among the variables – each individual (chromosome) in the GA population was a possible ordering. The class variable was not part of the chromosome; by default the class was always the first variable in any VO.

Each chromosome (i.e., each VO) was used in conjunction with the BC skeleton to induce a complete BC (skeleton + edge directions + CPTable). The BC was then input to a fitness function which, implementing a 10-fold cross validation process using training and testing sets, returned the average performance (Eval) of the BC. Based on performance results, the best chromosomes were then selected (tournament selection) and, using crossover and mutation operators, the next generation was built and the process repeated. Elitism of 1 was adopted i.e., in each generation the best ordering was kept and passed on to the next. The system VOGA-PC returns the best variable ordering (Best_VO) and the corresponding PC-induced BC. A more detailed description, results and analysis can be found in the previous cited reference.

5.3.3 The K2 Algorithm

The K2 algorithm (Cooper and Herskovits 1992) heuristically searches for the most probable BN structure given a dataset D containing n patterns and is based on four assumptions:

(1) Variables are discrete and all are observed (i.e., there are no hidden variables)
(2) Patterns occur independently, given a belief network model
(3) There are no patterns that have variables with missing values
(4) The density function $f(B_P|B_S)$ is uniform i.e., indifferent regarding the prior probabilities to place on a network structure B_S.

Considering the above assumptions, the algorithm looks for a Bayesian structure that best represents the patterns in D. The output of the K2 algorithm is a list of the parents of each node.

The variable preorder assumption is an important aspect of the algorithm. It is represented by an ordered list (containing all the variables, including the class) and asserts that variables can only be parents of variables that follow them in the list. The first variable in the list has no parents and that is why the head of the list is the class variable.

The network construction process uses a greedy method to search for the best structure. It begins as if every node has no parent. Then, beginning with the second variable of the ordered list, its possible parent candidates are evaluated and those that maximize the whole probability structure are added to the network. This process is repeated for each variable until the list finishes. It is done by maximizing the results of eq. (8).

$$P(B_S, D) = c\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}N_{ijk}! \tag{8}$$

where each discrete variable X_i ($i = 1, \ldots, n$) has r_i possible value assignments $\{v_{i_1}, v_{i_2}, \ldots, v_{i_{ri}}\}$. D is a dataset with m patterns, where each pattern contains a value assignment for each X_i ($i = 1, \ldots, n$). Let B_S be a network structure containing just the variables X_i ($i = 1, \ldots, n$). Each variable X_i ($i = 1, \ldots, n$) in B_S has a set of parents represented by the list π_i. Let w_{ij} represents the j-th unique instantiation of π_i relative to D and suppose there are q_i such unique instantiations of π_i. Let N_{ijk} be the number of patterns in D in which X_i has value v_{ik} and π_i is instantiated as w_{ij}. Let $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. Since by the fourth assumption previously stated the prior probabilities of all valid network structures are equal, $P(B_S)$ is a constant (c). Therefore, to maximize $P(B_S,D)$ requires finding the set of parents for each node that maximizes the second inner product of eq. (8).

With the best structure already defined, the network conditional probabilities must be determined. This 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 other ways to compute this probability as shown in (Cooper and Herskovits 1992).

5.4 Bayesian Classifiers in Feature Subset Selection

This section initially defines and contextualizes the feature subset selection (FSS) problem, discussing its main characteristics and impacts on machine learning processes. It also identifies a few research works related to using Bayesian formalism for solving FSS related problems. Next it approaches the solution to the problem via a particular BN-based method, describes its main contributions and then presents a few experiments and discusses their results.

5.4.1 Considerations about the Feature Subset Selection (FSS) Problem

In many real-world problems the size of a training set can be very large in both dimensions: vertically (number of training patterns) and horizontally (number of features that describe the patterns). Large numbers in both dimensions represent problems to machine learning algorithms. Vertically large datasets are generally dealt with via a technique called sampling and the horizontally large datasets are dealt with via feature subset selection methods.

The FSS problem i.e., the selection of features that play an important role in characterizing a concept has been receiving growing attention particularly in areas such as Machine Learning and Data Mining. Research in feature selection methods has intensified in application areas where datasets are usually described by tens or hundreds of thousands of features (Guyon and Elisseeff 2003). In real-world problems, relevant features are often unknown and generally many features are used to describe the training patterns in an attempt to better represent the domain. Many of these features are either partially or completely irrelevant/redundant to the concept description. Theoretically, having more features should result in more discriminating power. However, practical experience with machine learning algorithms has shown that this is not always the case.

If the available data is suitable for machine learning, then the task of inducing the concept representation can be made easier and less time consuming by removing features that are irrelevant or redundant with respect to the concept to be learnt. In a typical situation shared by many supervised machine learning methods, given a training set which generally is described as a set of training patterns, each of them represented as a vector of feature-value pairs and an associated class, a feature selection method tries to identify features that are irrelevant or redundant for describing the concept (to be learnt) embedded in the training set.

By identifying and removing irrelevant and redundant features, these methods contribute to reducing the dimensionality of the space where concepts are represented. Machine learning and data mining techniques benefit from this since a reduction in dimensionality generally promotes the accuracy and comprehensibility of the induced concepts (Nicoletti 2007). It is common to approach the FSS problem as a heuristic search in a space defined by all possible subsets of a feature set. According to this model, Blum and Langley (1997) characterize any FSS method in terms of its stance on four basic issues that determine the nature of the heuristic search process:

1) STARTING POINT – selecting a point in the feature subset space from which to begin the search can affect the direction of the search.

1.1) All features – the search begins at the state represented by all features and successively removes them.

1.2) No features – the search begins at the state represented by no features and successively adds features.

1.3) Random – the search begins at a state represented by a set of randomly selected features.

2) SEARCH ORGANIZATION – characterizes the way the search is organized. There are two basic approaches and a few variants.

2.1) Exhaustive search – it is the simplest one, which exhaustively visits all possible states. This it not a viable alternative for most problems, since the size of the search space is 2^N, for a problem defined by N features.

2.2) Heuristic search – it is a more feasible way to conduct the search for real situations. Generally, at each space state, all the local possible moves are considered, one is selected and then a new iteration is performed.

3) EVALUATION STRATEGY – the way feature subsets are evaluated is the single biggest differentiating factor among feature selection algorithms for machine learning.

3.1) Filter – based on the general characteristics of the training set to select some features and exclude others. John, Kohavi, and Pfleger (John *et al.* 1994) call these filter methods, because they filter out irrelevant features before the induction process occurs.

3.2) Wrapper – wrapper strategies for feature selection use an inductive learning algorithm to estimate the merit of feature subsets.

3.3) Embedded – the FSS is an inherent part of the ML algorithm itself and is implemented by the learning method evaluation criteria for selecting the most relevant features (e.g. information gain criteria used by ID3 (Quinlan 1986)).

The filter approach is characterized as an independent approach – an algorithm performs the reduction (hopefully) of the number of features according to a quality metric associated to features, generally based on statistical values. Filter methods conduct the process of FSS as a pre-processing step of the original training set, based on intrinsic data characteristics (such as high information contents). They are usually based on statistic techniques and are very fast. This contributes to promoting the scalability of these methods. Fig. 6 shows a general diagram of filters.

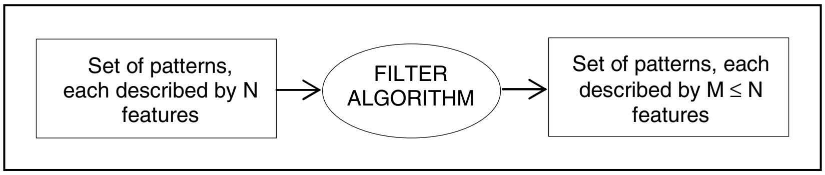
Set of patterns, each described by N features
FILTER ALGORITHM
Set of patterns, each described by M ≤ N features

Fig. 6 General scheme of a filter method for FSS

The wrapper approach works articulated to a ML method and combines a search method with a machine learning algorithm – the search is driven by the performance of the induced classifier. Fig. 7 shows a general diagram of wrappers.

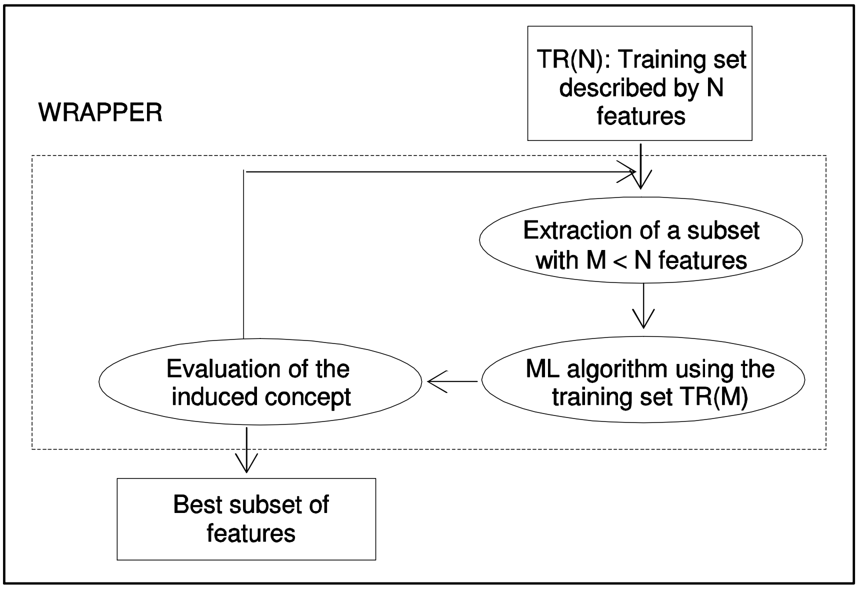
WRAPPER
TR(N): Training set described by N features
Extraction of a subset with M < N features
ML algorithm using the training set TR(M)
Evaluation of the induced concept
Best subset of features

Fig. 7 General scheme of a wrapper method for FSS

4) STOPPING CRITERION – the search for the feature subset can stop according to some pre-established criteria.

4.1) Number of feature has reached a pre-determined fixed value.

4.2) A feature selector might stop adding or removing features when none of the alternatives improves upon the merit of a current feature subset.

4.3) The algorithm might continue to revise the feature subset as long as the merit does not degrade. A further option could be to continue generating feature subsets until reaching the opposite end of the search space and then select the best.

