Data Min Knowl Disc
DOI 10.1007/s10618-011-0247-5

Score-based methods for learning Markov boundaries by searching in constrained spaces

Silvia Acid · Luis M. de Campos · Moisés Fernández

Received: 12 May 2010 / Accepted: 21 November 2011

Abstract Within probabilistic classification problems, learning the Markov boundary of the class variable consists in the optimal approach for feature subset selection. In this paper we propose two algorithms that learn the Markov boundary of a selected variable. These algorithms are based on the score+search paradigm for learning Bayesian networks. Both algorithms use standard scoring functions but they perform the search in constrained spaces of class-focused directed acyclic graphs, going through the space by means of operators adapted for the problem. The algorithms have been validated experimentally by using a wide spectrum of databases, and their results show a performance competitive with the state-of-the-art.

Keywords Feature subset selection · Markov boundary · Bayesian networks · Local learning · Score+search

1 Introduction

Automatic classification is a fundamental task in data mining and machine learning, that requires learning a classifier from a given training data. Basically, this task can be regarded as a problem of setting a function, the classifier, that maps an assignment of

Responsible editor: Charles Elkan.

S. Acid · L. M. de Campos (✉) · M. Fernández
Departamento de Ciencias de la Computación e Inteligencia Artificial, E.T.S.I. Informática y de Telecomunicación, CITIC-UGR, Universidad de Granada, 18071 Granada, Spain
e-mail: lci@decsai.ugr.es

S. Acid
e-mail: acid@decsai.ugr.es

M. Fernández
e-mail: moises@decsai.ugr.es

Published online: 06 December 2011

values for variables in the domain (the features) to a class label. Examples of domains where automatic classification has been applied include medicine (Inza et al. 2001), industry (Mamitsuka 2003), business (Li et al. 2010) and financial sector (Song et al. 2010).

In the last decades, progressively, it has been facing more complex classification domain problems, that arise from fields as molecular biology (Guyon et al. 2002), astronomy (Zheng and Zhang 2008) or text categorization (Yang and Pedersen 1997), among others. The current explored domains may contain hundreds, thousands or even tens of thousands of variables or features, sometimes with the restriction that the number of available samples may be severely limited. That is, domains with a high number of features and relatively few samples for training. With such data, processes that carry out some kind of dimensionality reduction become critical for the classification task. That is basically what is known as the feature subset selection (FSS) task. Besides the evident benefits to render problems approachable, the FSS task facilitates the interpretation of a classification model, provides reduction in the cost of acquisition or storage requirements, and may bring improvements in classification accuracy. Of course these benefits are also applicable to cases where the number of samples is high. The issue of the paper is focused on *selecting subsets of features* that are useful to build a good predictor. This contrasts with the problem of ranking independently all the potentially relevant variables.

We can find a variety of algorithms that perform FSS in the classification literature (Blum and Langley 1997; John and Kohavi 1994; Kohavi and John 1997; Langley and Sage 1994), where they are basically grouped into *filter* and *wrapper* approaches, although some authors (Guyon and Elisseeff 2003) distinguish a third group, the *embedded* methods. Wrappers use the learning machine of interest as a black box to score subsets of variables according to their predictive power. Filters select subsets of variables as a preprocessing step, independently of the chosen predictor. Embedded methods perform variable selection within the training process and are usually specific to given learning machines. We are interested in the embedded methods following (Guyon and Elisseeff 2003) or in the more general filter approach, following (Kohavi and John 1997). The goal of our algorithms will be to find a minimal set of features that renders the target (class) variable conditionally independent from the rest of the features in the domain. We want to directly find a good subset of features, not a ranking of good individual features. In the literature about Bayesian networks, such a set is known as a Markov boundary (MB) for the target variable (Pearl 1988).

For probabilistic models, the problem of learning the Markov boundary for a distinguished variable C (the class) is the optimal approach to the FSS problem (eliminating irrelevant as well as redundant features). A naive approach to obtain the Markov boundary of C could be to learn a complete Bayesian network from data for the domain and then to extract the MB for the class variable from the topology of the graph (the MB of any variable is composed by the set of its parents, children and parents of the children in the graph). But this approach is not operational for high dimensional databases: learning a global Bayesian network would be a waste of computing effort on discovering superfluous connectivity between irrelevant features. A better option could be to learn a local Bayesian network focused on the target variable and then to compose the MB for it.

There are two broad classes of Bayesian network structure learning algorithms: methods based on *conditional independence tests* (also called *constraint-based*) and *score+search-based* methods. The constraint-based techniques (Spirtes et al. 1993; de Campos and Huete 2000) try to infer the structure of the generating Bayesian network through statistical tests of conditional independence on the training data. On the other hand, the score+search methods seek a structure that best fits the given training data according to a selected metric or scoring function (Buntine 1991; Cooper and Herskovits 1992; Heckerman et al. 1995).

In the current state-of-the-art, we can find a large number of constraint-based algorithms for discovering Markov boundaries (Aliferis et al. 2003; Fu and Desmarais 2007, 2008a,b; Margaritis and Thrun 2000; Pellet and Elisseeff 2008; Peña et al. 2005, 2007; Ramsey 2006; Rodrigues de Morais and Aussem 2008a,b; Tsamardinos et al. 2003a,b; Yaramakala and Margaritis 2005; Yishi et al. 2009), as well as some algorithms for feature selection roughly based on the score+search approach (Koller and Sahami 1996; Sierra and Larrañaga 1998; Singh and Provan 1996). We propose two new filter algorithms, called DMB and RPDMB, that use standard scoring functions but search in special spaces of graphs. They also use some strategies for modifying/pruning the search inspired by the constraint-based methods. The goal for both algorithms is to get an accurate MB in order to improve classification rates for any potential classifier built on these data, although the network obtained can also be used directly as a classifier. In this sense they can also be considered as embedded methods. We demonstrate empirically that our algorithms outperform several state-of-the-art algorithms according to different measures (*precision*, *recall* and F), measured over a wide spectrum of different domains and a variety of database sizes.

The remainder of this paper is organized as follows: in Sect. 2 we review some preliminaries and describe some Bayesian network formalisms that allow us to connect the notion of Markov boundary to Bayesian networks. In Sect. 3 we review past works related to Markov boundary discovery. Section 4 presents, in an incremental way, the steps followed in order to develop the proposed algorithms for discovering Markov boundaries, including the specific searching spaces and the way to travel through. In Sect. 5 we give empirical results comparing these algorithms with earlier works. We end with the conclusions and some thoughts on possible future research directions.

2 Preliminaries

The next definitions are central to the development of the paper and consist in concepts widely known in the Bayesian network community, which can be extended in (Koller and Friedman 2009; Neapolitan 2003; Pearl 1988; Spirtes et al. 1993). Let $\mathbf{U}$ denote a set of unidimensional discrete random variables $\{X_1, X_2, \ldots, X_n\}$. We shall use capital letters for variable names, and lower-case letters to denote specific values taken by those variables. Sets of variables are denoted by boldface capital letters and assignments of values to the variables in these sets are denoted by boldface lower-case letters.

2.1 Conditional independence

Given a probability distribution p, two sets of variables $\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent given a set of variables $\mathbf{Z}$, when

$$p(\mathbf{x}|\mathbf{y},\mathbf{z}) = P(\mathbf{x}|\mathbf{z}), \quad \forall \mathbf{x},\mathbf{y},\mathbf{z} \text{ such that } p(\mathbf{y},\mathbf{z}) > 0,$$

and it is denoted as $\mathbf{X} \perp_p \mathbf{Y}|\mathbf{Z}$.

2.2 Bayesian network

A Bayesian network (BN) for $\mathbf{U}$ is an annotated directed acyclic graph (DAG) that represents a joint probability distribution over $\mathbf{U}$. Formally, it is a pair (G, θ), where $G = (\mathbf{U}, E_G)$ is a DAG (whose nodes correspond to the variables in $\mathbf{U}$ and the set of arcs, E_G, represents direct dependence relationships between these variables) and θ is the set of parameters that specify the probability distributions: for each variable X_i we have a family of conditional distributions $p(X_i|pa_G(X_i))$, one for each instantiation, $pa_G(X_i)$, from the parent set of X_i in the graph, $Pa_G(X_i) = \{X_j \in \mathbf{U}|X_j \to X_i \in E_G\}$. One of the benefits most relevant of a BN model is that it represents the joint distribution $p(\mathbf{U})$ in a abbreviated manner, through the factorization

$$p(x_1,\ldots,x_n) = \prod_{X_i \in \mathbf{U}} p(x_i|pa(X_i)),$$

saving additional parameters and computations in many reasoning tasks. When the arcs in a BN are interpreted as direct causal relations, then these so-called causal networks can be used also to predict the effects of interventions or manipulations, which generate structural changes in the joint distribution.

2.3 Markov boundary

A Markov blanket of a variable $X_i \in \mathbf{U}$, $MB(X_i)$, is any subset of $\mathbf{U}\backslash\{X_i\}$ such that $X_i \perp_p \mathbf{U}\backslash\{X_i\}\backslash MB(X_i)|MB(X_i)$, that is, any subset of variables such that X_i is conditionally independent of all the remaining variables given $MB(X_i)$. A minimal Markov blanket (none of its proper subsets satisfies the condition) of X_i is called a Markov boundary (MB) of X_i. The MB has the property of acting as a shield protecting the target variable from the effect of the rest of the variables in the domain. The class variable C in a classification problem would be described in the best possible way by means of the features in $MB(C)$. Therefore, learning the Markov boundary for the class variable consists in the optimal approach to feature selection. In fact, feature selection and causal structure learning are related through this concept, which coincides with the set of strongly relevant features, as defined in Kohavi and John (1997). As stated in Pellet and Elisseeff (2008), feature selection and causal network construction can both be stated to some extent as MB identification tasks. Several algorithms for feature

selection are based on finding the MB by using methods inspired from (constraint-based) causal structure learning, as we have already commented. But there are also algorithms that do the converse, i.e. they use a feature selection algorithm to build the structure of a causal network (Margaritis and Thrun 2000; Pellet and Elisseeff 2008).

2.4 Faithfulness

Any probability distribution p that can be represented by a BN with structure G satisfies certain independencies between variables, which may be deduced from the *Markov condition*,[1] and which can be identified from G via the d-separation criterion (or graphical conditional independence), denoted by $\mathbf{X} \perp_G \mathbf{Y}|\mathbf{Z}$ (Pearl 1988): $\mathbf{X} \perp_G \mathbf{Y}|\mathbf{Z} \Rightarrow \mathbf{X} \perp_p \mathbf{Y}|\mathbf{Z}$. A probability distribution p is said to be faithful with respect to G iff the d-separations in G identify all and only the conditional independencies in p, that is, $\mathbf{X} \perp_G \mathbf{Y}|\mathbf{Z} \Leftrightarrow \mathbf{X} \perp_p \mathbf{Y}|\mathbf{Z}$. If p is faithful to a DAG G, then the Markov boundary of any node X_i is unique. Moreover, this set is composed of the parents, the children and the parents of the children (also known as spouses) of X_i.

2.5 The metrics

The problem of learning a Bayesian network in the score+search paradigm can be formally expressed as follows: given a training data set D of instances of $\mathbf{U}$, find a DAG G^* such that

$$G^* = \arg \max_{G \in \mathcal{G}} f(G : D)$$

where $f(G : D)$ is the scoring function, or metric, measuring the degree of fitness of any candidate DAG G to the data D, and $\mathcal{G}$ is the family of all the DAGs defined on $\mathbf{U}$.

A desirable property for a scoring function is the *decomposability*, that enables to compute the global score of a DAG by aggregating local assessments. Formally, a scoring function f is decomposable if the value assigned to a structure can be expressed as a combination (sum in the logarithmic space) of local values for each node and its parents in the DAG G:

$$f(G : D) = \sum_{X_i \in \mathbf{U}} f(X_i; Pa_G(X_i) : D_{X_i, Pa_G(X_i)})$$

where $D_{X_i, Pa_G(X_i)}$ are the statistics of the variables X_i and $Pa_G(X_i)$ in D. In this way, only the statistics corresponding to the variables whose parent sets have been modified by the search process need to be recomputed.

Several scoring functions have been defined in the specialized literature, based on different principles, and giving rise to many algorithms for learning Bayesian networks, such as K2 (Cooper and Herskovits 1992), BDeu (Buntine 1991), BDe

[1] The Markov condition states that any node in a Bayesian network is conditionally independent of its nondescendants, given its parents.

(Heckerman et al. 1995), BIC/MDL (Lam and Bacchus 1994; Schwarz 1978), AIC (Akaike 1974) and MIT (de Campos 2006).

When we are trying to learn a BN model with classification purposes, it has been argued that these metrics measure the fitness between the joint distribution associated to the network and the data, but they do not measure the fitness between the conditional distribution of the class given the attributes and the data. As a good joint model may not be a good model for classification (Heckerman and Meek 1997b), then the use of these metrics could result in poor classifiers (Friedman et al. 1997). An option is to use the classifier obtained from the current network being explored to estimate its degree of accuracy, and use this value as a metric. This method is computationally expensive unless the selected model is very simple (due to overfitting problems that require using estimation schemes, such as cross-validation, that evaluate the accuracy on various data subsets). Moreover it leads us to the wrapper approach in which we are not interested here. Another option would be to develop a specialized metric that could account how well a network describes the probability distribution of the class variable given the attributes, such as conditional likelihood. Unfortunately, attempts in this direction have resulted computationally intractable (Friedman et al. 1997), except in some simplified network models (Roos et al. 2005).

We argue that some of the standard metrics for learning general Bayesian networks can also be useful in order to learn Markov boundaries, if they are used in conjunction with appropriate constrained search spaces.

So, we discuss briefly the metrics that we shall use in our experiments, namely, BIC/MDL, BDeu and MIT, all of them being decomposable and issued from different approaches. BIC/MDL and MIT are based on information theory, while BDeu is a Bayesian metric. Each one presents different tendency or bias to select simpler o more complex structures:

- The BIC/MDL metric (Lam and Bacchus 1994) measures how well a structure encodes compactly a training set. The best model is the one that minimizes the sum of the description lengths of the model and the data given the model. Simple models require shorter description lengths whereas the description length of the data given the model increases. Complex models have the contrary effect. The minimum description length (MDL) principle establishes an appropriate trade-off between complexity and precision.
- The BDeu metric, originally proposed in Buntine (1991), is a particular case of the Bayesian Dirichlet metric (Heckerman et al. 1995), where the prior network assigns a uniform probability to each configuration of a node and its parents in the network. The metric has the input parameter, η, representing the equivalent sample size.
- Finally, we shall use the MIT metric (de Campos 2006) by its affinity with the statistical tests of independence, as it quantifies independences based on a χ^2 distribution. The function uses the mutual information in order to measure the degree of interaction between each variable and its parents in the network, but including a penalizing term. This term takes into account the network complexity as well as its reliability based on the data. The MIT metric needs an input parameter, α, that represents the confidence level associated with the underlying statistical test.

3 Previous work

We have established that we want to perform FSS for potential classification improvement and that this problem leads to the problem of learning MBs for the target variable from data. So, we shall review what has been done in the literature in this sense.

Koller and Sahami (1996) first established the relevance of the MB concept to the learning of a classifier. They proposed a greedy algorithm for backward feature selection by using a measure of cross-entropy, which eliminates one by one features starting with the full feature set. Singh and Provan (1996) proposed another method for forward feature selection. They begin with an empty set of features and add features one by one, using several information-theoretic measures (e.g. conditional information gain). Sierra and Larrañaga (1998) proposed a wrapper approach for feature selection within a score+search-based algorithm for learning Bayesian networks: the search is carried out by means of genetic algorithms, in a space of DAGs where all the attributes must belong to the MB of the class variable, and each candidate network is evaluated using its accuracy as the classifier. The idea of learning the MB to build a classification model is also the basis of the work of Heckerman and Meek (1997a).

However, the dominant approach in the literature is to use constraint-based methods for discovering Markov boundaries, where we can find many algorithms (Aliferis et al. 2003; Fu and Desmarais 2007, 2008a,b; Margaritis and Thrun 2000; Pellet and Elisseeff 2008; Peña et al. 2005, 2007; Ramsey 2006; Rodrigues de Morais and Aussem 2008a,b; Tsamardinos et al. 2003a,b; Yaramakala and Margaritis 2005; Yishi et al. 2009). Among them we emphasize the top referenced IAMB (Tsamardinos et al. 2003a), KIAMB and PCMB (Peña et al. 2007), that have demonstrated to be accurate on high dimensional databases. These algorithms have three main characteristics in common.

First, running χ^2 independence tests with the G^2 statistic in order to decide whether the conditional independence of C and X given $\mathbf{Z}$ is supported by the data D, according to what they decide to rule out feature X from $MB(C)$. Second, in order to select the features for the set $MB(C)$, they use the negative p-value of the tests to measure the strength of the dependence between C and X, $dep(C, X|\mathbf{Z})$, with respect to D. Third, they all use a two-phase approach that includes a *growing* and a *shrinking* phase. Starting from an empty set, they *grow* the set of features $MB(C)$ with the attributes that have shown to be very dependent with the target variable C and have been found inseparable from C by the independence tests. This phase may include some false positives. The growing phase is complemented by a shrinking phase that identifies the false positives and removes them from $MB(C)$. Let us introduce briefly each one of the selected algorithms, which will be used in a comparative study within the experimental part of the paper.

On one hand, the algorithm IAMB uses a heuristics in the growing phase that consists in selecting those attributes that are more dependent from the target attribute C, given the current $MB(C)$ set, for including them into the Markov boundary. The shrinking phase is carried out when the growing phase has finished, and removes from $MB(C)$ those attributes which are independent from C given the rest of $MB(C)$. On the other hand, the algorithm PCMB breaks the problem of finding $MB(C)$ into two subproblems: first identifying the inseparable set of features, $PC(C)$, composed by

the parents and the children of C and, second, identifying the rest of the components of $MB(C)$, i.e. the parents of the children of C. The algorithm PCMB is more data efficient than IAMB because it takes into account the graph topology, and then it performs tests conditioned on smaller sets. Finally, the algorithm KIAMB is a stochastic variant of IAMB with the aim of being more data efficient and less prone to errors. It includes an input parameter $k \in [0, 1]$ which specifies a trade-off between greediness and randomness in the search ($k = 1$ means maximum greediness, whereas $k = 0$ represents maximum randomness) in such a way that the node added to the current $MB(C)$ is not always the most dependent node. The authors recommended $k = 0.8$ from their experiments.

Under the assumptions that the independence tests are correct and that the database is an independent and identically distributed sample from a probability distribution faithful to a DAG, all these algorithms recover the correct Markov boundary of the variable C. In practice, the algorithms perform a test only in case it is reliable and skip it otherwise. A test is considered reliable when the number of instances in D is at least five times the number of degrees of freedom in the test (Spirtes et al. 1993), which is exponential in the size of the conditioning set. The statistical power of the tests depends on the amount of data: when the sample size is small (but large enough to run the test, i.e. more than five samples per parameter), the test may fail and then a wrong relationship of independence between pairs of dependent variables is assumed. When the test cannot be carried out, the algorithms are assuming a dependence relationship that may also be wrong. The assumption of faithfulness may also be problematic, as it is not clear at all that it can be generally applied. In particular, it forbids the existence of hidden common causes, which does not seem reasonable in many domains (as we are then assuming that all the relevant variables have been taken into account).

4 Search spaces for local learning of Markov boundaries

Once a Bayesian network has been learned from data by any (unrestricted) learning algorithm, either the local subnetwork induced by the MB for the class variable C can be used directly for classification purposes, or $MB(C)$ can be extracted trivially from the network structure in order to build a classifier by another method. As learning a complete BN may be unnecessarily expensive in complex domains, a reasonable alternative consists in learning only the local structure *surrounding* the target variable C, thus reducing the computing effort.

We are interested here on the score+search paradigm for learning BNs, and more specifically on the search component (for the score component we will use standard metrics, as those commented in Sect. 2). We argue that by defining appropriately the restricted search space, a local structure learning algorithm can be a viable solution to the problem of learning Markov boundaries.

4.1 The space of C-DAGs

In order to design the restricted search space for looking for MBs, an important fact that has to be taken into consideration is that many different DAGs will generate the

same posterior probabilities for the class variable, and therefore they are indistinguishable from a classification point of view. In Acid et al. (2005) the minimum subgraph of any DAG G that performs exactly as G as a classifier for the variable C was identified. Therefore, to avoid exploring redundant configurations, the search space could be restricted to this kind of DAGs, which are called *class-focused* DAGs, or C-DAGs (Acid et al. 2005). Formally,

Definition 1 A DAG $G = (\mathbf{V}, E_G)$, where $\mathbf{V} = \mathbf{U} \cup \{C\}$, is a class-focused DAG with respect to the distinguished variable C iff it satisfies the following condition:

$$\forall X_i, X_j \in \mathbf{V}, \text{ if } X_i \rightarrow X_j \in E_G \text{ then either } X_j = C \text{ or } X_i = C \text{ or } C \rightarrow X_j \in E_G. \quad (1)$$

Therefore, C-DAGs contain only arcs linking C with other variables (in either direction) and arcs between variables different from C only in the case that C is the other parent of some of them. A C-DAG may be considered as a canonical representation of a class of DAGs (Acid et al. 2005), all of which share the same Markov boundary for C and are equivalent for classification (in fact they not only have the same MB for C but also the same arcs connecting the variables within $MB(C) \cup \{C\}$).

So, instead of searching in the whole space of DAGs, we restrict the search to the space of C-DAGs. In spite of the space reduction, the space is still exponential, so that we need to explore it using heuristic search. For efficiency reasons we chose a simple local search [which is so frequently used by the score+search BN learning algorithms (Chickering et al. 1995; Cooper and Herskovits 1992; Heckerman et al. 1995)], with the operators adapted to this space. These operators are designed in such a way that their application to a C-DAG always produces another C-DAG as the result (those operators that change the structure and composition of the Markov boundary for the variable C). We use following operators for adding or removing arcs to the current model:

- *A_ParentOfC*(X_i), addition of the arc $X_i \rightarrow C$.
- *A_ChildOfC*(X_i), addition of the arc $C \rightarrow X_i$.
- *A_ParentOfChild*(X_i, X_j), addition of the arc $X_i \rightarrow X_j$, if X_j is a child of C.
- *D_ParentOfC*(X_i), deletion of the arc $X_i \rightarrow C$.
- *D_ChildOfC*(X_i), deletion of the arc $C \rightarrow X_i$, together with all the other arcs pointing to X_i.
- *D_ParentOfChild*(X_i, X_j), deletion of the arc $X_i \rightarrow X_j$, if X_j is a child of C.

4.2 The space of C-RPDAGs

A different kind of reduction of the search space of DAGs can be obtained by considering the concept of independent-equivalence of DAGs (Pearl and Verma 1990): two DAGs are independent-equivalent iff they represent the same set of conditional independence assertions. The graphical criterion that determines the equivalence of two DAGs (Pearl and Verma 1990) is very simple:

> Two DAGs are independent-equivalent if and only if they have the same skeleton and the same v-structures.

The *skeleton* of a DAG is the undirected graph that results from ignoring the directionality of every edge. A *v-structure* in a DAG G is an ordered triplet of nodes, (X_i, X_z, X_j), such that (1) G contains the arcs $X_i \rightarrow X_z$ and $X_z \leftarrow X_j$, and (2) the nodes X_i and X_j are not adjacent in G. If we do not force this second condition, we speak of a *h-h pattern* (head-to-head pattern) instead of a v-structure.

The graphs used to represent independent-equivalence classes of DAGs are called *partially directed acyclic graphs* (PDAGs). They may contain both directed (arcs) and undirected (links) edges, but no directed cycles. Therefore, the nodes directly connected to a given node X_i may be parents, children and also *siblings* of X_i ($Sib_G(X_i) = \{X_j \in \mathbf{V} | X_i — X_j \in E_G\}$). There are algorithms for learning (unrestricted) Bayesian networks using scoring functions that carry out the search in spaces of PDAGs, more reduced than the one of DAGs [as *completed PDAGs* (Chickering 2002) and *restricted PDAGs* (RPDAGs) (Acid and de Campos 2003)].

Another type of representation of independent-equivalence classes of DAGs, particularly interesting to our purposes, is that of C-RPDAGs, which have been used for building Bayesian classifiers (Acid et al. 2005). C-RPDAGs are defined from RPDAGs in a similar way as C-DAGs are defined from DAGs. Essentially, C-RPDAGs are RPDAGs focused on the target (class) variable, i.e. RPDAGs which are different from a classification point of view. A useful characterization of the concept of C-RPDAG is the following (Acid et al. 2005):

Proposition 1 *A PDAG $G = (\mathbf{V}, E_G)$, where $\mathbf{V} = \mathbf{U} \cup \{C\}$, is a C-RPDAG iff it satisfies the following conditions:*

1. *G does not contain any directed cycle.*
2. *If $Pa_G(C) \neq \emptyset$ then $|Pa_G(C)| \geq 2$ and $Sib_G(C) = \emptyset$.*
3. *$\forall X_i \in \mathbf{U}$, if $Pa_G(X_i) \neq \emptyset$ then $C \in Pa_G(X_i)$ and either $|Pa_G(X_i)| \geq 2$ or $|Pa_G(C)| \geq 2$.*
4. *$\forall X_i \in \mathbf{U}$, if $Sib_G(X_i) \neq \emptyset$ then $Sib_G(X_i) = \{C\}$ and $Pa_G(X_i) = \emptyset$.*

Several examples of C-RPDAGs are displayed in Fig. 1. C-RPDAGs combine the concepts of independent-equivalence and equivalence in classification to get a more reduced search space. The Markov boundary of variable C is composed in this case by the parents, children, siblings and spouses of C in the corresponding C-RPDAG.