In short, the process of finding a feature subset which allows the induction of good classifiers can be approached as a search problem in the space defined by the power set of the initial number of features. As a search problem, it can be implemented using the many available techniques, such as hill-climbing, beam search/best-first, random bit climber, Las Vegas and genetic algorithms. Reunanen in (Reunanen 2003) observes that there can be a few benefits from feature selection for learning:

- It is cheaper to measure only a subset of variables;
- Prediction accuracy might be improved through exclusion of irrelevant variables;
- The predictor to be built is usually simpler and potentially faster when less input variables are used;
- Knowing which variables are relevant can give insight into the nature of the prediction problem at hand.

There have been a few proposals to applying BN-based methods to the FSS problem, such as the hybrid method described in (Inza *et al.* 2001) and the work in (Inza *et al.* 2000). Antal and colleagues in (Antal *et al.* 2008) discuss applications of the Bayesian approach to new challenges in relevance analysis, which can be seen as a continuation of their work described in (Antal *et al.* 2006), where the generalizations of the FSS problem in a Bayesian framework based on the structural properties of BNs is formulated.

Fu and Desmarais in (Fu and Desmarais 2010) provide a review on related works on FSS based on the induction of the MB; Zeng and co-workers (Zeng *et al.* 2009) also used the concept of MB for filtering features and use the reduced feature set for learning. Brown and Tsamardinos in (Brown and Tsamardinos 2008) describe a new filter algorithm called Feature Space Markov Blanket (FSMB) which combines ideas borrowed from both, kernel and Markov Blanket based feature selection.

Authors in (Koller and Sahami 1996) have shown that the Markov Blanket criterion only removes features that are really unnecessary. They propose a heuristic approach for dealing with this problem but acknowledge that their algorithm performance can be improved by using more refined techniques, such as BNs, to choose candidate MBs. They also observe that finding an exact or an approximate MB can be a hard task. The proposal described in Subsection 4.2 is similar to the one described in (Cheng *et al.* 1997). Their main difference is that the work in (Cheng *et al.* 1997) uses a Conditional Independence Learning method and the method described in Subsection 4.2 uses a heuristic search based learning algorithm.

5.4.2 Feature Subset Selection by Bayesian Networks – The K2χ^2 Method

This section condenses part of the work described in (Hruschka Jr. *et al.* 2004), (Santos *et al.* 2007), (Hruschka Jr. and Ebecken 2002) where a BN-based feature selection method specifically designed for classification problems is proposed and evaluated. The method can be characterized as filter and basically (1) creates a Bayesian network from a training set and then (2) uses the Markov Blanket of the class variable as the set of relevant features for the corresponding classification problem. Fig. 8 shows the general flowchart of the method.

In order to create the Bayesian network (or classifier) from data, the variant K2χ^2 that combines the K2 algorithm (see Section 3) with the χ^2 statistic test was used. The χ^2 was employed aiming at optimizing the variable ordering to be used by the K2 algorithm, since this statistics test can be used to assess the independence of two variables (Liu and Motoda 1998). The χ^2 was used to measure the degree of dependence between the class variable and each of the variables describing the training set. The variables were then listed in descending order of their χ^2 result and the information was passed on to K2. As can be seen in (Hruschka Jr. and Ebecken 2007), the only difference between K2 and K2χ^2 is the use, by the later, of the information given by the variable ordering to induce the BC.

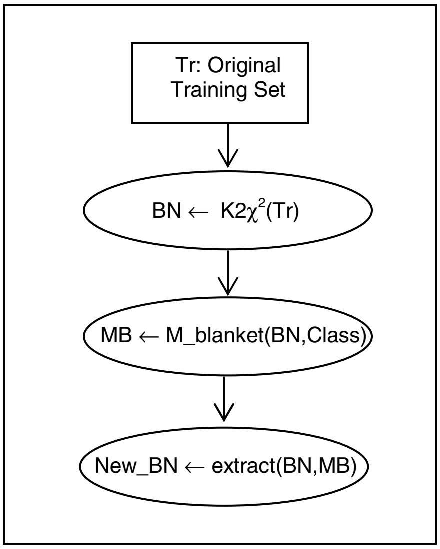

Fig. 8 Flowchart of the BN K2χ^2 inducer for feature selection (given by the Markov Blanket of the variable class)

Experiments were conducted using three knowledge domains from the UCI-Irvine Repository (Frank and Asuncion 2010), whose characteristics are presented in Table 1.

Table 1 Domain characteristics. The three domains have 2 classes. ONP/NP: original number of patterns/number of patterns, NP/C: number of patterns per class

Domain	ONP/NP	NP/C
Wisconsin Breast Cancer (WBC)	699/683	444/1 (benign) 239/2 (malignant)
Mushroom	8124/5644	1728/1 (edible) 3916/2 (inedible)
Congressional Voting Records (CVR)	435/232	124/1 (democrat) 108/2 (republican)

In the Wisconsin Breast Cancer (WBC) dataset the two classes are known to be linearly inseparable. The total number of features is 10 (including the class). The total number of patterns in the original WBC is 699; however 16 of them have missing feature values and were removed from the training data. The total number of patterns in the original Mushroom dataset is 8,124, each described by 22 features. However, 2,480 patterns whose eleventh feature was missing were removed and the remaining 5644 patterns were used in the experiments. The original Congressional Voting Records (CVR) dataset is described by 16 Boolean features and has 435 patterns, divided into 267 democrat and 168 republican. However, 203 patterns have missing values and were removed; the remaining 232 (124 democrat, 108 republican) were used in the experiments.

Aiming at identifying the influence of the variable ordering on the induced BCs, the correct classification rates (CCR) by both, K2 and K2χ^2 in the three knowledge domains, are shown in Table 2. In the table accuracy numbers refer to the average of a five-fold cross-validation process and μ and σ stand for average and standard deviation respectively. Simulations have shown that the ACCR (average correct classification rates) obtained using the training data with the original sequence of features and with the sequence given by the feature ordering were very close, leading to consistent results.

Table 2 ACCR of K2 *versus* K2χ^2. μ: average, σ: standard deviation

Dataset	Class	K2 (original variable ordering)		K2χ^2	
		μ	σ	μ	σ
WBC	1	96.84	1.66	96.61	1.58
	2	95.82	1.45	97.08	2.37
	Total	**96.48**	1.40	**96.78**	1.32
Mushroom	1	95.09	2.49	77.22	1.30
	2	5.92	3.23	87.93	1.90
	Total	**61.03**	0.72	**81.22**	0.63
CVR	1	63.33	28.49	96.0	5.65
	2	13.81	8.39	86.08	5.60
	Total	**40.17**	15.65	**91.40**	3.62

In Table 3 the selected set of features (i.e., the MB of the corresponding class feature), for each domain, is presented.

Table 3 |OF|: number of original features (class excluded), SF: selected feature set (class excluded)/|SF|

Domain	|OF|	SF/|SF|
WBC	9	{x2, x3, x4, x6, x7, x8}/6
Mushroom	22	{x3, x5, x9}/3
CVR	16	{x3, x4, x5, x12}/4

To verify the consistency of the generated networks and to provide evidence that the selected features are relevant to the model, classification tasks were performed (a) using the original features and (b) using the selected features. Results are summarized in Table 4. In all learning tasks a five-fold cross-validation process was applied.

Results from the just described FSS approach were compared against results given by classifiers induced by three different algorithms, using the original training set (all features present). The classifiers were (1) a Bayesian classifier induced by the Naïve Bayes method; (2) a decision tree induced by the C4.5 algorithm (Quinlan 1993) in its version available at the WEKA System, identified as J48 (Witten and Frank 2000); (3) a set of rules obtained by J48 PART (Witten and Frank 2000), a method that extracts rules from pruned partial decision trees (also built using the C4.5 algorithm).

Table 4 BC results (%) per domain

Class	Original Features		Selected Features	
	μ	σ	μ	σ
		WBC		
1	96.61	1.58	96.61	1.58
2	97.08	2.37	95.40	2.70
Total	**96.78**	**1.32**	**96.19**	**1.19**
		Mushroom		
1	77.22	0.70	94.61	1.03
2	87.93	0.59	96.25	1.90
Total	**81.22**	**0.41**	**95.11**	**0.63**
		CVR		
1	96.00	5.65	96.00	5.65
2	86.08	5.60	85.08	6.25
Total	**91.40**	**3.62**	**90.95**	**3.48**

As mentioned before, there are three main classes of algorithms for learning Bayesian networks. One refers to algorithms based on heuristic search, the second to algorithms based on the use the conditional independence concept and the third to algorithms that combine both previous strategies.

When using algorithms based on heuristic search, one important issue is the initial order by which features are presented to the algorithms. Algorithms use this information to determine the direction of arcs – a variable is a possible parent only of those that follow it in the ordering (Hruschka Jr. and Ebecken 2002). Conditional independence methods try to find the direction of arcs without the information given by the variable ordering. It has been reported, however, that algorithms have an improved performance when the ordering is provided (Spirtes *et al.* 1993).

Results achieved by each of the three other classifiers in the three domains are presented in tables 5, 6 and 7 respectively. In the three tables O and S stand for 'Original' (all features in the original dataset) and 'Selected' (selected features given by the MB) respectively.

The Naïve Bayes method uses all features and allows them to make contributions to the decision that are equally important and independent of one another, given the class. This leads to a simple scheme that works well in practice (Witten and Frank 2000). One can observe that the proposed method selects very predictive features, which in Mushroom and Congress provide even better ACCRs than those achieved in the dataset formed by all features. Table 5 shows the results obtained in complete datasets and in datasets described by the selected set of features with the Naïve Bayes.

Table 5 Naïve Bayes – Average Classification Rates (%).

O: original features, S: selected features (MB)

Dataset	Total	Class 1	Class 2
WBC O	96.49	95.70	97.90
WBC S	95.90	95.70	96.20
Mushroom O	97.36	99.60	93.80
Mushroom S	99.22	100.00	98.00
CVR O	91.40	88.70	94.40
CVR S	93.10	91.10	95.40

The J48 algorithm is the WEKA´s implementation of the popular C4.5 (Quinlan 1993). In fact, J48 is a C4.5 improved version, called revision 8 (Witten and Frank 2000). Table 6 shows the simulation results using J48. The table also shows the set of features present in the best induced classifier obtained in the five-fold cross validation process. It can be noticed that all features selected by the Bayesian approach were employed both in the Mushroom and WBC domains; in the CVR domain, however, the feature selection process was not so important, since only one feature is necessary to classify all examples. Besides, in the simulations the selection method was consistent in the context of J48, and consequently with the information gain criterion.

Table 6 J48 – Average Correct Classification Rates (%).