As in the case of C-DAGs, we shall also use local search to move through the C-RPDAG space, so that we need to specify the operators to move from one configuration to another neighboring configuration. To start with, we shall use the same operators proposed in Acid et al. (2005), which are defined in terms of C's parents, C's children and the parents of C's children:

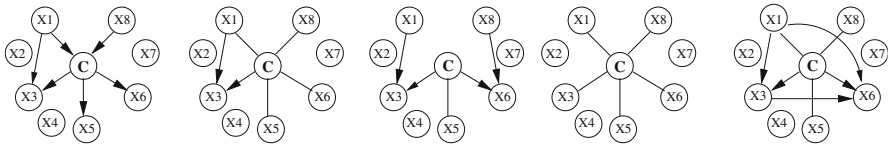

Fig. 1 Some examples of C-RPDAGs

- A_ParentOfC(X_i), addition of the arc $X_i \rightarrow C$.
- A_ChildOfC(X_i), addition of the arc $C \rightarrow X_i$.
- A_SiblingOfC(X_i), addition of the link X_i—C.
- A_HHOfC(X_i, X_j), creation of the h-h pattern $X_i \rightarrow C \leftarrow X_j$ by adding the arc $X_i \rightarrow C$ and transforming the link X_j—C into the arc $X_j \rightarrow C$.
- A_ParentOfChild(X_i, X_j), addition of the arc $X_i \rightarrow X_j$, if X_j is a child of C.
- A_HHOfChild(X_i, X_j), creation of the h-h pattern $X_i \rightarrow X_j \leftarrow C$ by adding the arc $X_i \rightarrow X_j$ and transforming the link C—X_j into the arc $C \rightarrow X_j$.
- D_ParentOfC(X_i), deletion of the arc $X_i \rightarrow C$.
- D_ChildOfC(X_i), deletion of the arc $C \rightarrow X_i$, together with all the other arcs pointing to X_i.
- D_SiblingOfC(X_i), deletion of the link X_i—C.
- D_HHOfC(X_i, X_j), destruction of the h-h pattern $X_i \rightarrow C \leftarrow X_j$ by deleting the arc $X_i \rightarrow C$ and transforming the arc $X_j \rightarrow C$ into the link X_j—C.
- D_ParentOfChild(X_i, X_j), deletion of the arc $X_i \rightarrow X_j$.
- D_HHOfChild(X_i, X_j), destruction of the h-h pattern $X_i \rightarrow X_j \leftarrow C$ by deleting the arc $X_i \rightarrow X_j$ and transforming the arc $C \rightarrow X_j$ into the link C—X_j.

The algorithms searching in the spaces of C-DAGs and C-RPDAGs (called CDAG and CRPDAG) start from an empty graph and carry out a local search using the corresponding operators until the scoring function does not improve. Then the set $MB(C)$ can be extracted from the obtained graph.

4.3 Avoiding the overfitting: using a modified search space and new operators

In Sect. 5 we will carry out experiments showing that CDAG and CRPDAG get only moderately good results. Actually, the resulting MBs are too large, the algorithms introduce many false positives, so that they are not appropriate for finding accurate MBs. This was somewhat unexpected, at least in the case of the CRPDAG algorithm, because it performs excellently when used to build a classifier (Acid et al. 2005). The conclusion is not that the score+search methods are inappropriate for learning MBs but that a metric together with a simple search space is not enough for looking for accurate MBs. In fact this tendency to choose overfitted models occurs using any of the three tested scoring functions, which assign better scores to complex configurations. This suggests that the cause lies in the search process itself. We still want to learn MBs based on score+search, but we reconsider the search space. Let us show first why this problem happens and next how we can solve it.

Let us assume that the BN displayed in Fig. 2 (a) is the correct global model and thus the correct Markov boundary is $MB(C) = \{X_1, X_2, X_5\}$. Let the configuration

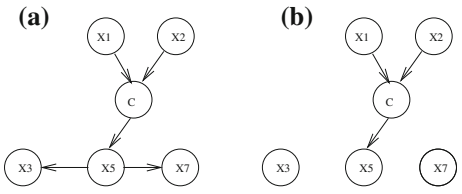

Fig. 2 **a** The correct Bayesian network, **b** an intermediate configuration in the search process

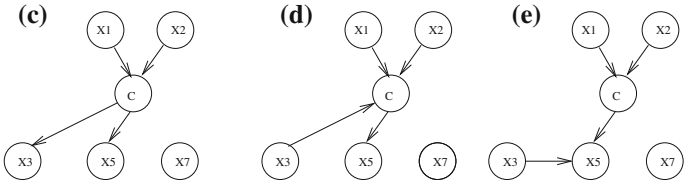

Fig. 3 Configurations in the search process where the dependence relationships between X_3 and C, and between X_3 and X_5 are reflected in several ways

(b) in the same figure be an intermediate C-DAG reached by any of the previous algorithms during the search process for learning $MB(C)$. From the true model we can extract the following independence relationships relative to the nodes C, X_3 and X_5: $\neg(X_3 \perp C|\emptyset)$, $\neg(X_3 \perp X_5|\emptyset)$ and $X_3 \perp C|X_5$. In Fig. 3 we show all the allowed configurations while exploring the neighboring configurations of the C-DAG in Fig. 2 (b), in order to consider the possible inclusion of node X_3 in $MB(C)$. All of them reflect the relationship $\neg(X_3 \perp X_5|\emptyset)$, two of them $\neg(X_3 \perp C|\emptyset)$, although none represents the relationship $X_3 \perp C|X_5$; despite of this fault, all they would improve the score. Then, one of the configurations (c), (d) or (e) in Fig. 3 would be chosen and therefore the node X_3 would be wrongly included in $MB(C)$. Observe that the arc $X_5 \rightarrow X_3$, which would give rise to a configuration satisfying all the (in)dependence relationships (and hence having a better score) will never be considered, because this configuration is not an element in our search space.

Using the same reasoning with the node X_7, it will also become connected to the target variable. Thus the final returned MB is overpopulated, $MB(C) = \{X_1, X_2, X_5, X_3, X_7\}$.

This problem, which apparently does not affect the behavior of the CRPDAG algorithm as a good classifier[2] (Acid et al. 2005), is damaging for the selection of accurate Markov boundaries.

Although the C-DAG and C-RPDAG spaces are good to represent the possible topologies of the class' MB, they are not expressive enough to represent intermediate states. By constraining the search to the C-DAG or C-RPDAG spaces, we are forcing the inclusion of some nodes in the MB because they cannot be taken into account in any other way and, from the perspective of a scoring function, it is better to connect these nodes in some way with the class variable than to leave them isolated. Therefore, one of the causes of the problem is the use of a non specialized scoring function, which measures the degree of fitness of the data to the global network. But precisely our objective in this paper is to show that even this type of score can do a good job if it is used in combination with the appropriate constrained search space, and the C-DAG or C-RPDAG spaces, although promising, are not good enough.

Therefore, it is necessary to change slightly the search spaces and to include new operators. We propose to use what we call "fringed" versions of the C-DAG and C-RPDAG spaces, where arcs connecting the children or siblings of the class to other

[2] Perhaps because, to improve classification rates, it is more important to capture all the relevant features, even at the price of using also several irrelevant features, than using only some of the relevant features.

nodes are permitted. Figure 2 (a) is an example of a fringed C-DAG. In this way, when a node X_i exhibits some dependence with C but this dependency turns out to be indirect (i.e. conditional independence), we do not need to connect X_i with C: we can connect one or several children of C with X_i, thus keeping X_i outside $MB(C)$. This additional level (grandchildren of C) constitutes the fringes of the search space.

More formally, to define a fringed C-RPDAG we only have to change one of the conditions characterizing C-RPDAGs in proposition 1:

Definition 2 A PDAG $G = (\mathbf{V}, E_G)$, where $\mathbf{V} = \mathbf{U} \cup \{C\}$, is a fringed C-RPDAG (fC-RPDAG) iff it satisfies the conditions 1, 2 and 4 in proposition 1 and the following condition:

$3'$ $\forall X_i \in \mathbf{U}$, if $Pa_G(X_i) \neq \emptyset$ then either (a) $C \in Pa_G(X_i)$ and either $|Pa_G(X_i)| \geq 2$ or $|Pa_G(C)| \geq 2$, or (b) $C \notin Pa_G(X_i)$ and $\forall X_j \in Pa_G(X_i)$, $X_j \in Ch_G(C) \cup Sib_G(C)$,

where $Ch_G(C) = \{X_i \in \mathbf{U} | C \rightarrow X_i \in E_G\}$ is the set of children of C in G.

The definition of a fringed C-DAG is also very similar to that of a C-DAG:

Definition 3 A DAG $G = (\mathbf{V}, E_G)$, where $\mathbf{V} = \mathbf{U} \cup \{C\}$, is a fringed C-DAG (fC-DAG) iff it satisfies the following condition:

$$\forall X_i, X_j \in \mathbf{V}, \text{ if } X_i \rightarrow X_j \in E_G \text{ then either } X_j = C \text{ or } X_i = C \text{ or } C \rightarrow X_j \in E_G \text{ or } C \rightarrow X_i \in E_G. \tag{2}$$

In addition to the sets of parents, children, siblings and parents of children of node C, a new set of nodes becomes relevant, the set of *children of children*, denoted by CoC:

$$CoC_G(C) = \{X_i \in \mathbf{U} \setminus Ad_G(C) \mid \exists X_j \in Ch_G(C) \cup Sib_G(C) \text{ and } X_i \in Ch_G(X_j)\}.$$

where $Ad_G(C) = Pa_G(C) \cup Ch_G(C) \cup Sib_G(C)$ is the set of nodes adjacent to C in G.[3] By virtue of either the condition [3′] (for C-RPDAGs) or the condition in Eq. 2 (for C-DAGs), this is equivalent to

$$CoC_G(C) = \{X_i \in \mathbf{U} \setminus Ad_G(C) \mid Pa_G(X_i) \neq \emptyset\}.$$

We need to define two additional operators for adding and removing the new type of allowed arcs:

- *A_ChildOfChild*(X_i, X_j), addition of the arc $X_i \rightarrow X_j$ if X_i is a child of C (or a sibling of C).
- *D_ChildOfChild*(X_i, X_j), deletion of the arc $X_i \rightarrow X_j$ if X_i is a child of C (or a sibling of C).

[3] Obviously, in the case of C-DAGs, $Sib_G(C) = \emptyset$.

These operators, in the case of the fC-DAG space, can be applied only if the node X_i is a child of C,[4] since all the edges are oriented in a fC-DAG; in the case of the fC-RPDAG space the same operators can also be applied when X_i is a sibling of C.[5]

Two of the previously used operators, *D_ChildOfC*(X_i) and *D_SiblingOfC*(X_i), will be redefined as follows:

- *D_ChildOfC*(X_i), deletion of the arc $C \rightarrow X_i$, together with all the other arcs $X_j \rightarrow X_i$ such that X_j is neither a child nor a sibling of C; deletion of the arcs $X_i \rightarrow X_k$ whenever X_k is not a child of C.
- *D_SiblingOfC*(X_i), deletion of the link C—X_i; deletion of the arcs $X_i \rightarrow X_k$ whenever X_k is not a child of C.

The reason for these changes is that, once X_i is deleted as either child or sibling of C, we need also to remove the fringes emanating from X_i. Moreover, it is not necessary to remove the arcs $X_j \rightarrow X_i$ when X_j is either a child or a sibling of C, as these arcs may act as fringes of X_j after removing the edge between C and X_i.

The remaining operators do not need to be modified, although for some of them the conditions of applicability are slightly different.[6] For the sake of completeness, Table 22 in the Appendix A shows the conditions that the current fC-RPDAG must verify in order to apply each of the fourteen operators, together with the actions required to transform it into a valid neighboring fC-RPDAG. Table 23 contains the same information about the eight operators used for fC-DAGs.

In the same appendix it is described how to compute the score of the fC-RPDAG (or fC-DAG) obtained after the application of every operator by using only two local scores (see Acid and de Campos 2003; Acid et al. 2005 for a justification). Essentially, all the operators can be evaluated by computing the difference between the new and the old local score of a node and its parents (and adding it to the global score of the current model). So, the algorithms proceed by evaluating at each step all the possible operators and selecting the one producing the greatest difference of local scores. If this difference is positive, the corresponding operator is applied to the current graph, and the cycle continues until no operator gets a positive difference. The Markov boundary is again composed by the parents, children, siblings and spouses of C in the final graph, and the rest of connections (the fringes) are ignored. We will call DMB and RPDMB to the new algorithms searching for MBs using the spaces of fC-DAGs and fC-RPDAGs, respectively.

4.4 Improving the efficiency

Although the previous algorithms equipped with the new search spaces and operators perform quite well, resulting in accurate Markov boundaries, their major drawback is the lack of efficiency due to the effort invested in the assessment of a big amount of

[4] And X_j is not a child of C; otherwise the operators to be considered would be *A_ParentOfChild*(X_i, X_j) and *D_ParentOfChild*(X_i, X_j).

[5] And X_j is not a sibling of C; otherwise the operators to be considered would be *A_HHOfChild*(X_i, X_j) and *D_HHOfChild*(X_i, X_j).

[6] Concretely for the operators *A_ParentOfC*, *A_ChildOfC*, *A_SiblingOfC* and *A_HHOfC*.

network configurations, many of them useless, as we shall see. Our purpose is then to increase the efficiency of the algorithms without losing (too much) in accuracy. This is achieved by means of two heuristic strategies that modify the search process, inspired by the constraint-based algorithms, which will be incorporated to the final versions of our algorithms.