O: original features, S: selected features (MB)

Dataset	Features in the best classifier induced	Total	Class 1	Class 2
WBC O	{x1,x2,x3,x4,x5,x6,x7}	95.17	96.40	92.90
WBC S	{x2,x3,x4,x6,x7,x8}	95.61	95.00	96.70
Mushroom O	{x5,x18,x17,x8,x4,x20}	100.00	100.00	100.00
Mushroom S	{x3,x5,x9}	99.86	99.80	100.00
CVR O	{x4}	95.26	95.20	95.40
CVR S	{x4}	96.98	95.20	99.10

The J48 PART extracts rules from pruned partial decision trees built using C4.5; it combines the divide-and-conquer strategy for decision trees learning with the separate-and-conquer for rule learning (Witten and Frank 2000). To make a single rule, a pruned decision tree is built based on the current set of patterns and then the path to the leaf with the largest coverage is made into a rule and the tree is discarded. Table 7 shows the obtained results with J48 PART. Again, all the selected features were employed both in Mushroom and in WBC datasets, whereas in CVR only one of the selected features was enough to classify all examples. The results are consistent with those in Table 6, indicating that the proposed method is a good option as a FSS method, allowing the extraction of simple rules with very good ACCRs.

Table 7 J48 PART – Average Correct Classification Rates (%).

O: original features, S: selected features (MB)

Dataset	Features in the best classifier induced	Total	Class 1	Class 2
WBC O	{x1,x2,x3,x4,x5,x6,x7,x8}	96.05	97.50	93.30
WBC S	{x2,x3,x4,x6,x7,x8}	95.31	95.90	94.10
Mushroom O	{x5,x8,x11,x17,x18,x20,x21}	100.00	100.00	100.00
Mushroom S	{x3,x5,x9}	99.86	99.80	100.00
CVR O	{x2,x3,x4,x9,x11}	94.40	95.20	93.50
CVR S (4)	{x4}	96.98	95.20	99.10

The main simulation results are condensed in Table 8, where the total ACCR values as well as the number of features employed per domain per classifier are presented. In general, the ACCRs obtained using as training data the datasets described by the original features were very close to those obtained using datasets described by the selected features. It is noticeable the significant improvements in relation to the number of employed features. In the WBC, 66.67% of the original features were enough to obtain high classification rates. A similar effect was observed in the Mushroom, where only 13.64% of the original number of features was selected. It is noticeable that the Bayesian network accuracy improved significantly. Finally, in the CVR, by using a specific 25% of the original number of features the classification rate still was high. Another important aspect is that the ACCR values obtained by the classifiers (using original or selected subset of features) are comparable to the best ones found in the literature. Duch and colleagues in (Duch *et al.* 2000), for instance, describe classification results obtained by 15 methods. Their results, in the Mushroom dataset, vary from 91% to 100%, while in the WBC they vary from 92.7% to 99%. The CVR dataset information file reports accuracies that vary from 90% to 95% (Schllimmer 1987).

Table 8 Main simulation results: Average Correct Classification Rates (%).

O: original set of features, S: selected set of features

Classifier	WBC		Mushroom		CVR	
	O (9)	S (6)	O (22)	S (3)	O (16)	S (4)
BN	96.78	96.19	81.22	95.11	91.40	90.95
Naïve Bayes	96.49	95.90	97.36	99.22	91.40	93.10
J48	95.17	95.61	100.00	99.86	95.26	96.98
J48 PART	96.05	95.31	100.00	99.86	94.40	96.98

5.5 Bayesian Classifiers in Imputation Processes

This section initially defines and contextualizes imputation processes, discussing its main characteristics, uses and impacts on machine learning processes. Next it describes the proposal of a BN-based imputation process and its main contributions.

5.5.1 Considerations about Imputation Processes

The absence of information is common in real-world databases and it can occur due to a number of reasons, such as malfunctioning measurement equipment, changes in experimental design during data collection, collation of several similar but not identical datasets, refusal of some respondents to answer certain questions in surveys, etc. Such missing data are usually problematic. Therefore, several approaches have been proposed to deal with them as can be seen in (Rubin 1976), (Rubin 1987), (Little and Rubin 1987), (Pyle 1999) and (Schafer 2000). A simple approach to deal with missing values ignores patterns and/or features containing missing values; the loss of data however can be considerable and reduced datasets may lead to biased statistical analyses. Alternatively, some approaches for data analysis (e.g. (Breiman *et al.* 1983), (Quinlan 1993)) can be tolerant to missing values. Finally, a significant number of machine learning methods work only with complete datasets. For these methods, approaches aimed at filling in missing values are particularly relevant.

The task of filling in missing data is often referred to as missing values substitution or imputation and it can be performed in a number of ways such as by the widely used naïve mean/mode method. The substitution of missing values by the mean/mode can eventually lead to reasonable results. This procedure, however, assumes that all missing values represent the same value, possibly leading to considerable distortions. The mean/mode method underestimates the population variance and does not take into account the relationships between features, which are usually relevant to the process of missing values replacement. Moreover, machine learning methods usually explore relationships between features and, thus, it is critical to preserve them, as far as possible, when replacing missing values (Pyle 1999). In this sense, imputation is aimed at carefully substituting missing values, trying to avoid the insertion of bias in the dataset. If imputation is performed in a suitable way, higher quality data might becomes available and results from machine learning tasks can be improved.

Techniques to deal with missing values have already been studied for many years (e.g. see (Anderson 1946), (Preece 1971), (Dempster *et al.* 1977), (Rubin 1977), (Rubin 1987), (Schafer 2000)). Although most of these techniques have been applied to survey data analysis, they can also be useful for machine learning applications. Therefore, before focusing on the specific imputation method described in (Hruschka Jr. *et al.* 2007), a brief survey on imputation methods is presented next.

5.5.2 Commonly Used Imputation Methods

The expectation-maximisation (EM) algorithm (Dempster *et al.* 1977), (Redner and Walker 1984), (Wu 1983), (Ghahramami and Jordan 1995), (Jordan and Xu 1996), (Bilmes 1997) has been widely used for imputation. EM assumes that missing data (Y) are governed by a distribution $f(Y|X,\theta)$, where X (data without missing values) and the parameters θ (mean and variance) are fixed. The EM algorithm is based on the likelihood function, and it fills in the missing data based on an initial estimate of θ. Then, it 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 (Little and Rubin 1987). In addition, the computations performed by EM are dependent on the assumption of a particular density function and its parameters.

Multiple imputation (MI) (Rubin 1977) has been widely used for multivariate analysis, and it consists in using more than one value to fill in the gaps in the sample (e.g. the mean of probable values). MI can provide good results, but the involved computational cost is considerably higher when compared to single imputations (Rubin 1987).

Data augmentation (DA) (Tanner and Wong 1987) can be informally described as the process in which observed data Y (whose distribution depends on the parameters θ) is augmented by the quantity Z (using a Monte Carlo sampling strategy). Based on the MI idea, multiple values for Z can be generated using the $p(Z|Y)$ distribution and then obtaining $p(\theta|Y)$ as the average of $p(\theta|Y,Z)$ over the imputed Zs. In theory, this method provides a way to improve the inference in small samples, a situation where EM has pitfalls.

Considering decision trees, some practical results about ignoring patterns with missing values can be found in Quinlan (1986) and White (1987). Another approach involves replacing missing values with the most frequent value (Kononenko *et al.* 1984). In the probability method (Quinlan 1989), (Lobo and Noneao 2000), a decision tree is constructed to substitute missing values of each feature, using the information contained in the other features. The dynamic path generation (Quinlan 1986) and the lazy decision tree approach (Friedman *et al.* 1996) do not generate the whole tree, but only the most promising path instead.

Several imputation methods based on nearest neighbours can be found in the literature. They basically select patterns with feature values similar to the pattern of interest to impute missing values. For instance, see (Troyanskaya *et al.* 2001), (Batista and Monard 2003) and (Hruschka *et al.* 2003).

According to Schaffer and Graham (2002), maximum likelihood methods (e.g. EM and Bayesian algorithms) represent the state of the art for imputation. Considering high-dimensional datasets, BNs are usually more efficient than methods based on the EM algorithm (Zio *et al.* 2004). Zio and co-workers describe the use of BNs for imputing missing values, arguing that two relevant advantages of using BNs as imputation models are the possibility of preserving statistical relationships between variables, and dealing with high-dimensional datasets.

5.5.3 Imputation by Bayesian Networks and the K2Iχ2 Method

There have been a few attempts towards imputation processes articulated to Bayesian networks, such as the proposal described in (Kong *et al.* 1994). Zio and colleagues in (Zio *et al.* 2004) discuss the use of BNs for imputation aiming at dealing with the problem of the consistency of imputed values: preservation of statistical relationships between variables (statistical consistency) and preservation of logical constraints in data (logical consistency).

In order to tackle the missing value problem in classification tasks, the K2Iχ^2 method was proposed to impute (substitute) missing values based on Bayesian networks as described in (Hruschka Jr. *et al.* 2007).

K2Iχ^2 relies on the construction of a BN to infer the most suitable values to fill in the gaps produced by missing values. K2χ^2 learning algorithm (as described in Section 4.2) is applied to construct a BN to be used as a prediction model to substitute the missing values. Instead of generating one BN for each feature with missing values, as described in (Hruschka Jr. and Ebecken 2002), K2Iχ^2 builds a single BN to infer the best values to substitute the missing ones in all features. The unrestricted BN is therefore used considering all variables as potential predictors.

The imputation process performed by K2Iχ^2 can be summarised by the following steps: (i) generate a single clean (i.e., without missing values) training dataset C; (ii) build an unrestricted BN' using C and (iii) use BN' to infer the best values to replace the missing ones.

In (Hruschka Jr. *et al.* 2007) K2Iχ^2 is evaluated in the context of both prediction and classification tasks, and its performance is compared with those obtained by classical imputation methods (EM, Data Augmentation, Decision Trees, and Mean/Mode). The simulations were performed on four UCI (Frank and Asuncion 2010) datasets (Congressional Voting Records, Mushroom, Wisconsin Breast Cancer and Adult), which are benchmarks for data mining methods. Missing values were simulated by the elimination of some known values. Thus, it was possible to compare original values with imputed ones, evaluating the prediction capability of the imputation methods. In addition, a methodology to estimate the bias inserted by imputation methods in classification tasks is proposed. Four classifiers (One Rule, Naïve Bayes, J48 and J48 PART) were used to evaluate the five imputation methods in classification scenarios. Computing times consumed to perform imputations were also reported for each imputation method. Simulation results in terms of prediction, classification and computing times allow to perform several analyses, leading to interesting conclusions. K2Iχ^2 has shown to be competitive against classical imputation methods and achieved good results when variable relationships as well as the imputation bias were taken into account. More discussions on imputation bias and BN imputation methods can be found in (Hruschka *et al.* 2009).