The first strategy is to use a *two phase exploration process*. When testing the algorithms, we found that in the first stages the operators which are actually applied (improving the scoring function the most) are almost exclusively the operators for addition, that allow to grow the model reflecting the found dependencies. Nevertheless, at every step all the deletion operators are also considered and thus we spend time evaluating configurations which will be, almost surely, useless. Taking this into account, we decomposed the exploring process into two consecutive phases. 1) A *growing phase* that expands the current configuration while improving the metric (only the addition operators are considered); followed by 2) a *shrinking phase* that reviews every connection in the current configuration (only the deletion operators are considered) in order to assess simpler configurations not explored, again while the metric improves. These two phases can follow one another repeatedly until no improvement is achieved any more.

The second strategy is to *prune bad candidates* in the exploration process. In many constraint-based algorithms for learning Bayesian networks, once two variables have been found to be independent given some subset of variables, the corresponding edge connecting them is removed from the graph, and its inclusion is never reconsidered. Similarly, we have designed a way of definitively discarding candidate edges within our search spaces, thus pruning many branches in the exploration process.

Until now, at any step, every valid operator is tentatively applied to the current model and the new configurations are then assessed, giving to each one a scoring value. Among all these configurations, only the one having the highest score is recorded, the rest are disregarded. Some of these configurations may also improve the score of the current model (although to a lesser extent than the best configuration), whereas other even can make the score worse. Remember that any operator can be evaluated by computing the difference of two local scores. For the addition operators, if this difference is negative (and therefore the global score gets worse), this may be interpreted as the existence of an independence relationship between the two connected nodes (Cowell 2001; Gámez et al. 2007; Moral 2004). Therefore, those addition operators which turn the score worse obviously constitute a bad choice, and we could record this information in order to avoid to reconsider them definitively. More precisely, each time that the evaluation of an addition operator[7] connecting two nodes gives a negative difference of local scores, then we include the edge between these nodes in a list of forbidden links. In this way we are dynamically including structural restrictions in the learning process (de Campos and Castellano 2007), and consequently saving computation effort.

Similarly, for the deletion operators, if the difference of local scores is positive, this means that the existing edge connecting the involved nodes should be removed,

[7] Excluding the operator A_ChildOfChild, because its behavior is different from the rest of operators which have a direct relation with the class.

i.e. it again points out to an independence relationship and, therefore, we put the edge between these nodes in the list of forbidden links. In this way, if the operator is eventually applied, the removed edge will never be reconsidered for inclusion.

In Algorithm 1 we show an outline of the RPDMB algorithm, searching in the space of fC-RPDAGs and using the two previous strategies. The algorithm returns the Markov boundary of the variable C and optionally the topology of the subgraph induced by the Markov boundary (i.e. we remove the fringes and obtain the associated C-RPDAG). If desired, this network can also be directly used as a classification device, once the corresponding probability distributions are estimated from the data. The same pseudocode is valid for the algorithm DMB searching in the fC-DAG space, simply changing the function $EvalBestOperator$ by its counterpart using the operators for fC-DAGs.

Algorithm 1 RPDMB

Require: an initial empty network G_0 whose set of variables is $\mathbf{U} \cup \{C\}$
Require: a data stream D
Require: a struct $operator = \{value, op, nodex, nodey\}$.
Require: a vector of forbidden connections, $tabu[\,]$.
Require: a variable, $phase$, to control the phase in development
Require: 2 flags $change1$, $change2$ to record if a change occurred at each phase
Require: 3 flags $stop$, $stop1$, $stop2$ to control the end of the search and the two phases
Ensure: mb, Markov Boundary for C and graph G

$G \leftarrow G_0, stop \leftarrow false$
while $(stop = false)$ **do**
  $phase \leftarrow 1, change1 \leftarrow false$
  $stop1 \leftarrow false, stop2 \leftarrow true$
  while $(stop1 = false)$ **do** // phase 1, apply only operators for adding
    $operator = EvalBestOperator(G, C, D, phase, tabu[])$
    if $(operator \neq null)\ AND\ (operator.value > 0)$ **then**
      $change1 \leftarrow true, stop2 \leftarrow false$
      $G \leftarrow ApplyOperation(G, operator)$
    else
      $stop1 \leftarrow true$
    end if
  end while
  $phase \leftarrow 2, change2 \leftarrow false$
  while $(stop2 = false)$ **do** // phase 2, apply only operators for deleting
    $operator = EvalBestOperator(G, C, D, phase, tabu[])$
    if $(operator \neq null)\ AND\ (operator.value > 0)$ **then**
      $change2 \leftarrow true$
      $G \leftarrow ApplyOperation(G, operator)$
    else
      $stop2 \leftarrow true$
    end if
  end while
  if $(change2 = false)$ **then**
    $stop \leftarrow true$
  end if
end while
$mb \leftarrow ObtainMB(G, C)$
return (mb, G)

Table 1 Details of the Bayesian networks used in the experiments with synthetic data

Networks	N. vars	N. arcs	N. states (avg.)
Alarm	37	46	2–4 (2.8)
Barley	48	84	2–67 (8.8)
Boblo	23	24	2–6 (2.7)
Hailfinder	56	66	2–11 (3.3)
Insurance	27	52	2–5 (3.3)
Pigs	441	592	3 (3.0)

5 Experiments and analysis

We have designed several experiments over a wide spectrum of databases, with several sample sizes and running a variety of algorithms to compare with. The selected algorithms are the already mentioned PCMB, KIAM and IAMB (as state-of-the-art algorithms), CDAG and CRPDAG (because our algorithms are based on them), as well as the proposed algorithms DMB and RPDMB, using three different metrics: BDeu, BIC/MDL and MIT. We have also used a state-of-the-art FSS method, not based on Markov boundaries, namely mRMR (Ding and Peng 2005). All the score+search-based algorithms have been implemented on JAVA in the Weka platform (Hall et al. 2009). Moreover, we used the C++ implementation of PCMB, KIAMB and IAMB provided by J.M Peña[8] and the C++ implementation of mRMR provided by Peng.[9]

5.1 The data

We have used synthetic databases sampled from six well-known Bayesian networks: Alarm (Beinlich et al. 1989), Barley (Kristensen and Rasmussen 2002), Boblo (Rasmussen 1995), Hailfinder (Abramson et al. 1996), Insurance (Binder et al. 1997) and Pigs (Jensen 1997), all of them (except Boblo) available from the Bayes net repository[10]; they all come from very different domains such as medical diagnosis, insurance risk, meteorology and agriculture. Table 1 gives some details on the networks from which the databases are sampled, including the number of variables and arcs that they contain, as well as the minimum, maximum and average number of states per variable.

Each network has been used to generate several databases with different sample sizes; more precisely, for each network and each size we have generated five datasets, and the sample sizes considered are 100, 200, 500, 1000 and 10,000. Therefore, we have used a total of 150 databases.

An experimental unit consists in learning the Markov boundary for a variable X from a dataset of a given size in a domain problem, using X as the target variable and then comparing the learned MB with the true MB of X. The learning process is then repeated for each of the variables in the domain and each of the datasets. This

[8] http://www.ida.liu.se/~jospe/.

[9] http://penglab.janelia.org/proj/mRMR/.

[10] http://www.cs.huji.ac.il/labs/compbio/Repository.

Table 2 Details of the databases selected from the genetic domain

Alias	Full name	N. attr.	N. train	N. test
ALL-AML (Golub 1999)	Leukemia-ALLAML (WhiteHead, MIT)	7,129	38	34
BCR-ABL (Yeoh 2002)	Leukemia-subtype (Stjude)	12,558	215	112
LungCancer (Gordon 2002)	LungCancer (Brigham and Women's Hospital, Harvard Medical School)	12,533	32	149
MLL (Armstrong 2002)	Leukemia-MLL (WhiteHead, MIT)	12,582	57	15
ProstateCancer (Singh 2002)	Prostate Cancer (Tumor vs Normal)	12,600	102	34
ColonTumor (Alon 1999)	Colon Tumor	2,000	62	–
DLBCL-Stanford (Alizadeh 2000)	DLBCL (Stanford)	4,026	47	–
CNS (Pomeroy 2002)	Central Nervous System	7,129	60	–
DLBCL-Harvard (Shipp 2002)	DLBCL (Harvard Medical School, Whitehead, MIT) Outcome	6,817	58	–

represents a total of 15,800 experimental units. Each one of these experimental units has been used by the different algorithms commented previously. The results that we will show are the averages, across the number of variables and across the five datasets of the same size for each domain, of the selected performance measures.

Due to the extension of the results for the synthetic data (six domain problems, five sample sizes, together several algorithms and scoring functions), we will display only part of them in the main body of the paper, although the rest can be found in the Appendix B.

We have also chosen databases from the genetic domain used for classification purposes. The nine selected databases[11] represent real problems with a large number of attributes and one target variable. In Table 2 we can see some details of these datasets, as the number of attributes and the sizes of the training and test sets. Observe that test files are not provided in all the cases.

5.2 Performance measures

In the experiments with synthetic data, to compare the learned and the true Markov boundaries we use the criteria of *precision*, *recall*, and two different aggregated performance measures from the previous ones, that we shall mention later. Precision is the number of true positives returned by the algorithm (i.e. the number of variables in the learned MB which are also in the true MB), divided by the number of variables in the output (the size of the learned MB). Recall is the number of true positives in the output, divided by the size of the true MB. Precision is a measure of exactness, whereas recall is a measure of completeness, both ranging between 0.0 and 1.0. A high value for precision indicates that most of the features selected by the algorithm are correct, whereas a low value suggests that we have wrongly included many

[11] The source site for the data is the Institute for Infocomm Research http://datam.i2r.a-star.edu.sg/datasets/krbd.

variables in MB. On the other hand, a high recall means that the algorithm has selected a large proportion of the attributes that are part of the true MB. None of these two measures makes sense in isolation from each other. For example, a non selective blind algorithm that systematically includes all the features in MB (hence the true positives are necessarily included) would obtain maximum recall but a very low precision; at the other extreme, a very selective algorithm that returns a MB composed of only a single true positive feature would get maximum precision, although the recall could be very low if the true MB contains many features.

So, it is necessary to consider both measures and combine them into a unique performance measure. In Peña et al. (2007) the *Euclidean distance* from perfect precision and recall was proposed:

$$Ed = \sqrt{(1 - precision)^2 + (1 - recall)^2} \tag{3}$$

This measure ranges from 0.0 (correct MB) to $\sqrt{2.0}$. A much more standard way of combining these two measures is the F measure,[12] which is the *harmonic mean* of precision and recall:

$$F = \frac{2 * precision * recall}{precision + recall} \tag{4}$$

F ranges between 0.0 and 1.0, with $F = 1$ meaning that a totally precise and complete MB has been found. We shall only use the Euclidean distance occasionally, because no significant differences have been appreciated with respect to the F-measure, and we prefer F because it is more commonly used.

In the experiments with real genetic data we do not know the true Markov boundary of the class variable. Therefore we will measure the quality of the algorithms indirectly: by building a classifier from the training set, using *only* the attributes found in the corresponding learned MBs, and then measuring the predictive accuracy (the percentage of successful predictions), either on the test set or using the leave-one-out technique[13] in the cases where no test set is provided.

5.3 Experimental results

We have divided our experiments into five groups, each one trying to analyze a different aspect of the problem.

5.3.1 Comparing the score+search algorithms

First, we are going to compare the four score+search algorithms considered in this paper, CDAG and CRPDAG against the two new algorithms DMB and RPDMB. In this

[12] Which is so frequently used in the information retrieval and classification fields.

[13] For a data set with n samples, this means running n learning processes, each one using $n - 1$ training samples and a single sample to classify, and computing the average accuracy.

Table 3 Precision values for the algorithms based on score+search, for databases of size 1,000

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
CDAG	0.5250	0.6031	0.2959	0.5587	0.5003	0.6524
CRPDAG	0.2657	0.4923	0.1957	0.4116	0.3230	0.2894
DMB	0.8801	**0.6989**	**0.6053**	**0.8428**	0.8759	0.9939
RPDMB	**0.9178**	0.6953	0.6047	0.8307	**0.8823**	**0.9987**

Table 4 Recall values for the algorithms based on score+search, for databases of size 1,000

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
CDAG	0.8663	0.4510	0.7070	0.4913	0.6013	0.9914
CRPDAG	**0.9512**	**0.4885**	**0.7983**	**0.7601**	**0.8168**	**1.0000**
DMB	0.8210	0.2793	0.5443	0.5395	0.5773	0.9210
RPDMB	0.8514	0.3077	0.6961	0.5591	0.5961	**1.0000**

Table 5 F values for the algorithms based on score+search, for databases of size 1,000

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
CDAG	0.6116	**0.4608**	0.4000	0.4790	0.4956	0.7445
CRPDAG	0.3831	0.4353	0.3012	0.4714	0.4159	0.3971
DMB	0.8189	0.3609	0.5448	0.6115	0.6611	0.9466
RPDMB	**0.8570**	0.3923	**0.6443**	**0.6270**	**0.6810**	**0.9992**

case we have considered the versions of the algorithms using the BDeu metric (using an equivalent sample size equal to 1) and the databases of size 1,000. In Tables 3 and 4 we show the precision and the recall values of these experiments. We have emphasized in bold font the best results for each domain in all the tables.

First, looking at the precision values in Table 3 we can observe that the new algorithms get much better results than CRPDAG and CDAG, except for the Barley database where the differences are not so big. That means that the MBs returned by RPDMB and DMB contain less false positives that those of CDAG and CRPDAG, as it was expected from the analysis carried out in Sect. 4.3.

However, the best results in recall are obtained by CDAG and specially CRPDAG, i.e. the MBs returned by these algorithms contain a higher number of true positives. In general, the sizes of the MBs returned by CRPDAG and CDAG are greater (and consequently the running times invested in learning them were several times longer) than those of RPDMB and DMB. In particular, CRPDAG is scarcely selective and introduces a lot of false positives.

In order to obtain a final conclusion about the behavior of these algorithms, let us observe the values of the measure F displayed in Table 5. The best algorithms are clearly RPDMB and DMB (except in Barley), with RPDMB outperforming slightly DMB. Thus, the relative completeness of the MBs generated by CDAG and CRPDAG

Table 6 Precision values for the algorithm RPDMB combined with different scores, for databases of size 1,000

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
RPDMB+BDeu	**0.9178**	0.6953	0.6047	0.8307	0.8823	**0.9987**
RPDMB+MIT	0.8810	**0.7698**	0.7739	**0.9246**	0.8881	0.9921
RPDMB+BIC	0.9034	0.6500	**0.7811**	0.9000	**0.8890**	0.9970

Table 7 Recall values for the algorithm RPDMB combined with different scores, for databases of size 1,000

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
RPDMB+BDeu	**0.8514**	0.3077	**0.6961**	0.5591	0.5961	**1.0000**
RPDMB+MIT	0.8361	**0.3791**	0.5378	**0.7385**	**0.6101**	0.9999
RPDMB+BIC	0.7144	0.1985	0.4978	0.5872	0.4531	0.9671

Table 8 F values for the algorithm RPDMB combined with different scores, for databases of size 1,000

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
RPDMB+BDeu	**0.8570**	0.3923	**0.6443**	0.6270	0.6810	**0.9992**
RPDMB+MIT	0.8374	**0.4698**	0.6303	**0.7740**	**0.6953**	0.9950
RPDMB+BIC	0.7662	0.2843	0.6071	0.6636	0.5661	0.9778

cannot compensate their lack of precision. The more selective algorithms RPDMB and DMB generate more balanced MBs.

5.3.2 Comparing different scoring functions

Once we have shown that the proposed algorithms RPDMB and DMB outperform the other score+search method, we consider the question of the possible influence of the standard metrics used to guide the search process on the quality of the results. In other words, are there some metrics more appropriate than other to deal with this task?

In order to answer the previous question, we have carried out experiments with the databases of size 1,000, using RPDMB and the three different scoring functions already mentioned: BDeu (again with equivalent sample size $\eta = 1$), BIC/MDL and MIT (using $\alpha = 0.999$ as the parameter representing the confidence level). The results of these experiments are displayed in Tables 6, 7 and 8.

We can observe that, although the situation is more or less balanced with respect to precision, where no metric is clearly better or worse than the others, things are different with respect to recall: the BIC/MDL metric behaves almost systematically worse than both BDeu and MIT (only in one case BIC beats BDeu). This fact can also be observed in the results of the combined F measure.

Table 9 Constraint-based versus score+search algorithms

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
PCMB	**0.9380**	0.3760	0.6435	*0.7323*	*0.8661*	*0.9837*
KIAMB	0.8580	0.5638	0.6855	0.5679	0.8121	0.6989
IAMB	0.8983	*0.6167*	*0.7450*	0.5890	0.8576	0.7210
DMB+BDeu	0.8801	0.6989	0.6053	0.8428	0.8759	0.9939
DMB+MIT	0.8948	**0.7965**	**0.7761**	**0.9483**	**0.8909**	0.9830
RPDMB+BDeu	*0.9178*	0.6953	0.6047	0.8307	0.8823	**0.9987**
RPDMB+MIT	0.8810	0.7698	0.7739	0.9246	0.8881	0.9921

Precision on databases of size 1,000

Table 10 Constraint-based versus score+search algorithms

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
PCMB	*0.7894*	**0.6158**	0.2522	*0.6313*	0.5279	*0.9999*
KIAMB	0.7622	0.2108	0.4130	0.5173	0.5570	0.9009
IAMB	0.7886	0.2426	*0.4543*	0.5323	*0.5795*	0.9258
DMB+BDeu	0.8210	0.2793	0.5443	0.5395	0.5773	0.9210
DMB+MIT	0.7730	0.3400	0.3326	0.6051	0.5934	0.8795
RPDMB+BDeu	**0.8514**	0.3077	**0.6961**	0.5591	0.5961	**1.0000**
RPDMB+MIT	0.8361	*0.3791*	0.5378	**0.7385**	**0.6101**	0.9999

Recall on databases of size 1,000

Therefore, we conclude that, at least from the perspective of learning accurate MBs using score+search methods, we should discard BIC/MDL and use either BDeu or MIT instead.

5.3.3 Comparing score+search versus constraint-based algorithms

In order to have an idea of the relative accuracy of our proposed algorithms based on score+search, we want to compare them with some algorithms from the constraint-based approach, namely, PCMB, KIAMB and IAMB.

We have performed a comparative study of these three algorithms versus four instances of our proposed algorithms: DMB and RPDMB equipped with the BDeu and MIT metrics. In Tables 9 and 10 we display the precision and recall values obtained by learning from datasets of 1,000 samples. The best absolute results for each domain are still displayed in bold font, and the best results within the other block (either constraint-based or score-based) are represented in italic font.

In Table 9 we can see that better precision is reached by the score-based algorithms most of the time (and the best precision in five from the six domains), in particular by the algorithm DMB+MIT (best in four from the six domains). Within the constraint-based block, the algorithm PCMB has a better behavior in four from the six domains.

Table 11 Constraint-based versus score+metric algorithms

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
Euclidean						
PCMB	0.2432	*0.7904*	1.0333	*0.4985*	0.5389	*0.0163*
KIAMB	0.2827	0.9357	0.7692	0.6479	0.4936	0.3170
IAMB	*0.2394*	0.8810	*0.6995*	0.6227	*0.4544*	0.2887
DMB+BDeu	0.2197	0.8333	0.6147	0.4928	0.4472	0.0792
DMB+MIT	0.2605	0.7118	0.8002	0.3983	0.4323	0.1217
RPDMB+BDeu	**0.1728**	0.8106	**0.5126**	0.4791	0.4269	**0.0013**
RPDMB+MIT	0.2168	**0.6865**	0.6406	**0.2725**	**0.4176**	0.0079
F-measure						
PCMB	*0.8363*	*0.4268*	0.3398	*0.5863*	0.6352	*0.9902*
KIAMB	0.7819	0.2901	0.4794	0.4907	0.6293	0.7431
IAMB	0.8136	0.3295	*0.5264*	0.5104	*0.6602*	0.7661
DMB+BDeu	0.8189	0.3609	0.5448	0.6115	0.6611	0.9466
DMB+MIT	0.7977	0.4415	0.4796	0.6897	0.6801	0.9137
RPDMB+BDeu	**0.8570**	0.3923	**0.6443**	0.6270	0.6810	**0.9992**
RPDMB+MIT	0.8374	**0.4698**	0.6303	**0.7740**	**0.6953**	0.9950

Euclidean distance and F measures on databases of size 1,000

Observing Table 10, we can see that in general the recall values are lower than their respective precision values. This means that, although the algorithms may be very precise, the returned MBs may lose many true positives (except for the case of the Pigs domain). The patterns for recall are similar to those observed for precision: the algorithms based on score+search get in general better recall values than the constraint-based. The algorithms that get the most complete MBs are RPDMB+BDeu and RPDMB+MIT. In the block of constraint-based algorithms, PCMB usually reaches the best values.

The aggregated performance measures from precision and recall, i.e. Euclidean distance[14] and F-measure, are displayed in Table 11. As we can observe, the algorithms that reach the best values for Euclidean distance match those that reach the best values for F, so that it seems that there are not very important differences in the behavior of these two measures. In this case the best values are always obtained by the score-based algorithms (RPDMB+BDeu and RPDMB+MIT are each the best on three of the six domains). From the 72 pairwise comparisons between a score-based and a constraint-based algorithm (using the F measure), in 64 cases score-based algorithms are the winners. The results of the constraint-based methods are particularly bad in the Boblo and Hailfinder domains, and in general PCMB obtains the best results in this block.

Concerning the computational costs of the different algorithms, Table 12 displays their running times (in seconds). It seems that constraint-based methods are, in general, several times faster than score-based ones, although in most cases these times

[14] We use it here because this measure was proposed by the authors of PCMB and KIAMB.

Table 12 Constraint-based versus score+metric algorithms

Time (s)	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
PCMB	0.0399	0.1220	0.0176	0.0858	0.0472	6.6886
KIAMB	0.0249	0.0244	0.0135	0.0394	0.0197	0.3651
IAMB	0.0229	0.0243	0.0132	0.0382	0.0185	0.3793
DMB+BDeu	0.1183	0.0792	0.0794	0.1400	0.0979	32.4051
DMB+MIT	0.1026	0.0995	0.0339	0.0800	0.1047	48.8053
RPDMB+BDeu	0.1498	0.1082	0.0902	0.1710	0.1320	2.9440
RPDMB+MIT	0.1661	0.2075	0.0553	0.1544	0.1544	5.1402
N. statistics						
PCMB	483	105,198	134	102,297	560	57,331
KIAMB	158	118	66	243	114	1,836
IAMB	143	116	64	226	104	1,759
DMB+BDeu	211	146	140	217	170	2,362
DMB+MIT	181	138	54	158	173	2,044
RPDMB+BDeu	256	208	160	294	207	3,022
RPDMB+MIT	266	245	107	280	222	3,124

Running times and number of different statistics computed by the algorithms on databases of size 1,000

Table 13 Pairwise comparison between score-based and constraint-based algorithms

Sample size	100	200	500	1,000	10,000	Total
Times score-based better	50	50	56	64	49	269
Times constraint-based better	22	22	16	8	15	83

are rather small (less than one second). As the implementation of the constraint-based and score-based algorithms is quite different, we have also used another crude way of measuring complexity: the number of different statistics truly computed *from the data*, in order to calculate either the result of a statistical test or a score.[15] Table 12 also displays these numbers. In general terms, the advantage is still for constraint-based methods, but the differences are much smaller.

In order to assess the sensitivity of the algorithms under different circumstances, we have carried out the same experiments using four different sample sizes, ranging from small (100) to rather large sample sizes (10,000). The results of these experiments are displayed in Tables 24, 25, 26 and 27 in the Appendix B. In addition to the expected conclusion that the performance of all the algorithms deteriorates systematically when decreasing the sample size,[16] we can observe essentially the same pattern as before, where score-based methods tend to outperform constraint-based ones. A comparative summary between score-based and constraint-based methods can be seen in Table 13,

[15] Note that this is usually the most costly process for scoring-based and constraint-based learning algorithms, as it requires accessing to the dataset and computing frequencies.

[16] The exception is the dataset Pigs with almost all algorithms, where performance decreases when changing from size 1,000 to 10,000.

Table 14 Number of times that the algorithm in row i is better than the algorithm in column j

	PCMB	KIAMB	IAMB	DMB+ BDeu	DMB+ MIT	RPDMB+ BDeu	RPDMB+ MIT	Total better
PCMB	–	18	13	12	13	10	5	71
KIAMB	10	–	0	3	6	2	3	24
IAMB	15	30	–	6	12	5	5	73
DMB+BDeu	16	27	24	–	19	4	5	95
DMB+MIT	15	24	18	11	–	8	0	76
RPDMB+BDeu	18	28	25	26	22	–	12	131
RPDMB+MIT	23	27	25	25	30	18	–	148
Total worse	97	154	105	83	102	47	30	

based on pairwise comparisons between each score-based and each constraint-based algorithm, for each sample size. In global terms, 76% of times score-based methods are better than constraint-based.

Going into more details, in Table 14 we compare each algorithm with the others.[17] The entry in row i column j represents the number of times that algorithm i is better than algorithm j. Each row displays the times that the corresponding algorithm is better, whereas each column says how many times the algorithm is worse.

The results are quite conclusive. The best algorithm is clearly RPDMB (with both the MIT and BDeu metrics), followed by DMB+BDeu. DMB+MIT is always outperformed by RPDMB+MIT, and the same happens with KIAMB and IAMB. In turn, IAMB and PCMB are roughly comparable.

5.3.4 Learning MBs for real problems with high dimensional data

In order to check the capabilities of the algorithms for learning Markov boundaries in high dimensionality problems, we have used the real genetic databases described in Sect. 5.1. We measure indirectly the quality of these MBs by computing the predictive accuracy obtained by training a classifier using only the attributes that belong to the Markov boundaries.

The algorithms used in these experiments are IAMB and PCMB from the constraint-based paradigm and RPDMB+BDeu and RPDMB+MIT from the score-based approach. The baseline classifier considered is naive Bayes (Langley et al. 1992). As it is known, it sets all the selected attributes dependent on the class variable and also assumes conditional independence of the attributes given the class. However, the C-RPDAG returned by RPDMB can be used directly as a classifier, once the corresponding probability distributions are estimated from the data; we will also use it to compute the predictive accuracy (thus evaluating how this more complex topology may affect the performance). The results of these experiments are displayed in

[17] The number of comparisons between each pair of algorithms is 30 (six domains and five sample sizes), except in those cases involving PCMB, where this number is 28. This happens because we could not run the PCMB algorithm with two of the databases of size 10,000.