5.6 Post-processing a Bayesian Classifier into a Set of Rules

This section describes in detail how BNs can be post-processed in order to create a set of rules. The main motivation for post-processing a BC into a set of rules is for the sake of understandability. Many automatic learning tasks are used in real data domains which, generally, are described by a large number of features; in such domains the induced classifiers tend to be large and complex and usually, hard to be completely understood by humans.

It is a fact that knowledge represented by BCs is not as comprehensible as knowledge represented by some other forms such that of classification rules. This can be a drawback in areas where the understandability of the representation plays a major role. Reasoning with rules is comprehensible, provides explanations, and may be validated by human inspection. It can also increase the user's confidence in the system and eventually can help to discover important relationships among variables.

The BayesRule method described in this section, originally proposed in (Hruschka Jr. *et al.* 2008), translates a learnt BC into a set of classification rules, a much more suitable knowledge representation for promoting understandability. The experiments show that the reduced set of rules extracted from a BC can be reasonably condensed and still maintain the original BC classification accuracy.

Subsection 6.1 addresses the description of the BayesRule algorithm. The experimental results, described in Section 6.2, show that it is possible to extract a reduced number of simple rules from a BC and, thus, circumvent the dimensionality problem without the use of complex procedures of optimization and pruning.

5.6.1 Translating a Bayesian Classifier into a Reduced Set of Rules – The BayesRule Algorithm

As discussed in Section 2 and proved in (Pearl 1988), the only nodes that have influence on the conditional probability distribution of a given node X (given the state of all the remaining nodes of a BC) are the nodes that belong to the Markov Blanket of X. Thus, after learning a BC from data, the Markov Blanket of the class node identifies, among all nodes that define the BC those that influence on the class node. In the BayesRule method the MB concept is used to reduce both the number as well as the complexity (in relation to the number of variable tests in the antecedent) of classification rules. When generating the set of propositional classification rules, the only variables taken into account are those in the MB of the class variable.

As standard propositional if-then classification rule is the simplest and the most comprehensible way to represent classification knowledge it has been adopted by BayesRule. The BayesRule method is based on the intuition that the best explanation for a piece of evidence is the most probable state of the world, given the evidence. This approach is called maximum a posteriori (MAP) approach (see (Henrion and Druzdzel 1990) and (Pearl 1988) for details).

MAP is a standard approach to parameter estimation and inference in statistics. When concerning classification tasks, many algorithms consider some candidate classes (or hypothesis) $\{C_1, C_2, \ldots, C_j\}$ and try to identify the one that best fits a given background (BK) knowledge. The choice of the best class is often based on the most probable class C_J ($J = 1, \ldots, j$) given the BK. Any such maximally probable class is called Maximum a Posteriori (MAP) class (C_{MAP}). As proved in (Mitchell 1997), a BC, as well as any other classifier based on the Bayes Theorem, can be used to calculate the posterior probability of each candidate hypothesis as follows:

$$C_{MAP} = C_{J_k} \text{ such that } P(BK \mid C_{J_k})P(C_{J_k}) = \max_{J \in \{1,\ldots,j\}} P(BK \mid C_J)P(C_J) \qquad (9)$$

In the particular situation where all classes C_J are equally probable *a priori* (i.e. $P(C_m) = P(C_n)$, $\forall$ m, n $\in$ $\{1, 2, \ldots, j\}$, $m \neq n$), the term $P(C_J)$ can be removed from eq. (9) and C_{MAP} is renamed as Maximum Likelihood (ML) class (C_{ML}).

When using the MAP approach to extract rules from a BC, one rule is created for each possible value of the involved variables, a computationally expensive procedure. This happens mainly because it is very common the presence of hundreds or thousands of variables in probabilistic models (Druzdzel 1996). In most cases, however, many variables may only be relevant for some types of reasoning; very rarely all of them will be relevant in the reasoning process associated to one single query, for instance. Therefore, it is essential to focus only on the relevant part of the model (i.e. the class variable and the variables that belong to its Markov Blanket) when translating it into a set of rules. In this sense, the proposed BayesRule method uses the Markov Blanket concept to select the variables that will be in the antecedent of rules. Thus both, the number and the complexity of rules are minimized along with the rule extraction process. The variable selection strategy, however, does not guarantee a minimal rule set.

The extracted rule set can still undergo a pruning step which will remove from the rule set the rules containing superfluous conditions.

In accordance with the MAP approach, a BC evidence propagation algorithm must be used to propagate the variable values aiming at inferring the class value. Let A be a set of variables that are instantiated and B a set of their corresponding values. An evidence propagation algorithm determines $P(V_{i,J_i} | A,B)$ for all values V_{i,J_i} of all variables in the network except those in A. For singly-connected (only one edge is allowed between two nodes) BNs there are simple and efficient evidence propagation algorithms; when the BN structure is a multiply-connected graph, however, the evidence propagation process is, in the worst case, NP-hard (Cooper 1990).

It is agreed that the most popular exact BN inference algorithm is Lauritzen and Spiegelhalter's clique-tree propagation algorithm (Lauritzen and Spiegelhalter 1988). Their algorithm is based on Pearl's polytree propagation algorithm (Pearl 1988). However, considering that Pearl's algorithm can only be used when the BN structure is a polytree, the clique-tree propagation algorithm first transforms a multiply-connected network into a clique tree by clustering the triangulated moral graph of the underlying undirected graph, then performs message propagation over the clique tree using the classic polytree algorithm.

Pearl's polytree algorithm exploits the fact that a polytree is a singly-connected structure and consequently, there is only one path between two nodes; this special characteristic allows only one choice for transmitting the evidence in the BN, i.e., there is no risk of redundant propagation. The polytree propagation algorithm may be summarized as follows: each node (variable) exchanges messages with its parents and its children. Messages sent from parents to children are called π-messages and messages sent from children to parents are called λ-messages. In a BN, for each new evidence impacting on a node X, this node must update its own CPTable and propagate the new values to all its parents nodes $\pi_{(X)}$ and to all its children nodes $\lambda_{(X)}$. Once a node U (parent of node X) receives the new probability values from X, it must update its own CPTable and propagate the updated values to all its parents nodes $\pi_{(U)}$ and to all (except X) its children nodes $\lambda_{(U)}$. Once a node Y (child of node X) receives the new probability values from X, the same updating process must be repeated. In this sense, Y must update its own CPTable, and propagate its new probability values to all (except X) its parents nodes $\pi_{(Y)}$ and to all its children nodes $\lambda_{(Y)}$. According to the updating procedure, the new evidence impacted on X will be propagated to all nodes in the BN without redundancy.

As previously described, when a node receives new evidence, or a message from its parents or children, it must update its own CPTable. This is done simply by multiplying its probability estimation matrix (CPTable) by the probability estimation vector received from its parents (or children).

The BayeRule implements the evidence propagation algorithm proposed by Lauritzen and Spiegelhalter because it is an efficient algorithm even when applied to BNs consisting of very large number of variables. Fig. 9 presents a simplified version of the pseudocode of the BayesRule method; the procedure expects as input a Bayesian Classifier with N nodes and assumes by default that the class variable is X_1.

```
procedure BayesRule;
input     BC: Bayesian Classifier with N nodes
          $X_1$: Class variable
output: RSR {Reduced Set of Rules}

begin

 1.  RSR ← ∅ {reduced set of rules is empty}
 2.  CMB ← MB($X_1$) {Markov Blanket of $X_1$ (class variable)}
 3.  M ← |CMB|
 4.  Rename the variables in CMB as $X_2$, $X_3$,…, $X_{M+1}$
 5.  for i ← 2 to M+1 do
 6.  begin
 7.    $V_i$ ← set of possible values of variable $X_i$
 8.    $j_i$ ← $|V_i|$
 9.  end
 10. RI ← 1   {rule index}
 11. for $k_2$ ← 1 to $j_2$ do
 12.   for $k_3$ ← 1 to $j_3$ do
 13.     ......................
 14.       for $k_{M+1}$ ← 1 to $j_{M+1}$ do
 15.       begin
 16.         Rule_antecedent ← $X_2 = v_{2_{k_2}}$ and $X_3 = v_{3_{k_3}}$ and … and $X_{M+1} = v_{N_{k_{M+1}}}$
 17.           • propagate Rule_antecedent throughout BC and determine
                 the class value Val_Class and certainty factor F
 18.           • define rule $R_{RI}$ as: if Rule_antecedent then $X_1$ =
               Val_Class(F%)
 19.         RSR ← RSR ∪ {$R_{RI}$}
 20.         RI ← RI + 1
 21.       end
 22. RSR ← remove_irrelevant_rules(RSR)
end
```

Fig. 9 Pseudocode of the BayesRule algorithm

The BayesRule procedure for extracting classification rules from BCs is quite simple. As mentioned before, the BC structure provides a simple and efficient mechanism (Markov Blanket) to reduce the number and the complexity of the rule set. The procedure described in Fig. 9 creates probabilistic rules as:

If < *condition* > then < *class* > with certainty F

where *condition* is the antecedent part of the rule, *class* the consequent and F is a percentage value.

Let $V_1, V_2, \ldots, V_n$, C be the sets of variable values for $X_1, X_2, \ldots, X_n$ and C, respectively. Also, let $|V_i| = j_i$, $i = 1, \ldots, n$ and $|C| = j$. A probabilistic if-then rule can be characterized as:

$$\text{If } X_1 \text{ is } V_{1,J_1} \text{ and } \ldots \text{ and } X_n \text{ is } V_{n,J_n} \text{ then C is } C_J \text{ (F\%)}$$

where $J_i \in \{1, \ldots, j_i\}$, $i = 1, \ldots, n$ and $J \in \{1, \ldots, j\}$.

By using the BayesRule method, the number of variables involved in the antecedent of a rule is reduced since the method only considers the Markov Blanket of the class variable C. Considering a particular situation where the Markov Blanket of the class variable C is the set $\{X_1, \ldots, X_k\}$, the *a posteriori* probability of class C = C_j given the values of the variables in the Markov Blanket of class C for a particular instantiation of indexes J_i, i = 1, …, k is

$$P(C = C_j \mid V_{1,J_1}, \ldots, V_{k,J_k}) = \max_{J \in \{1,\cdots,j\}}\{P(C = C_J \mid V_{1,J_1}, \ldots, V_{k,J_k})\}$$

where each $P(C = C_J \mid V_{1,J_1}, \ldots, V_{k,J_k})$ is calculated using eq.(4).