Table 15 Predictive accuracy for the genetic databases

	Naive Bayes				C-RPDAG	
	IAMB	PCMB	BDeu	MIT	BDeu	MIT
ALL-AML	**91.18**	**91.18**	*85.29*	**91.18**	**91.18**	**91.18**
BCR-ABL	92.86	94.64	*98.21*	94.64	**99.11**	94.64
LungCancer	79.19	10.07	**97.99**	81.88	*89.93*	81.88
MLL	*93.33*	80.00	86.67	*93.33*	**100.00**	*93.33*
ProstateCancer	26.47	41.18	26.47	**73.53**	29.41	*70.59*
ColonTumor	**82.26**	*80.65*	*80.65*	79.03	*80.65*	79.03
DLBCL-Stanford	80.85	72.34	**97.87**	78.72	*93.62*	74.47
CNS	**78.33**	*76.67*	65.00	63.33	60.00	60.00
DLBCL-Harvard	**74.14**	63.79	68.97	*72.41*	*72.41*	*72.41*
Average	77.62	67.84	78.57	**80.89**	79.59	*79.73*
SD	20.39	26.78	23.02	10.57	22.89	11.71

Table 16 Number of features selected by each algorithm for the genetic databases

	IAMB	PCMB	BDeu	MIT
ALL-AML	1	1	9	1
BCR-ABL	3	1	46	5
LungCancer	1	0	13	1
MLL	3	193	40	1
ProstateCancer	4	1	130	6
ColonTumor	3.92	1.76	21.48	3.00
DLBCL-Stanford	2.66	0.77	28.23	2.70
CNS	3.07	3.28	32.12	2.17
DLBCL-Harvard	6.00	6.03	21.02	5.88
Average	3.07	23.09	37.87	3.08

Table 15. The best results for each domain are displayed in bold font, and the second best in italic font. The number of features selected by each algorithm for each database is displayed in Table 16.

In general, the differences between the different algorithms are not so clear as in the previous experiments, although in some cases their behavior is quite different. Overall, the worst algorithm is PCMB (it obtains an average accuracy 10% lesser than the others and the highest standard deviation), but there is not a clear winner. IAMB gets more frequently (4 times from 9) the best results, but all the score-based algorithms obtain better average accuracy. The standard deviation of the score-based algorithms using MIT is considerably lower than in the other cases, and they also get the best average accuracies, which suggests a more stable behavior.

With respect to the use as classifier of the more complex structure (the C-RPDAG) returned by RPDMB instead of naive Bayes, it is clearly positive in the case of using BDeu (better average accuracy, lower standard deviation, better results in five from

Table 17 Running times of each algorithm for the genetic databases

	IAMB (s)	PCMB	BDeu	MIT
ALL-AML	1.10	1.19 s	6.17 min	5.32 min
BCR-ABL	2.39	2.33 s	24.89 s	8.97 s
LungCancer	1.63	2.37 s	1.24 h	26.33 min
MLL	1.71 s	16.43 min	17.27 h	1.33 h
ProstateCancer	3.08	2.15 s	15.96 h	16.55 min
ColonTumor	0.26	0.26 s	15.64 s	4.31 s
DLBCL-Stanford	0.57	1.20 s	1.74 min	38.74 s
CNS	0.83	0.86 s	12.79 s	1.42 s
DLBCL-Harvard	0.83	0.85 s	6.44 s	0.82 s

Table 18 Predictive accuracy for the HIVA dataset

	Naive Bayes				C-RPDAG	
	IAMB	PCMB	BDeu	MIT	BDeu	MIT
HIVA	96.51	–	96.08	95.57	99.15	99.15

nine cases). In fact the C-RPDAG obtained by RPDMB+BDeu is the best classifier if we consider pairwise comparisons with the other algorithms. However, when using the MIT score the results of the C-RPDAG do not improve those of naive Bayes.

In Table 17 the running times of the different algorithms are displayed. We can observe that constraint-based methods are considerably more efficient than score-based ones in most of the cases. BDeu is less efficient than MIT, although this is probably due to the greater size of the MBs generated by BDeu (see Table 16).

A possible explanation for the difference between the results obtained for the synthetic and the genetic databases is the number of attributes and samples used in both cases (the number of attributes is small compared with the number of samples in the synthetic databases, whereas in the genetic datasets the number of attributes is much greater than the number of samples). Another possible explanation lies in the different performance measures used in each case (F measure and predictive accuracy): the differences between constraint-based and score-based methods are greater when we use them to estimate accurate MBs[18] than when using them as FSS methods. Given the relatively high running time of RPDMB in some of the datasets, we believe that, in its current version, this algorithm is not suitable for very large-scale datasets.

Anyway we also want to test the behavior of the algorithms in situations where there are not only many variables but also many samples. To this end we have used the HIVA dataset, which contains 1,617 variables and 42,678 samples (half for training and half for testing). HIVA is a chemoinformatics dataset where the objective is to predict which compounds are active against the AIDS HIV infection.[19] The results obtained in this

[18] Which are more useful in learning causal models.

[19] This dataset has been previously used, among others, in the active learning challenge of the Pascal2 network of excellence and is available at http://www.causality.inf.ethz.ch/al_data/HIVA.html.

Table 19 F values of mRMR, RPDMB+BDeu and RPDMB+MIT, for the synthetic databases of size 100, 200, 500, 1000 and 10,000

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
Size 100						
mRMR	0.5019	0.2226	0.3817	0.3490	0.4671	0.4227
RPDMB+BDeu	0.5423	0.2228	0.4124	0.3152	0.4236	0.7530
RPDMB+MIT	0.4310	0.2618	0.4285	0.5404	0.3451	0.7326
Size 200						
mRMR	0.5789	0.2803	0.3926	0.3577	0.5018	0.4315
RPDMB+BDeu	0.6915	0.2831	0.4535	0.3867	0.4696	0.9273
RPDMB+MIT	0.5477	0.3008	0.4964	0.6745	0.4424	0.9304
Size 500						
mRMR	0.6187	0.3729	0.4822	0.3892	0.5650	0.4347
RPDMB+BDeu	0.7931	0.3368	0.5527	0.5338	0.6088	0.9879
RPDMB+MIT	0.7267	0.4332	0.5690	0.7505	0.6213	0.9896
Size 1,000						
mRMR	0.6452	0.4285	0.4983	0.4035	0.5907	0.4362
RPDMB+BDeu	0.8570	0.3923	0.6443	0.6270	0.6810	0.9992
RPDMB+MIT	0.8374	0.4698	0.6303	0.7740	0.6953	0.9950
Size 10,000						
mRMR	0.6436	0.5053	0.5791	0.3912	0.6235	0.4408
RPDMB+BDeu	0.8942	0.5700	0.7959	0.7799	0.7773	0.9985
RPDMB+MIT	0.8868	0.6727	0.7897	0.8282	0.8239	0.9979

new experiment are displayed in Table 18. We can observe that in this case the use of the more complex C-RPDAG structure is positive for both BDeu and MIT, which outperform the result obtained by IAMB. PCMB could not be run on this dataset.

5.3.5 Comparing with mRMR

Finally, we are going to compare our algorithms with an extra state-of-the-art method for feature selection coming outside the Markov boundary field. We have selected[20] the minimum Redundancy Maximum Relevance (mRMR) algorithm (Ding and Peng 2005; Peng et al. 2005). This is a filter method which does not rank variables independently but it tries to find a balance between relevance (dependence between the attributes and the class) and redundancy (dependence between attributes). A difficulty with mRMR is that it requires the number of variables to be selected as the input parameter. Therefore, in our experiments we have used, depending on the case, different values for this parameter.

Table 19 displays the results obtained by mRMR, measured with the F values, when it is applied to the synthetic datasets. For the sake of clarity, and in order to facilitate

[20] Following the recommendation of a reviewer.

Table 20 Predictive accuracy of mRMR for the genetic databases

ALL-AML	82.35
BCR-ABL	98.21
LungCancer	97.32
MLL	86.67
ProstateCancer	70.59
ColonTumor	90.32
DLBCL-Stanford	95.74
CNS	81.67
DLBCL-Harvard	87.93

Table 21 Predictive accuracy of mRMR for the HIVA dataset, for different number n of selected variables

n	1	3	5	10	25	50	75	100
HIVA	96.38	96.54	96.53	96.34	95.58	95.04	94.69	93.36

the comparison of mRMR with the methods based on MBs, we have also included in the table the results obtained by two of these methods, namely RPDMB using the BDeu and MIT scores. In this case, the number of variables used by mRMR has been fixed to the size of the true MB in each experimental unit. Notice that we are giving to mRMR a valuable information that will not be known by RPDMB, which has to estimate the size of the MB.

We can observe that RPDMB gets better results than mRMR most of the time, more precisely 25 and 26 times from the 30 trials when using BDeu and MIT, respectively.

For the genetic databases we have also used mRMR in combination with a naive Bayes classifier trained only with the attributes selected. In this case the parameter representing the number of variables to be selected by mRMR has been fixed to three, which is the average number of variables included in the MBs of the class variables by the algorithms IAMB and RPDMB+MIT (see Table 16). The obtained results are displayed in Table 20.

In these experiments mRMR obtains good results, outperforming all the MB-based methods in three from the nine datasets. In the other six databases mRMR does not perform better than either RPDMB+MIT or RPDMB+BDeu.

Finally, the results obtained by mRMR with the HIVA dataset, using in this case several values for the number of variables to be selected, are displayed in Table 21. We can observe that the predictive accuracy of mRMR and the MB-based methods which also use naive Bayes is very similar, although the use of the C-RPDAG structure is still advantageous.

6 Concluding remarks

We have proposed to perform FSS by means of methods for learning Markov boundaries based on score+search. Two different algorithms, searching in two specialized spaces by means of greedy adapted methods but using standard scoring functions, have been studied. This represents a rather novel way to tackle the problem of learning Markov boundaries, where the dominant approach is to use constraint-based methods.

The proposed algorithms, DMB and RPDMB, search in the new spaces of fringed C-DAGs and fringed C-RPDAGs, respectively, which are modifications, more adapted to the task of learning Markov boundaries, of the C-DAGs and C-RPDAGs spaces. The search methods have also been modified to improve the efficiency, by defining a two phase search process and a pruning strategy, inspired by the constraint-based algorithms.

We have tested our algorithms through a series of experiments with a variety of synthetic and real domains, using databases with different complexity and comparing with several algorithms. In those cases where the true Markov boundaries were available (the synthetic domains), the quality of the results was measured as a combination of precision and recall (mainly the so-called F measure). When the underlying models were unknown (the real domains), we measured the quality indirectly through the predictive accuracy of the classifiers built using the attributes found in the learned Markov boundaries. The experimental results support the conclusion that the proposed score+search-based algorithms (especially RPDMB, using either the BDeu or MIT metrics) can compete favorably with the state-of-the-art constraint-based algorithms in terms of effectiveness, although constraint-based methods are more efficient.

For future work we will investigate how to modify the proposed methods in order to make them more efficient and, in consequence, able to deal with larger domain problems. Another interesting task to explore in the future would be to consider the possibility of designing hybrid methods for learning Markov boundaries, using both score-based and constraint-based approaches, in the same way as hybrid methods for learning Bayesian networks have proven successful (Tsamardinos et al. 2006).

Acknowledgments This work has been supported by the Spanish research programme Consolider Ingenio 2010 and the Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía under projects MIPRCV:CSD2007-00018 and P09-TIC-4526, respectively. We are also grateful to J.M. Peña for providing his implementation of the PCMB, KIAMB and IAMB algorithms.

Appendix A: Details of the applicability and evaluation of the operators

Tables 22 and 23 give details about the operators used for searching in the fC-RPDAG and fC-DAG spaces, respectively.

We shall also explain how an fC-RPDAG (fC-DAG respectively) obtained after applying each one of the fourteen (eight respectively) operators can be evaluated using a decomposable and score-equivalent[21] metric.

So, let G be an fC-RPDAG (fC-DAG) and G' be any fC-RPDAG (fC-DAG) obtained by applying one of the operators described in Table 22 (Table 23) to G; let g be a score equivalent and decomposable function.

(a) If the operator is A_ParentOfC(X) then
$g(G' : D) = g(G : D) - g_D(C, Pa_G(C)) + g_D(C, Pa_G(C) \cup \{X\})$
(b) If the operator is A_ChildOfC(X) then

[21] A function that gives the same score to independent-equivalent DAGs.