The confidence degree associated to a rule can be defined using inferential results. In doing so, the probability given to the inferred class may be used as a confidence value and it is embedded in the inference algorithm. The rule coverage can be obtained from the numerical parameters (CPTables) already stored in the BC, and consequently no extra computation is needed for defining it.

As an example of how the BayesRule procedure works, consider a BN with 5 nodes and 5 arcs, as depicted in Fig. 10 (borrowed from (Cooper 1984)), referred to as Example_BN. Consider also that all nodes (variables) are binary (present/absent) and the class variable is CA.

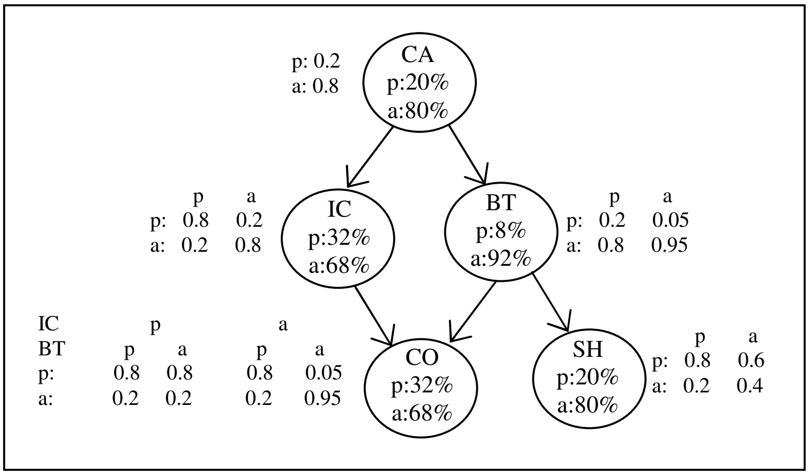

Fig. 10 Example_BN: an example for explaining how the rule extraction process conducted by BayesRule works

As can easily be seen in Fig. 10, MB(CA) = {IC, BT}. Since the step 4 of BayesRule procedure renames variables, CA, IC and BT as X_1, X_2 and X_3 respectively. Considering that the two variables (X_2 and X_3) defining the antecedent part of the rules are binary, four rules will be created. The Lauritzen and Spigelhalter propagation algorithm (Lauritzen and Spigelhalter 1988) was used to determine the value of X_1 for the possible combinations of X_2 and X_3. In the example the four extracted rules shown in Fig. 11 define the final RSR (reduced set of rules) created by BayesRule.

R_1: if X_2 = present and X_3 = present then X_1 = present (80%)
R_2: if X_2 = present and X_3 = absent then X_1 = absent (54%)
R_3: if X_2 = absent and X_3 = present then X_1 = absent (80%)
R_4: if X_2 = absent and X_3 = absent then X_1 = absent (95%)

Fig. 11 Reduced Set of Rules (RSR) extracted by BayesRule from the BN shown in Fig. 10

It is worth remembering that the number of rules has a significant impact on the system accuracy as well as on its understandability as a motivation for the introduction of the last step of the BayesRule procedure. While a high number of rules may improve classification accuracy, it may also disrupt understandability. It is also well known that rules with too many conditions in their antecedent parts are more difficult to understand than those with a lesser number of conditions. Taking into account both issues, the last step of BayesRule prunes the RSR particularly when the set has a large number of long rules. When having a small RSR, however, the pruning step may not be applied. As described in line 22 of the BayesRule algorithm, a pruning step may be applied to the final RSR. Considering the RSR extracted from Example_BN (Fig. 10) the pruning step would be skipped (as the RSR has only four short rules). Nevertheless, to illustrate the *remove_irrelevant_rules(RSR)* procedure consider applying it to the RSR shown in Fig. 11. A careful look at rules R_3 and R_4 reveals that when X_2 = absent, X_1 is always classified as absent (X_3 value has no influence in the class definition in such a situation). Thus, the *remove_irrelevant_rules(RSR)* replaces rules R_3 and R_4 by a new rule defined as: "if X_2 = absent then X_1 = absent (96%)". As rules R_3 and R_4 were removed, the new rule is named R_3 and a more reduced set of rules is generated as depicted in Fig. 12.

R_1: if X_2 = present and X_3 = present then X_1 = present (80%)
R_2: if X_2 = present and X_3 = absent then X_1 = absent (54%)
R_3: if X_2 = absent then X_1 = absent (96%)

Fig. 12 Reduced Set of Rules (RSR) extracted by BayesRule using a naïve pruning strategy from the BN shown in Fig. 10

In addition, two important issues should be taken into consideration in the pruning process. The first one is that the probability estimation (96%) of the new R_3 rule was obtained running the Lauritzen and Spiegelhalter clique tree algorithm (considering X_2 = absent as the only evidence to be propagated). The second issue is that this pruning strategy is very simple and more elaborated techniques should be investigated in further implementations. The experiments described in the next subsection show that in addition to produce a reduced set of rules, BayesRule can still maintain a good classification performance.

5.6.2 Using BayesRule - Experiments and Results

In order to empirically validate BayesRule, a few experiments were conducted. Considering that BayesRule expects as input a Bayesian Network and a class variable the experiments were based on well-known BNs. The two main advantages of working with a well-known BN are (1) as it is possible to know *a priori* the real dataset probability distribution and its characteristics, the results obtained in the experiments can be analyzed in a more consistent and reliable way; (2) it is possible to inspect the behavior of BayesRule in specific situations of interest.

Five well-known BNs namely Alarm (Beinlich *et al.* 1989), Asia (Lauritzen and Spiegelhalter 1988), Credit (Druzdzel 1996), Engine Fuel System (Engine) (Druzdzel 1996), Win95pts (Horvitz *et al.* 1998) and two artificially created BNs, referred to as Synthetic 1 (Syn_1) and Synthetic 2 (Syn_2), were employed in the experiments. Table 9 summarizes the dataset characteristics.

Table 9 Dataset description where AT: number of features plus class, IN: number of patterns and Cl: number of classes

	Alarm	Asia	Credit	Engine	Win95pts	Syn_1	Syn_ 2
AT	38	8	12	9	76	32	32
IN	10^4	10^4	10^4	10^4	10^4	10^4	10^4
Cl	See Table 10	2	2	2	See Table 10	2	2

Table 10 Alarm and Win95pts variable class names and their respective domain sizes

Alarm		Win95pts	
Class name	|Class|	Class name	|Class|
Anaphylaxis	2	Repeatable Problem	2
Intubation	3	Driver File Status	2
KinkedTube	2		
Disconnect	2		
Hypovolemia	2		
InsuffAnesth	2		
LVFailure	2		
PulmEmboulus	2		

The Alarm BN, a network for monitoring patients in intensive care, is based on expert knowledge and was originally described in (Beinlich *et al.* 1989). It is defined by 37 variables and 46 arcs and represents 8 diagnostic variables, 16 measurements, and 13 intermediate variables that connect diagnostic problems to findings. The diagnostic variables have no predecessors and are assumed to be mutually independent *a priori*. These variables represent the presence, absence or the severity of a particular disease. The measurement variables represent quantitative information available when a patient is being monitored. The intermediate variables cannot be measured and, thus, are inferred using the available information. In the conducted experiments, the Alarm BN was used as a BC. The eight Alarm diagnostic variables *Hypovolemia, LVFailure, Anaphylaxis, Insufficient Anesthesia, PulmEmboulus, Intubation, KikedTube* and *Disconnect* were used for classification purposes (i.e., were considered one at a time, as the class variable). The experiments also had the purpose of exploring the BC ability to try different class variables using the same model (instead of building a customized model for each variable). Fig. 13 shows the structure of the Alarm BN. Due to the reduced dimensions of this figure, variable names have been replaced by numbers following the convention described in (Cooper and Herskovits 1992). The variables originally named *Hypovolemia, LVFailure, Anaphylaxis, Insufficient Anesthesia, PulmEmboulus, Intubation, KikedTube* and *Disconnect* are represented by numbers 17, 18, 19, 20, 21, 22, 23 and 24 respectively.

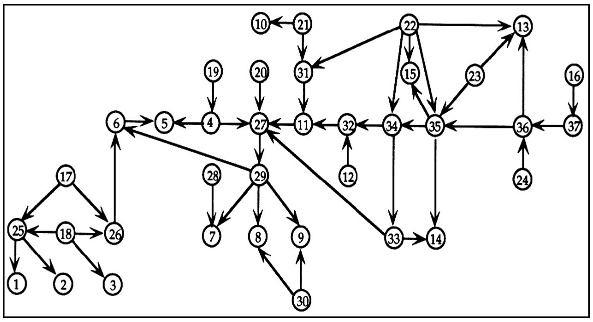

Fig. 13 Bayesian Network structure representing the Alarm problem

As briefly mentioned in Section 6.1, the MAP approach to extract classification rules from a BC takes into account the whole BC structure (variables and arcs). In the Alarm network this will produce over 2^{36} rules, each one with an antecedent part containing 36 variable tests. The BayesRule method, however, by using the Markov Blanket of the class feature, minimizes the number of rules as well as the number of variable tests in the antecedent part of each rule. Table 10 shows the variables used as class variables in the experiments conducted using the Alarm domain. To extract classification rules for each Alarm class variable, the BayesRule procedure was run eight times (one for each class) using the same Alarm network as input.

The Asia BN is a simple graphical model having 8 nodes and 8 arcs. It is commonly used in the literature to illustrate basic concepts of Bayesian networks in diagnosis and learning problems. It was first mentioned in (Lauritzen and Spiegelhalter 1988) and the name Asia came from the fact that, in this BN, there is a node (considered the class variable in the experiments) which models whether an individual has recently visited Asia, which is considered to be a risk factor in tuberculosis. Fig. 14 depicts the Asia network structure.

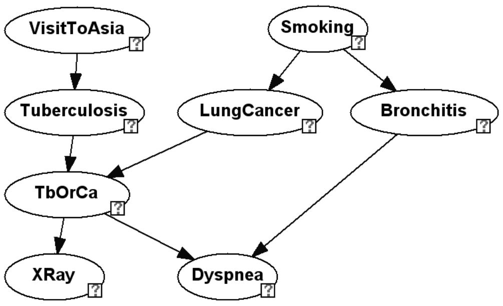

Fig. 14 Bayesian network structure representing the Asia domain

As described in the GeNIe[1] software (Druzdzel 1999), the Credit BN is a simple network for assessing credit worthiness of an individual. The node *CreditWorthiness* is of interest to the user and as such should be assigned as the class variable. All parentless nodes are described by uniform distributions; this is a weakness of the model, although it can be compensated by the fact that most of the time all the corresponding variables will be observed and the network will compute the probability distribution over credit worthiness correctly. Fig. 15 shows the Credit BN structure.