Table 22 The operators for searching in the space of fringed C-RPDAGs, their conditions of applicability and the required actions

Operator	Conditions	Actions
A_ParentOfC(X)	$X \notin Ad_G(C)$; $Pa_G(C) \neq \emptyset$ $X \notin CoC_G(C)$	insert($X \rightarrow C$)
A_ChildOfC(X)	$X \notin Ad_G(C)$; $Pa_G(C) \neq \emptyset$ or $X \in CoC_G(C)$	insert($C \rightarrow X$)
A_SiblingOfC(X)	$X \notin Ad_G(C)$; $Pa_G(C) = \emptyset$ $X \notin CoC_G(C)$	insert(X—C)
A_HHOfC(X, Y)	$X \notin Ad_G(C)$; $Pa_G(C) = \emptyset$ $Y \in Sib_G(C)$; $X \notin CoC_G(C)$	insert($X \rightarrow C$) delete(Y—C); insert($Y \rightarrow C$) $\forall Z \in Sib_G(C) \setminus \{Y\}$ { delete(C—Z); insert($C \rightarrow Z$)}
A_ParentOfChild(X, Y)	$X \notin Ad_G(Y)$; $Y \in Ch_G(C)$ There is no directed path from Y to X in G	insert($X \rightarrow Y$)
A_HHOfChild(X, Y)	$X \notin Ad_G(Y)$; $Y \in Sib_G(C)$ There is no directed path from Y to X in G	insert($X \rightarrow Y$) delete(C—Y) insert($C \rightarrow Y$)
D_ParentOfC(X)	$X \in Pa_G(C)$; $\lvert Pa_G(C) \rvert \geq 3$	delete($X \rightarrow C$)
D_ChildOfC(X)	$X \in Ch_G(C)$	delete($C \rightarrow X$) $\forall Z \in Ch_G(X) \setminus Ch_G(C)$ delete($X \rightarrow Z$) $\forall Z \in Pa_G(X) \setminus (Ch_G(C) \cup Sib_G(C))$ delete($Z \rightarrow X$)
D_SiblingOfC(X)	$X \in Sib_G(C)$	delete(X—C) $\forall Z \in Ch_G(X) \setminus Ch_G(C)$ delete($X \rightarrow Z$)
D_HHOfC(X, Y)	$X \in Pa_G(C)$; $Y \in Pa_G(C)$ $\lvert Pa_G(C) \rvert = 2$	delete($X \rightarrow C$) delete($Y \rightarrow C$); insert(Y—C) $\forall Z \in Ch_G(C)$ if $Pa_G(Z) = \{C\}$ { delete($Z \rightarrow C$); insert(Z—C)}
D_ParentOfChild(X, Y)	$X \in Pa_G(Y)$; $Y \in Ch_G(C)$ $\lvert Pa_G(Y) \rvert \geq 3$ or $Pa_G(C) \neq \emptyset$	delete($X \rightarrow Y$)
D_HHOfChild(X, Y)	$X \in Pa_G(Y)$; $Y \in Ch_G(C)$ $\lvert Pa_G(Y) \rvert < 3$; $Pa_G(C) = \emptyset$	delete($X \rightarrow Y$) delete($C \rightarrow Y$); insert(C—Y)
A_ChildOfChild(X, Y)	$X \in Ch_G(C) \cup Sib_G(C)$ $Y \notin Ad_G(X) \cup Ad_G(C)$ There is no directed path from Y to X in G	insert($X \rightarrow Y$)
D_ChildOfChild(X, Y)	$Y \in Ch_G(X)$; $Y \notin Ch_G(C)$ $X \in Ch_G(C) \cup Sib_G(C)$	delete($X \rightarrow Y$)

Table 23 The operators for searching in the space of fringed C-DAGs, their conditions of applicability and the required actions

Operator	Conditions	Actions
A_ParentOfC(X)	$X \notin Ad_G(C) \cup CoC_G(C)$;	insert($X \to C$)
A_ChildOfC(X)	$X \notin Ad_G(C)$;	insert($C \to X$)
A_ParentOfChild(X, Y)	$X \notin Ad_G(Y)$; $Y \in Ch_G(C)$ There is no directed path from Y to X in G	insert($X \to Y$)
D_ParentOfC(X)	$X \in Pa_G(C)$	delete($X \to C$)
D_ChildOfC(X)	$X \in Ch_G(C)$	delete($C \to X$) $\forall Z \in Ch_G(X) \setminus Ch_G(C)$ delete($X \to Z$) $\forall Z \in Pa_G(X) \setminus Ch_G(C)$ delete($Z \to X$)
D_ParentOfChild(X, Y)	$X \in Pa_G(Y)$; $Y \in Ch_G(C)$	delete($X \to Y$)
A_ChildOfChild(X, Y)	$X \in Ch_G(C)$ $Y \notin Ad_G(X) \cup Ad_G(C)$ There is no directed path from Y to X in G	insert($X \to Y$)
D_ChildOfChild(X, Y)	$Y \in Ch_G(X)$; $Y \notin Ch_G(C)$ $X \in Ch_G(C)$	delete($X \to Y$)

$g(G' : D) = g(G : D) - g_D(X, Pa_G(X)) + g_D(X, Pa_G(X) \cup \{C\})$

(c) If the operator is A_SiblingOfC(X) then
$g(G' : D) = g(G : D) - g_D(X, \emptyset) + g_D(X, \{C\})$

(d) If the operator is A_HHOfC(X, Y) then
$g(G' : D) = g(G : D) - g_D(C, \{Y\}) + g_D(C, \{X, Y\})$

(e) If the operator is A_ParentOfChild(X, Y) then
$g(G' : D) = g(G : D) - g_D(Y, Pa_G(Y)) + g_D(Y, Pa_G(Y) \cup \{X\})$

(f) If the operator is A_HHOfChild(X, Y) then
$g(G' : D) = g(G : D) - g_D(Y, \{C\}) + g_D(Y, \{C, X\})$

(g) If the operator is D_ParentOfC(X) then
$g(G' : D) = g(G : D) - g_D(C, Pa_G(C)) + g_D(C, Pa_G(C) \setminus \{X\})$

(h) If the operator is D_ChildOfC(X) then
$g(G' : D) = g(G : D) - g_D(X, Pa_G(X)) + g_D(X, Pa_G(X) \setminus \{C\})$

(i) If the operator is D_SiblingOfC(X) then
$g(G' : D) = g(G : D) - g_D(X, \{C\}) + g_D(X, \emptyset)$

(j) If the operator is D_HHOfC(X, Y) then
$g(G' : D) = g(G : D) - g_D(C, Pa_G(C)) + g_D(C, Pa_G(C) \setminus \{X\})$

(k) If the operator is D_ParentOfChild(X, Y) then
$g(G' : D) = g(G : D) - g_D(Y, Pa_G(Y)) + g_D(Y, Pa_G(Y) \setminus \{X\})$

(l) If the operator is D_HHOfChild(X, Y) then
$g(G' : D) = g(G : D) - g_D(Y, \{C, X\}) + g_D(Y, \{C\})$

(m) If the operator is A_ChildOfChild(X, Y) then

$$g(G' : D) = g(G : D) - g_D(Y, Pa_G(Y)) + g_D(Y, Pa_G(Y) \cup \{X\})$$

(n) If the operator is D_ChildOfChild(X, Y) then

$$g(G' : D) = g(G : D) - g_D(Y, Pa_G(Y)) + g_D(Y, Pa_G(Y) \setminus \{X\})$$

It is important to remark that the evaluation of the operators D_ChildOfC and D_SiblingOfC is somewhat particular, because we evaluate the effect of removing *only* either the arc $C \to X$ or the link X—C but after that we also eliminate other arcs (e.g. the fringes), in order to keep the resulting graph within the search space. For example, the strict evaluation of the operator D_SiblingOfC(X) would give rise to

$$g(G' : D) = g(G : D) - g_D(X, \{C\}) + g_D(X, \emptyset) + \sum_{Z \in Ch_G(X) \setminus Ch_G(C)} (-g_D(Z, Pa_G(Z)) + g_D(Z, Pa_G(Z) \setminus \{X\}))$$

If we evaluate these operators taking into account all the changes that their application requires, then we would almost never use them. The reason is that the other arcs involved represent true dependencies in the model and their removal almost surely would decrease the score of the graph.

Appendix B: Some additional experiments

As we described in Sect. 5, the experiments comparing score+search versus constraint-based algorithms were carried out considering different sample sizes: 100, 200, 500, 1000 and 10,000 samples. The results for 1,000 samples were already displayed there. We present here the results obtained for the remaining sample sizes.

Table 24 Results on databases of size 100

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
Precision						
PCMB	0.5428	0.1816	0.5514	0.5450	0.6249	0.8246
KIAMB	0.6469	0.1875	0.6260	0.5685	0.6976	0.7354
IAMB	**0.6929**	0.2000	0.6913	0.6149	**0.7803**	0.8104
DMB+BDeu	0.6322	0.4570	0.4966	0.5509	0.6616	0.9183
DMB+MIT	0.6712	**0.6313**	**0.7000**	**0.8453**	0.7568	0.8804
RPDMB+BDeu	0.5949	0.4514	0.3668	0.5459	0.6874	**0.9325**
RPDMB+MIT	0.6838	0.6271	0.6811	0.8351	0.7630	0.8699
Recall						
PCMB	0.3649	0.0747	0.2113	0.3561	0.3180	**0.8161**
KIAMB	0.3761	0.0341	0.2378	0.2403	0.2637	0.5194
IAMB	0.4100	0.0359	0.2630	0.2759	0.3121	0.5855
DMB+BDeu	0.5769	0.1480	0.4565	0.2311	0.3336	0.6515
DMB+MIT	0.2968	0.1523	0.2309	0.4029	0.2015	0.5700
RPDMB+BDeu	**0.5804**	0.1693	**0.5809**	0.2452	**0.3367**	0.6825
RPDMB+MIT	0.3601	**0.1794**	0.2839	**0.4477**	0.2415	0.6837

Table 24 continued

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
F measure						
PCMB	0.3861	*0.0871*	0.2884	*0.3740*	0.3863	**0.7980**
KIAMB	0.4386	0.0561	0.3306	0.3186	0.3553	0.5795
IAMB	*0.4744*	0.0592	*0.3645*	0.3576	*0.4121*	0.6481
DMB+BDeu	**0.5522**	0.2059	0.4076	0.3014	0.4022	0.7186
DMB+MIT	0.3720	0.2330	0.3837	0.4968	0.2966	0.6544
RPDMB+BDeu	0.5423	0.2228	0.4124	0.3152	**0.4236**	*0.7530*
RPDMB+MIT	0.4310	**0.2618**	**0.4285**	**0.5404**	0.3451	0.7326

Table 25 Results on databases of size 200

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
Precision						
PCMB	0.7055	0.2969	0.6609	0.6342	0.7810	**0.9668**
KIAMB	0.6829	0.4167	0.6435	0.6264	0.7409	0.7462
IAMB	**0.7623**	0.4708	0.6667	0.6507	**0.8027**	0.7947
DMB+BDeu	0.7490	0.5403	0.4997	0.6383	0.7286	0.9621
DMB+MIT	0.7609	**0.6813**	**0.7478**	0.9280	0.7930	0.9439
RPDMB+BDeu	0.7522	0.5332	0.4051	0.6286	0.7443	0.9643
RPDMB+MIT	0.7609	**0.6813**	0.7173	**0.9159**	0.7833	0.9364
Recall						
PCMB	0.4543	**0.2630**	0.2252	0.4815	0.3512	**0.9560**
KIAMB	0.4785	0.0819	0.2848	0.3749	0.3730	0.7612
IAMB	0.5412	0.0961	0.2909	0.3923	0.3952	0.8206
DMB+BDeu	0.6788	0.1766	0.4339	0.2883	**0.3986**	0.8024
DMB+MIT	0.3895	0.1818	0.2670	0.5247	0.3148	0.7832
RPDMB+BDeu	**0.7081**	0.2205	**0.5748**	0.3120	0.3736	0.9078
RPDMB+MIT	0.4796	0.2102	0.3691	**0.5969**	0.3329	0.9431
F measure						
PCMB	0.5115	*0.2380*	0.3304	0.4392	0.4651	**0.9543**
KIAMB	0.5270	0.1321	0.3733	0.4375	0.4661	0.7135
IAMB	*0.5939*	0.1531	*0.3839*	*0.4562*	**0.4991**	0.7657
DMB+BDeu	0.6658	0.2475	0.4245	0.3679	*0.4845*	0.8537
DMB+MIT	0.4716	0.2683	0.4234	0.6189	0.4221	0.8271
RPDMB+BDeu	**0.6915**	0.2831	0.4535	0.3867	0.4696	0.9273
RPDMB+MIT	0.5477	**0.3008**	**0.4964**	**0.6745**	0.4424	*0.9304*

Table 26 Results on databases of size 500

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
Precision						
PCMB	0.8356	0.3540	0.6435	0.6971	0.8351	0.9825
KIAMB	0.8102	0.6042	0.6574	0.5625	0.8207	0.7843
IAMB	**0.8710**	0.6771	0.7181	0.5896	0.8588	0.8189
DMB+BDeu	0.8391	0.6289	0.5369	0.7665	0.8381	**0.9873**
DMB+MIT	0.8330	**0.8044**	**0.7565**	**0.9479**	0.8794	0.9764
RPDMB+BDeu	0.8557	0.6125	0.5155	0.7599	0.8401	0.9955
RPDMB+MIT	0.8247	0.7896	0.7348	0.9240	**0.8826**	0.9847
Recall						
PCMB	0.6911	**0.5602**	0.2304	0.6296	0.4617	**0.9993**
KIAMB	0.6795	0.1728	0.3630	0.4209	0.4938	0.7989
IAMB	0.7173	0.1971	0.4022	0.4449	0.5184	0.8385
DMB+BDeu	0.7728	0.2330	0.5139	0.4207	**0.5259**	0.8938
DMB+MIT	0.6545	0.2688	0.2957	0.5874	0.4879	0.8694
RPDMB+BDeu	**0.7931**	0.2628	**0.6230**	0.4557	0.5187	0.9842
RPDMB+MIT	0.6986	0.3363	0.4717	**0.7055**	0.5146	0.9983
F measure						
PCMB	0.7238	*0.3946*	0.3210	*0.5377*	0.5727	*0.9893*
KIAMB	0.7119	0.2551	0.4288	0.4427	0.5876	0.7499
IAMB	*0.7574*	0.2906	*0.4755*	0.4675	*0.6156*	0.7860
DMB+BDeu	0.7749	0.3095	0.4836	0.5057	0.6188	0.9250
DMB+MIT	0.7022	0.3725	0.4463	0.6760	0.5900	0.9037
RPDMB+BDeu	**0.7931**	0.3368	0.5527	0.5338	0.6088	0.9879
RPDMB+MIT	0.7267	**0.4332**	**0.5690**	**0.7505**	**0.6213**	**0.9896**

Table 27 Results on databases of size 10,000

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
Precision						
PCMB	**0.9708**	–	0.8000	0.6573	**0.9526**	–
KIAMB	0.8700	0.6514	0.7431	0.5684	0.8456	0.5357
IAMB	0.8863	0.7320	0.8123	0.5922	0.8786	0.5434
DMB+BDeu	0.9047	**0.8298**	0.8285	0.8985	0.8828	0.9876
DMB+MIT	0.8720	0.7843	**0.8696**	0.9260	0.8575	0.9780
RPDMB+BDeu	0.9068	0.8297	0.8234	0.8770	0.9087	**0.9977**
RPDMB+MIT	0.8955	0.7699	**0.8696**	**0.9323**	0.8940	0.9969
Recall						
PCMB	0.9036	–	0.4261	0.4432	0.6951	–
KIAMB	0.8784	0.4017	0.5787	0.6714	0.7566	0.9698
IAMB	0.8915	0.4407	0.6283	0.6599	0.7681	0.9752

Table 27 continued

	Alarm	Barley	Boblo	Hailfinder	Insurance	Pigs
DMB+BDeu	0.8812	0.5344	0.6283	0.6729	0.7239	0.9191
DMB+MIT	0.8596	0.5765	0.4978	0.6689	0.7661	0.8813
RPDMB+BDeu	**0.9121**	0.5034	**0.7496**	0.7416	0.7240	**1.0000**
RPDMB+MIT	0.9110	**0.6568**	0.7261	**0.8173**	**0.8076**	**1.0000**
F measure						
PCMB	**0.9279**	–	0.5230	0.4977	0.7860	–
KIAMB	0.8606	0.4698	0.6059	0.5430	0.7730	0.6482
IAMB	0.8757	*0.5176*	*0.6635*	*0.5630*	*0.7954*	*0.6555*
DMB+BDeu	0.8727	0.5973	0.7255	0.7286	0.7667	0.9413
DMB+MIT	0.8454	0.6189	0.6403	0.7240	0.7755	0.9115
RPDMB+BDeu	*0.8942*	0.5700	**0.7959**	0.7799	0.7773	**0.9985**
RPDMB+MIT	0.8868	**0.6727**	0.7897	**0.8282**	**0.8239**	0.9979

References

Abramson B, Brown J, Murphy A, Winkler RL (1996) Hailfinder: a Bayesian system for forecasting severe weather. Int J Forecast 12:57–71

Acid S, de Campos LM (2003) Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. J Artif Intell Res 18:445–490

Acid S, de Campos LM, Castellano JG (2005) Learning Bayesian network classifiers: searching in a space of partially directed acyclic graphs. Mach Learn 59(3):213–235

Akaike H (1974) A new look at the statistical model identification. IEEE Trans Autom Control 19:716–723

Aliferis CF, Tsamardinos I, Statnikov A (2003) HITON, a novel Markov blanket algorithm for optimal variable selection. In: Proceedings of the 2003 American Medical Informatics Association annual symposium, pp 21–25

Alizadeh AA et al (2000) Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. Nature 403:503–511

Alon U et al (1999) Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc Natl Acad Sci USA 96:6745–6750

Armstrong SA et al (2002) MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia. Nat Genet 30:41–47

Beinlich IA, Suermondt HJ, Chavez RM, Cooper GF (1989) The alarm monitoring system: a case study with two probabilistic inference techniques for belief networks. In: Proceedings of the European conference on artificial intelligence in medicine, pp 247–256

Binder J, Koller D, Russell S, Kanazawa K (1997) Adaptive probabilistic networks with hidden variables. Mach Learn 29:213–244

Blum A, Langley P (1997) Selection of relevant features and examples in machine learning. Artif Intell 97(1-2):245–271

Buntine W (1991) Theory refinement of Bayesian networks. In: Proceedings of the seventh conference on uncertainty in artificial intelligence, pp 52–60

Chickering DM (2002) Learning equivalence classes of Bayesian network structures. J Mach Learn Res 2:445–498

Chickering DM, Geiger D, Heckerman D (1995) Learning Bayesian networks: search methods and experimental results. In: Preliminary papers of the fifth international workshop on artificial intelligence and statistics, pp 112–128

Cooper GF, Herskovits E (1992) A Bayesian method for the induction of probabilistic networks from data. Mach Learn 9:309–348

Cowell R (2001) Conditions under which conditional independence and scoring methods lead to identical selection of Bayesian network models. In: Proceedings of the seventeenth annual conference on uncertainty in artificial intelligence, pp 91–97
de Campos LM (2006) A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. J Mach Learn Res 7:2149–2187
de Campos LM, Castellano JG (2007) Bayesian network learning algorithms using structural restrictions. Int J Approx Reason 45(2):233–254
de Campos LM, Huete JF (2000) A new approach for learning belief networks using independence criteria. Int J Approx Reason 24(1):11–37
Ding C, Peng H (2005) Minimum redundancy feature selection from microarray gene expression data. J Bioinform Comput Biol 3:185–205
Friedman N, Geiger D, Goldszmidt M (1997) Bayesian network classifiers. Mach Learn 29:131–163
Fu S, Desmarais MC (2007) Local learning algorithm for Markov blanket discovery. Lect Notes Comput Sci 4830:68–79
Fu S, Desmarais MC (2008a) Tradeoff analysis of different Markov blanket local learning approaches. Lect Notes Artif Intell 5012:562–571
Fu S, Desmarais MC (2008b) Fast Markov blanket discovery algorithm via local learning within single pass. Lect Notes Comput Sci 5032:96–107
Gámez JA, Mateo JL, Puerta JM (2007) A Fast hill-climbing algorithm for Bayesian networks structure learning. Lect Notes Comput Sci 4724:585–597
Golub TR et al (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286:531–537
Gordon GJ et al (2002) Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. Cancer Res 62:4963–4967
Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182
Guyon I, Weston J, Barnhill S, Vapnik V (2002) Gene selection for cancer classification using support vector machines. Mach Learn 46(1–3):389–422
Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. SIGKDD Explor 11(1):10–18
Heckerman D, Meek C (1997a) Embedded Bayesian network classifiers. Technical Report MSR-TR-97-06, Microsoft Research, Redmond
Heckerman D, Meek C (1997b) Models and selection criteria for regression and classification. In: Proceedings of the thirteenth conference on uncertainty in artificial intelligence, pp 223–228
Heckerman D, Geiger D, Chickering DM (1995) Learning Bayesian networks: The combination of knowledge and statistical data. Mach Learn 20:197–243
Inza I, Merino M, Larrañaga P, Quiroga J, Sierra B, Girala M (2001) Feature subset selection by genetic algorithms and estimation of distribution algorithms—a case study in the survival of cirrhotic patients treated with TIPS. Artif Intell Med 23(2):187–205
Jensen CS (1997) Blocking Gibbs sampling for inference in large and complex Bayesian networks with applications in genetics. PhD Thesis, Aalborg University
John GH, Kohavi R (1994) Irrelevant features and the subset selection problem. In: Proceedings of the eleventh international conference on machine learning, pp 121–129
Kohavi R, John GH (1997) Wrappers for feature subset selection. Artif Intell 97(1-2):273–324
Koller D, Friedman N (2009) Probabilistic graphical models: principles and techniques. MIT, Cambridge
Koller D, Sahami M (1996) Toward optimal feature selection. In: ICML-96: Proceedings of the thirteenth international conference on machine learning, pp 284–292
Kristensen K, Rasmussen IA (2002) The use of a Bayesian network in the design of a decision support system for growing malting barley without use of pesticides. Comput Electron Agric 33:197–217
Lam W, Bacchus F (1994) Learning Bayesian belief networks. An approach based on the MDL principle. Comput Intell 10:269–293
Langley P, Sage S (1994) Induction of selective Bayesian classifiers. In: Proceedings of the tenth conference on uncertainty in artificial intelligence, pp 399–406
Langley P, Iba W, Thompson K (1992) An analysis of Bayesian classifiers. In: Proceedings of the national conference on artificial intelligence, pp 223–228
Li H, Huang H, Sun J, Lin C (2010) On sensitivity of case-based reasoning to optimal feature subsets in business failure prediction. Expert Syst Appl 37(7):4811–4821

Mamitsuka H (2003) Empirical evaluation of ensemble feature subset selection methods for learning from a high-dimensional database in drug design In: Proceedings of the third IEEE symposium on bioinformatics and bioengineering, pp 253–257
Margaritis D, Thrun S (2000) Bayesian network induction via local neighborhoods. In: Proceedings of neural information processing systems, pp 505–511
Moral S (2004) An empirical comparison of score measures for independence. In: Proceedings of the 10th IPMU international conference, pp 1307–1314
Neapolitan RE (2003) Learning Bayesian Networks. Prentice Hall
Pearl J (1988) Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufman, San Francisco
Pellet JP, Elisseeff A (2008) Using Markov blankets for causal structure learning. J Mach Learn Res 9:1295–1342
Peña JM, Björkegren J, Tegner J (2005) Growing Bayesian network models of gene networks from seed genes. Bioinformatics 21:ii224–ii229
Peña JM, Nilsson R, Björkegren J, Tegner J (2007) Towards scalable and data efficient learning of Markov boundaries. Int J Approx Reason 45(2):211–232
Peng H, Long F, Ding C (2005) Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans Pattern Anal Mach Intell 27(8):1226–1238
Pearl J, Verma T (1990) Equivalence and synthesis of causal models. In: Proceedings of the sixth conference on uncertainty in artificial intelligence, pp 220–227
Pomeroy SL et al (2002) Prediction of central nervous system embryonal tumour outcome based on gene expression. Nature 415:436–442
Ramsey J (2006) A PC-style Markov blanket search for high dimensional datasets. Technical Report CMU-PHIL-177, Carnegie Mellon University
Rasmussen LK (1995) Bayesian Network for blood typing and parentage verification of cattle. Ph.D. Thesis, Research Centre Foulum, Denmark
Rodriguesde Morais S, Aussem A (2008a) A novel scalable and data efficient feature subset selection algorithm. Lect Notes Comput Sci 5212:298–312
Rodrigues de Morais S, Aussem A (2008b) A novel scalable and correct Markov boundary learning algorithm under faithfulness condition. In: Proceedings of the 4th European workshop on probabilistic graphical models, pp 81–88
Roos T, Wettig H, Grünwald P, Myllymäki P, Tirri H (2005) On discriminative Bayesian network classifiers and logistic regression. Mach Learn 59:267–296
Schwarz G (1978) Estimating the dimension of a model. Ann Stat 6:461–464
Shipp MA et al (2002) Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. Nat Med 8(1):68–74
Sierra B, Larrañaga P (1998) Predicting survival in malignant skin melanoma using Bayesian networks automatically induced by genetic algorithms. An empirical comparison between different approaches. Artif Intell Med 14(1–2):215–230
Singh M, Provan GM (1996) Efficient learning of selective Bayesian network classifiers. In: Proceedings of the thirteenth international conference on machine learning, pp 453–461
Singh D et al (2002) Gene expression correlates of clinical prostate cancer behavior. Cancer Cell 1:203–209
Song X, Ding Y, Huang J, Ge Y (2010) Feature selection for support vector machine in financial crisis prediction: a case study in China. Expert Syst 27(4):299–310
Spirtes P, Glymour C, Scheines R (1993) Causation, prediction, and search. Springer, New York
Tsamardinos I, Aliferis CF, Statnikov A (2003a) Algorithms for large scale Markov blanket discovery. In: Proceedings of the sixteenth international Florida artificial intelligence research society conference, pp 376–380
Tsamardinos I, Aliferis CF, Statnikov A (2003b) Time and sample efficient discovery of Markov blankets and direct causal relations. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining, pp 673–678
Tsamardinos I, Brown LE, Aliferis CF (2006) The max-min hill-climbing Bayesian network structure learning algorithm. Mach Learn 65:31–78
Yang Y, Pedersen JO (1997) A comparative study on feature selection in text categorization. In: Proceedings of the fourteenth international conference on machine learning, pp 412–420
Yaramakala S, Margaritis D (2005) Speculative Markov blanket discovery for optimal feature selection. In: Proceedings of the fifth ieee international conference on data mining, pp 809–812

Yeoh EJ et al (2002) Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. Cancer Cell 1:133–143

Yishi Z, Hong X, Yang H, Gangyi Q (2009) S-IAMB algorithm for Markov blanket discovery. In: Proceedings of the Asia-Pacific conference on information processing, vol 2, pp 379–382

Zheng H, Zhang Y (2008) Feature selection for high-dimensional data in astronomy. Adv Space Res 41(12):1960–1964