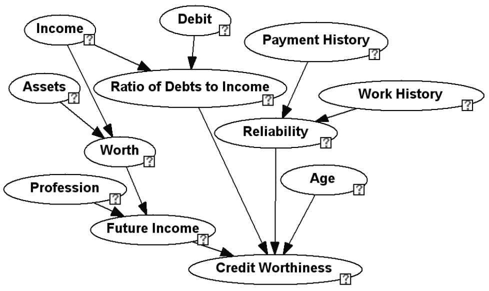

Fig. 15 Bayesian network structure representing the Credit domain

Also in GeNIe software, the Engine Fuel System (Engine) BN describes a simple diagnostic domain of a vehicle fuel system. It has 9 nodes and was created to verify whether the "Fuel Filters and Bypass Valves" are defective or not. Fig. 16 shows the BN structure.

The Win95pts BN was created to be used as an expert system for printer troubleshooting in Windows 95. It was developed at the Microsoft Research Center and was part of the Lumiere Project (Horvitz *et al.* 1998) at Microsoft Research that was initiated in 1993 with the goal of developing methods and an architecture for reasoning about the goals and needs of software users as they

[1] GeNIe modeling environment developed by the Decision Systems Lab. of the University of Pittsburgh (http://www.sis.pitt.edu/~dsl).

work with software. At the heart of the Lumiere are Bayesian models that capture the uncertain relationships between the goals and needs of a user and observations about a program state, sequences of actions over time, and words in a user's query (when such a query has been made). Fig. 17 depicts the Win95pts BN structure.

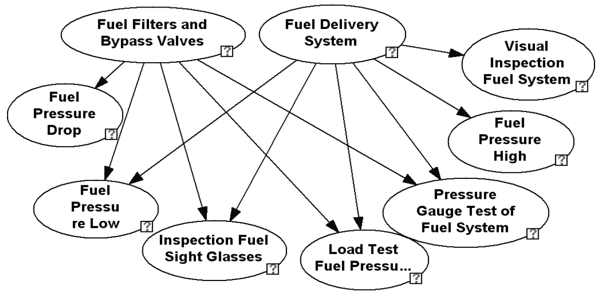
Fuel Filters and Bypass Valves
Fuel Delivery System
Visual Inspection Fuel System
Fuel Pressure Drop
Fuel Pressure High
Fuel Pressu re Low
Pressure Gauge Test of Fuel System
Inspection Fuel Sight Glasses
Load Test Fuel Pressu...

Fig. 16 Bayesian network structure representing the Engine Fuel System domain

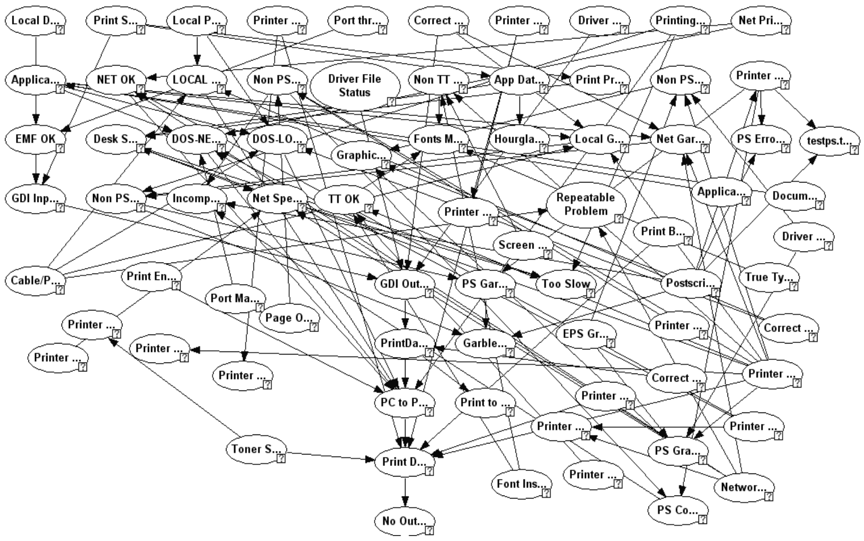

Fig. 17 Bayesian network structure representing the Win95pts domain

Table 10 shows the variables used as class variables in the experiments conducted using the Win95pts domain. Two variables (one at a time) were used as class variables, namely the “Repeatable Problem” and the “Driver File Status”.

As with Alarm BN, the Win95pts BN was not used to classify a single variable. Using the Win95pts BN, however, only two variables (one at a time) were considered as class, namely the “Repeatable Problem” and the “Driver File Status” variables.

Two artificial domains named Synthetic 1 (Syn_1) and Synthetic 2 (Syn_2) were simulated in order to verify the behavior of BayesRule. Such simulations were performed manually by building BNs to encode a joint probability distribution over a set of random variables and, thus, reproducing hypothetical circumstances. Two synthetic BNs also named Synthetic 1 (Syn_1) and Synthetic 2 (Syn_2) were built and are depicted in Fig. 18.

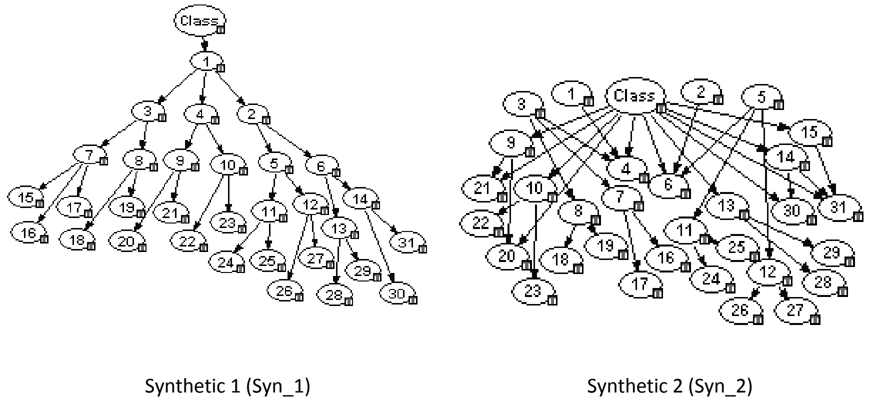

Fig. 18 Bayesian networks representing Synthetic 1 and Synthetic 2 domains

Syn_1 BN represents a domain with 32 variables where only one variable directly influences the class variable (the MB of the class variable has only one variable) and all variables have at most one parent. Therefore, the Syn_1 structure represents a polytree which is a suitable structure to verify the behavior of a classifier in problems where variables have simple interdependencies relationships (Pearl 1988). The BN was created to simulate a situation that favors BayesRule. Considering that the BN has 32 binary variables and only one is present in MB of the class, BayesRule should generate only two rules, each having a single variable test in its antecedent part.

Syn_2 BN describes a domain having 32 variables with 14 variables directly influencing the class variable. In this BN, each variable has 3 parents at most and this fact allows the establishing of more complex interdependency relationships among variables than the polytree structures. Therefore, Syn_2 is a lesser restrictive model than Syn_1. This BN was created in an attempt to simulate a situation that does not favor BayesRule. Considering that the BN has 32 binary variables and that 14 of them are in the MB of the class, BayesRule should generate 2^{14} rules having 14 variable tests in their antecedents. This illustrates a situation where the use of BayesRule is not recommended. In scenarios like that BayesRule should be used with a pruning mechanism (such as confidence and coverage). For measuring the accuracy of the set of generated rules from each BN, a testing set containing 10,000 patterns was created using the GeNIe software.

For a more robust comparative analysis, besides presenting the classification results (ACCRs) obtained using BayesRule, this section also shows the performance of the traditional decision tree based C4.5 algorithm (Quinlan 1993) (using the WEKA data mining environment (Witten and Frank 2000)) in the same domains; the obtained C4.5 trees were translated into sets of classification rules. It is important to remind that an unique BN can be used to extract classification rules having any of involved variables as consequent. Thus, when using BayesRule, a single BN was induced for each dataset. When extracting classification rules from decision trees, however, one particular decision tree will give rise to rules having one specific variable as consequent. Therefore, for the Alarm domain 8 different decision trees (one for each class variable) were induced and for the Win95pts domain, two different decision trees were induced. The performance results of BayesRule and C4.5 are presented in Table 11.

Table 11 Results obtained using BayesRule and C4.5 without pruning. A: Alarm, W:Win95pts and ACCR: Average Correct Classification Rate

Domain		Number of rules		Max. number of variable tests per rule		ACCR(%)	
		BayesRule	C4.5	BayesRule	C4.5	BayesRule	C4.5
A	Hypovolemia (17)	18	151	3	24	98.36	98.96
	LVFailure (18)	36	101	4	16	99.02	99.42
	Anaphylaxis (19)	3	35	1	10	98.97	99.02
	Insufficient Anesthesia (20)	54	795	4	22	88.01	90.68
	PulmEmboulus (21)	18	61	3	14	99.54	99.67
	Intubation (22)	8223	180	8	15	98.61	99.13
	KinkedTube (23)	192	114	4	15	99.19	99.42
	Disconnect (24)	16	89	2	25	98.98	99.23
Asia		2	1	1	1	98.98	98.98
Credit		24	3016	4	11	72.51	66.90
EngineFuelSystem		64	10	6	5	99.94	99.94
W	Repeatable Problem	4	96	2	14	98.22	98.42
	Driver File Status	2	47	1	28	98.23	98.23
Syn_1		2	550	1	18	89.16	81.86
Syn_2		16384	363	14	17	88.12	87.82

The ACCR values in Table 11 were obtained in a 10-fold cross-validation strategy and all the datasets used by both BayesRule and C4.5 were the same. Analyzing the results shown in Table 11 it is possible to observe that the ACCR values produced using either BayesRule or C4.5 set of rules are very similar. The only significant difference occurred in the Credit domain where BayesRule produced a more accurate rule set.

Focusing on the number of rules, however, it can be seen that results produced by BayesRule and C4.5 are not so similar. In ten out of fifteen classification experiments, BayesRule outperformed C4.5. For the class variable *Intubation* (22), BayesRule generated 8,223 rules having at most 8 variable tests each, while the decision tree based approach generated 180 rules having at most 15 variable tests each. The considerably high number of rules generated by BayesRule for the class variable 22 is not surprising. As the Markov Blanket of *Intubation* (22) has eight variables, the number of generated rules tends to be large. Thus, BayesRule may not be convenient when extracting classification rules for a variable having a large MB. The same situation happened with the Synthetic 2 domain, which confirmed this expected behavior. Based on these facts the following rule of thumb is suggested: a variable X may be not suitable for undergoing a 'translation into rules' process (using BayesRule) if |MB(X)| ≥ 6.

In an attempt to improve the results obtained with variable 22 (from Alarm) and with the Synthetic 2 domain, a simple pruning strategy was implemented, that of removing rules containing superfluous variable tests. The C4.5 algorithm also implements a pruning procedure. Results using the pruned rule sets obtained with both algorithms are shown in Table 12. With pruning, BayesRule and C4.5 results have improved, since the number of rules has dropped in most of the experiments.

Table 12 Summary of the results obtained using BayesRule and C4.5 with pruning. A: Alarm, W:Win95pts and ACCR: Average Correct Classification Rate

Domain		Number of rules		Max. number of variable tests per rule		ACCR(%)	
		BayesRule	C4.5	BayesRule	C4.5	BayesRule	C4.5
A	Hypovolemia (17)	12	121	3	17	98.36	98.96
A	LVFailure (18)	32	79	3	10	99.02	99.42
A	Anaphylaxis (19)	1	29	0	7	98.97	99.02
A	Insufficient Anesthesia (20)	34	678	4	22	88.01	90.68
A	PulmEmboulus (21)	9	45	3	8	99.54	99.67
A	Intubation (22)	1905	164	8	14	98.61	99.13
A	KinkedTube (23)	84	83	4	9	99.19	99.42
A	Disconnect (24)	13	79	2	20	98.98	99.23
Asia		2	1	1	1	98.98	98.98
Credit		20	114	4	11	72.51	72.39
EngineFuelSystem		34	10	6	5	99.94	99.94
W	Repeatable Problem	4	44	1	14	98.22	98.33
W	Driver File Status	2	17	1	12	98.23	98.99
Syn_1		2	2	14	1	89.16	89.16
Syn_2		8056	363	14	17	88.12	87.92

The results displayed in Table 12 show that BayesRule generated considerably smaller rule sets than the C4.5 in nine out of fifteen experiments. Even after pruning the resulting rule set for the Alarm, variable 22 remains with 1,905 rules, and for Syn_2, with 8,056 rules. In both cases the number of rules is still considerably high when compared to the number of rules produced by C4.5. There is no enough evidence to state that one method is better than the other. One may conclude, however, that BayesRule is a consistent way of extracting relevant classification rules from a BN; specifically in the conducted experiments, it generated smaller rule sets, when compared to those generated by the C4.5. The use of the MB concept was crucial for simplifying the rule set, while maintaining accuracy. Taking into account the MB of the class variable, the maximum number of generated rules was substantially reduced.

It is important to mention that the number of variables in the Markov Blanket of the class variable is not the only criteria to identify the number of rules to be generated. This situation is illustrated in the Alarm domain with variable 20 and variable 24 as classes. Notice that, although |MB(20)|=4 and |MB(24)| = 2, BayesRule produced a rule set with 12 and 16 rules respectively. This is a consequence of the number of possible values each of these variables can have. Variables that have a greater number of possible associated values generate a higher number of combinations and, consequently, a greater number of rules. Results show that there are two main factors influencing the number of rules generated by BayesRule. One is the Markov Blanket of the class variable and the other, the number of possible values each variable in the Markov Blanket of the class variable has.

In spite of the motivating results, there are still some issues that should be further investigated. There is a possibility that a more elaborated pruning procedure, involving the concepts of confidence and coverage of rules, could improve the results. Another relevant aspect to be explored is related to the fact that the rules extracted from BCs may be in a causal context (Pearl 2000).

5.7 Conclusion

The main goal of this chapter was to show that BN is a sound formalism that has a broad use in many different machine leaning tasks, starting with pre-processing, followed by learning and finally contributing in post processing. Although the chapter describes a few methods related to the three main ML tasks, it is important to mention that (a) there are many approaches that have only been cited and (b) several other roles that Bayesian networks can play have not been addressed. In addition, all the discussed algorithms are based on BNs having only discrete variables. When continuous variables are employed BN variations such as Gaussian BNs (Neapolitan 2003) should be used.

One of the advantages of using the BN approach in all the three ML tasks is to maintain the same inductive bias throughout the whole sequence of ML steps i.e., preparing the data, learning and pruning. Although in particular domains there is a chance of this not being a particularly convenient feature, we believe that, in general, by maintaining the same bias, the whole three-step process can be more efficient.

Due to their similarities to BNs, other probabilistic graphical models such as Markov Networks (Pearl 1988) and Conditional Random Fields (Lafferty *et al.* 2001) can also play the same roles discussed in this paper in spite of not being able to represent causal relationships among variables as BN does.

Acknowledgments. This research was partially supported by Brazilian Research Agencies FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo-Brazil) and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico-Brazil). We wish to express our gratitude to Dr. Sheela Ramanna for her insightful comments and suggestions concerning our manuscript.

References

Abellán, J., Gómez-Olmedo, M., Moral, S.: Some variations on the PC algorithm. In: Proc. of The 3rd European Workshop on Probabilistic Graphical Models (PGM 2006), Prague, pp. 1–8 (2006)

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

Antal, P., Hullám, G., Gézsi, A., Millinghoffer, A.: Learning complex Bayesian network features for classification. In: Proc. of The 3rd European Workshop on Probabilistic Graphical Models, pp. 9–16 (2006)

Antal, P., Millinghoffer, A., Hullam, G., Szalai, C., Falus, A.: A Bayesian view of challenges in feature selection: multilevel analysis, feature aggregation, multiple targets, redundancy and interaction. In: Journal of Machine Learning Research: Workshop and Conference Proceedings, vol. 4, pp. 74–89 (2008)

Batista, G.E.A.P., Monard, M.C.: An analysis of four missing data treatment methods for supervised learning. Applied Artificial Intelligence 17(5-6), 519–534 (2003)

Beinlich, I., Suermondt, H.J., Chavez, R.M., Cooper, G.F.: The ALARM monitoring system: a case study with two probabilistic inference techniques for belief networks. In: Proc. of the 2nd European Conference on Artificial Intelligence in Medicine, London, UK, vol. 38, pp. 247–256 (1989)

Ben-Gal, I.: Bayesian networks. In: Ruggeri, F., Faltin, F., Kenett, R. (eds.) Encyclopedia of Statistics in Quality & Reliability. Wiley & Sons (2007)

Bilmes, J.: 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 (1997)

Blum, A.L., Langley, P.: Selection of relevant features and examples in machine learning. Artificial Intelligence, 245–271 (1997)

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: CART: Classification and Regression Trees. Chapman & Hall, Wadsworth (1983)

Bressan, G.M., Oliveira, V.A., Hruschka Jr., E.R., Nicoletti, M.C.: Using Bayesian networks with rule extraction to infer the risk of weed infestation in a corn-crop. Engineering Applications of Artificial Intelligence 22, 579–592 (2009)

Brown, L.E., Tsamardinos, I.: Markov blanket-based variable selection in feature space. Technical Report DSL TR-08-01, Department of Biomedical Informatics, Vanderbilt University (2008)

Chajewska, U., Halpern, J.Y.: Defining explanation in probabilistic systems. In: Proc. of Conference of Uncertainty in Artificial Intelligence, Providence, RI, pp. 62–71 (1997)

Cheng, J., Bell, D.A., Liu, W.: Learning belief networks from data: an information theory based approach. In: Proc. of The 6th ACM International Conference on Information and Knowledge Management, pp. 325–331 (1997)

Cheng, J., Greiner, R.: Comparing Bayesian network classifiers. In: Proc. of The 15th Conference on Uncertainty in Artificial Intelligence, pp. 101–107 (1999)

Cheng, J., Greiner, R.: Learning Bayesian Belief Network Classifiers: Algorithms and System. In: Stroulia, E., Matwin, S. (eds.) Canadian AI 2001. LNCS (LNAI), vol. 2056, pp. 141–151. Springer, Heidelberg (2001)

Cheng, J., Greiner, R., Kelly, J., Bell, D., Liu, W.: Learning Bayesian networks from data: an information-theory based approach. Artificial Intelligence 137(1), 43–90 (2002)

Chickering, D.M.: Learning Bayesian networks is NP-complete. In: Fisher, D., Lenz, A. (eds.) Learning from Data: Artificial Intelligence and Statistics V, pp. 121–130. Springer (1996)

Chickering, D.M.: Optimal structure identification with greedy search. Journal of Machine Learning Research 3, 507–554 (2002)

Cooper, G.F.: The computational complexity of probabilistic inference using Bayesian belief networks (research note). Artificial Intelligence 42(2-3), 393–405 (1990)

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

Cooper, G.F.: NESTOR: A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge. PhD thesis, Medical Information Sciences, Stanford University, Stanford, CA (1984)

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

Díez, F.J., Mira, J., Iturralde, E., Zubillaga, S.: Diaval, a Bayesian expert system for echocardiography. Artificial Intelligence in Medicine 10(1), 59–73 (1997)

Duda, R.O., Hart, P.E.: Pattern classification and scene analysis. John Wiley & Sons (1973)

Druzdzel, M.J.: Qualitative verbal explanations in Bayesian belief networks. Artificial Intelligence and Simulation of Behaviour Quarterly 94, 43–54 (1996)

Druzdzel, M.J.: SMILE: Structural modeling, inference, and learning engine and GeNIe: A development environment for graphical decision-theoretic models. In: Proc. of the 16th National Conference on Artificial Intelligence, Orlando, FL, pp. 902–903 (1999)

Duch, W., Adamczak, R., Grabczewski, K.: A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. IEEE Transactions on Neural Networks 11(2), 1–31 (2000)

Fast, A., Jensen, D.: Constraint relaxation for learning the structure of Bayesian networks. Technical Report 09-18, Computer Science Department, University of Massachusetts, Amherst (2009)

Fayyad, U.M., Shapiro, G.P., Smyth, P.: From data mining to knowledge discovery: an overview. In: Fayyad, et al. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 1–37. MIT Press (1996)

Frank, A., Asuncion, A.: UCI Machine Learning Repository. School of Information and Computer Science. University of California, Irvine (2010), `http://archive.ics.uci.edu/ml`

Friedman, N., Linial, M., Nachman, I., Pe'er, D.: Using Bayesian network to analyze expression data. Journal of Computational Biology 7, 601–620 (2000)

Friedman, N.: Inferring cellular networks using probabilistic graphical models. Science 303, 799–805 (2004)

Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning 29, 131–163 (1997)

Friedman, N., Goldszmidt, M.: Building classifiers using Bayesian networks. In: Proc. of the AAAI 1996, vol. 2, pp. 1277–1284 (1996)

Friedman, H.F., Kohavi, R., Yun, Y.: Lazy decision trees. In: Proc. of the 13th National Conference on Artificial Intelligence, pp. 717–724. AAAI Press/MIT Press, Cambridge, MA (1996)

Fu, F.S., Demarais, M.C.: Markov blanket based feature selection: a review of past decade. In: Proc. of the World Congress on Engineering (WCE 2010), London, UK, pp. 321–328 (2010)

Ghahramami, Z., Jordan, M.: Learning from incomplete data. Technical Report AI Lab Memo no. 1509, CBCL paper no. 108. MIT AI Lab. (1995)

Guo, H., Hsu, W.: A survey on algorithms for real-time Bayesian network inference. In: Proc. of The AAAI-02/KDD-02/UAI-02 Joint Workshop on Real-Time Decision Support and Diagnosis Systems, Edmonton, Alberta, Canada (2002)

Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research 3, 1157–1182 (2003)

Heckerman, D.: Bayesian networks for data mining. Data Mining and Knowledge Discovery Journal 1(1), 79–119 (1997)

Heckerman, D., Geiger, D.: Learning Bayesian networks: a uni. cation for discrete and Gaussian domains. In: Proc. 11th Conference on Uncertainty in Artificial Intelligence (UAI 1995), pp. 274–284 (1995)

Heckerman, D., Chickering, D.M., Meek, C., Rounthwaite, R., Kadie, C.: Dependency networks for inference, collaborative filtering, and data visualization. Journal of Machine Learning Research 1(1), 49–75 (2000)

Henrion, M., Druzdzel, M.J.: Qualitative propagation and scenario-based approaches to explanation of probabilistic reasoning. In: Proc. of 6th Conference on Uncertainty in Artificial Intelligence, Cambridge, MA, pp. 17–32 (1990)

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., Rommelse, K.: The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In: Proc. of the 14th Conference on Uncertainty in Artificial Intelligence, Madison, WI, pp. 256–265. Morgan Kaufmann, San Francisco (1998)

Hruschka Jr., E.R., Nicoletti, M.C., Oliveira, V., Bressan, G.: BayesRule: a Markov-blanket based procedure for extracting a set of probabilistic rules from Bayesian classifiers. Int. Journal of Hybrid Intelligent Systems 76(2), 83–96 (2008)

Hruschka, E.R., Garcia, A., Hruschka Jr., E.R., Ebecken, N.F.F.: On the influence of imputation in classification: practical issues. Journal of Experimental and Theoretical Artificial Intelligence 21, 43–58 (2009)

Hruschka Jr., E.R., Hruschka, E.R., Ebecken, N.F.F.: Bayesian networks for imputation in classification problems. Journal of Intelligent Information Systems 29, 231–252 (2007)

Hruschka Jr., E.R., Hruschka, E.R., Ebecken, N.F.F.: Feature Selection by Bayesian Networks. In: Tawfik, A.Y., Goodwin, S.D. (eds.) Canadian AI 2004. LNCS (LNAI), vol. 3060, pp. 370–379. Springer, Heidelberg (2004)

Hruschka Jr., E.R., Ebecken, N.F.F.: Missing values prediction with K2. Intelligent Data Analysis Journal (IDA) 6(6), 557–566 (2002)

Hruschka Jr., E.R., Ebecken, N.F.F.: Ordering attributes for missing values prediction and data classification. In: Data Mining III - Management Information Systems Series, 6th edn., WIT Press, Southampton (2002)

Hruschka, E.R., Hruschka Jr., E.R., Ebecken, N.F.F.: Evaluating a Nearest-Neighbor Method to Substitute Continuous Missing Values. In: Gedeon, T(T.) D., Fung, L.C.C. (eds.) AI 2003. LNCS (LNAI), vol. 2903, pp. 723–734. Springer, Heidelberg (2003)

Husmeier, D., Dybowski, R., Roberts, S. (eds.): Probabilistic modeling in bioinformatics and medical informatics. Springer, London (2005)

Inza, I., Larrañaga, P., Etxeberia, R., Sierra, B.: Feature subset selection by Bayesian networks based optimization. Artificial Intelligence 123(1-2), 157–184 (2000)

Inza, I., Larrañaga, P., Sierra, B.: Feature subset selection by Bayesian networks: a comparison with genetic and sequential algorithms. International Journal of Approximate Reasoning 27, 143–164 (2001)

Jansen, R., et al.: A Bayesian network approach for predicting protein-protein interactions from genomic data. Science 302, 449–453 (2003)

John, G., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: Proc. of the 11th International Conference on Machine Learning, pp. 121–129 (1994)

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

Kalisch, M., Bühlmann, P.: Estimating high-dimensional directed acyclic graphs with the PC-algorithm. Journal of Machine Learning Research 8, 613–636 (2007)

Kohavi, R., Becker, B., Sommerfield, D.: Improving simple Bayes. In: van Someren, M., Widmer, G. (eds.) Poster papers of the ECML 1997, pp. 78–87. Charles University, Prague (1997)

Koller, D., Sahami, M.: Toward optimal feature selection. In: Proc. of the 13th International Conference on Machine Learning, pp. 284–292 (1996)

Kong, A., Liu, J.S., Wong, W.H.: Sequential imputations and Bayesian missing data problems. Journal of the American Statistical Association 89(425), 278–288 (1994)

Kononenko, I., Bratko, I., Roskar, E.: Experiments in automatic learning of medical diagnostic rules. Technical Report, Jozef Stefan Institute, Ljubjana (1984)

Lacave, C., Díez, F.: A review of explanation methods for Bayesian networks. The Knowledge Engineering Review 17(2), 107–127 (2002)

Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proc. 18th International Conference on Machine Learning, pp. 282–289. Morgan Kaufmann, San Francisco (2001)

Lam, W., Bacchus, E.: Using causal information and local measures to learn Bayesian networks. In: Proceedings of 9th Conference on Uncertainty in Artificial Intelligence, Washington, DC, pp. 243–250 (1993)

Langley, P., Iba, W., Thompson, K.: An analysis of Bayesian classifiers. In: Proc. of the AAAI 1992, pp. 223–228 (1992)

Langley, P., Sage, S.: Induction of selective Bayesian classifiers. In: Proc. of the 10th Conference on Uncertainty in Artificial Intelligence, pp. 399–406. Morgan Kaufmann Publishers, Seattle (1994)

Lauritzen, S.L.: Some modern applications of graphical models. In: Green, P.J., Hjort, N.L., Richardson, S. (eds.) Highly Structured Stochastic Systems. Oxford University Press (2003)

Lauritzen, S., Spiegelhalter, D.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society B 50, 157–224 (1988)

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

Liu, H., Motoda, H.: Feature selection for knowledge discovery and data mining. Kluwer Academic (1998)

Lobo, O.O., Noneao, M.: Ordered estimation of missing values for propositional learning. Journal of the Japanese Society for Artificial Intelligence 15(1), 162–168 (2000)

Madden, M.G.: Evaluation of the performance of the Markov blanket Bayesian classifier algorithm. Technical Report No. NUIG-IT-011002, NUI Galway, Ireland (2002)

Mitchell, T.: Machine learning. The McGraw-Hill Companies, Inc. (1997)

Moore, A.: Data Mining Tutorials (2011), `http://www.autonlab.org/tutorials/`

Murphy, K.: A brief introduction to graphical models and Bayesian networks (1998), `http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html`

Neapolitan, R.E.: Learning Bayesian networks. Prentice Hall (2003)

Nicoletti, M.C.: The feature subset selection problem in machine learning – Talk presented at The Seventh International Conference on Intelligent Systems Design and Applications, Rio de Janeiro, Brazil (2007) (unpublished)

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

Pearl, J.: Causality: models, reasoning, and inference. Cambridge University Press (2000)

Pourret, O., Nai, P., Marcot, B.: Bayesian networks: a practical guide to applications. Wiley, Chichester (2008)

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

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

Quinlan, J.R.: C4.5 program for machine learning. Morgan Kaufmann, San Francisco (1993)

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

Redner, R., Walker, H.: Mixture densities, maximum likelihood and the EM algorithm. SIAM Review 26(2), 195–239 (1984)

Reunanen, J.: Overfitting in making comparisons between variable selection methods. Journal of Machine Learning Research 3, 1371–1382 (2003)

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

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

Rubin, D.B.: Multiple imputation for non-responses in surveys. John Wiley & Sons, New York (1987)

Russel, S., Norvig, P.: Artificial intelligence: a modern approach. Prentice Hall Series in Artificial Intelligence (1995)

Sachs, K., Perez, O., Pe'er, D., Lauffenburguer, D.A., Nolan, G.P.: Causal protein-signaling networks derived from multiparameter single-cell data. Science 308, 523–529 (2005)

Santos, E.B., Hruschka Jr., E.R., Nicoletti, M.C.: Conditional independence based learning of Bayesian classifiers guided by a variable ordering genetic search. In: Proc. of CEC 2007, vol. 1, pp. 1–10. IEEE Press, Los Alamitos (2007)

Schllimmer, J.C.: Concept acquisition through representational adjustment. Doctoral Dissertation, Department of Information and Computer Science. University of California, Irvine (1987)

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

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

Sebastiani, P., Yu, Y.-H., Ramoni, M.F.: Bayesian machine learning and its potential applications to the genomic study of oral oncology. Advances in Dental Research 17, 104–108 (2003)

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

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

Spirtes, P., Meek, C.: Learning Bayesian networks with discrete variables from data. In: KDD 1995, pp. 294–299 (1995)

Suzuki, J.: A construction of Bayesian networks from databases based on an MDL scheme. In: Proc. of 9th Conference on Uncertainty in Artificial Intelligence, Washington, DC, pp. 266–273 (1993)

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

Troyanskaya, O.G., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., Altman, R.B.: Missing value estimation methods for DNA microarrays. Bioinformatics 17(6), 520–525 (2001)

White, A.P.: Probabilistic induction by dynamic path generation in virtual trees. In: Bramer, M.A. (ed.) Research and Development in Expert Systems III, pp. 35–46. Cambridge University Press (1987)

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

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

Zeng, Y., Luo, J., Lin, S.: Classification using Markov blanket for feature selection. In: Proc. of The International Conference on Granular Computing (GrC 2009), pp. 743–747 (2009)

Zio, M.D., Scanu, M., Coppola, L., Luzi, O., Ponti, A.: Bayesian networks for imputation. Journal of the Royal Statistical Society, Series A (Statistics in Society) 167(2), 309–322 (2004)