# Multivariate Cluster-based Discretization for Bayesian Network Structure Learning

Ahmed Mabrouk[1], Christophe Gonzales[2], Karine Jabet-Chevalier[1], and Eric Chojnaki[1]

[1] Institut de radioprotection et de sûreté nucléaire, France
{Ahmed.Mabrouk,Karine.Chevalier-Jabet,Eric.Chojnaki}@irsn.fr
[2] Sorbonne Universités, UPMC Univ Paris 06, CNRS, UMR 7606, LIP6, France
Christophe.Gonzales@lip6.fr

**Abstract.** While there exist many efficient algorithms in the literature for learning Bayesian networks with discrete random variables, learning when some variables are discrete and others are continuous is still an issue. A common way to tackle this problem is to preprocess datasets by first discretizing continuous variables and, then, resorting to classical discrete variable-based learning algorithms. However, such a method is inefficient because the conditional dependences/arcs learnt during the learning phase bring valuable information that cannot be exploited by the discretization algorithm, thereby preventing it to be fully effective. In this paper, we advocate to discretize while learning and we propose a new multivariate discretization algorithm that takes into account all the conditional dependences/arcs learnt so far. Unlike popular discretization methods, ours does not rely on entropy but on clustering using an EM scheme based on a Gaussian mixture model. Experiments show that our method significantly outperforms the state-of-the-art algorithms.

**Keywords:** Multivariate discretization, Bayesian network learning.

## 1 Introduction

For several decades, Bayesian networks (BN) have been successfully exploited for dealing with uncertainties. However, while their learning and inference mechanisms are relatively well understood when they involve only discrete variables, their coping with continuous variables is still often unsatisfactory. One actually has to trade-off between expressiveness and computational complexity: on one hand, conditional Gaussian models and their mixing with discrete variables are computationally efficient but they definitely lack some expressiveness [12]; on the other hand, mixtures of exponentials, bases or polynomials are very expressive but at the expense of tractability [15,20]. In between lie discretization methods which, by converting continuous variables into discrete ones, can provide a satisfactory trade-off between expressiveness and tractability.

**Acknowledgment:** This work was supported by the French Institute for Radioprotection and Nuclear Safety (IRSN), the Belgium's nuclear safety authorities (Bel V) and European project H2020-ICT-2014-1 #644425 Scissor.

In many real-world applications, BNs are learnt from data and, when there exist continuous attributes, those are often discretized prior to learning, thereby opening the path to exploiting efficient discrete variable-based learning algorithms. However such an approach is doomed to be ineffective because the conditional dependences/arcs learnt during the learning phase bring valuable information that cannot be exploited by the discretization algorithm, thereby severely limiting its effectiveness. However, there exist surprisingly few papers on discretizing while learning, probably because it incurs substantial computational costs and it requires multivariate discretization instead of just a univariate one. In this direction, MDL and Bayesian scores used by search algorithms have been adapted to include multivariate discretizations taking into account the BN structure learnt so far [6,13]. But, to be naturally included into these scores, the latter heavily rely on entropy-related maximizations which, as we shall see, is not very well suited for BN learning. In [21], a non-linear dimensionality reduction process called GP-LVM combined with a Gaussian mixture model-based discretization is proposed for BN learning. Unfortunately, GP-LVM looses the random variable's semantics and the discretization does not rely on the BN structure. As a consequence, the method does not exploit all the useful information.

Unlike in BN learning, multivariate discretization has often been exploited in Machine Learning for supervised classification tasks [1,2,5,9,22]. But the goal is only to maximize the classification power w.r.t. one target variable. As such, only the individual correlations of each variable with the target are of interest and, thus, only bivariate discretization is needed. BN structure learning is fundamentally different because the complete set of conditional dependences between all sets of variables is of interest and multivariate discretization shall most often involve more than two variables. This makes these approaches not easily transferable to BN learning. In [11], the authors propose a general multivariate discretization relying on genetic algorithms to construct rulesets. However, the approach is very limited because it is designed to cope with only one target and the domain size of this variable needs to be small to keep the method tractable.

Discretizations have also been exploited in unsupervised learning (UL), but those are essentially univariate [4,8,16,17], which make them usable *per se* only as a preprocess prior to learning. However, BN learning can be related to UL in the sense that all the BN's variables can be thought of as targets whose discretized values are unobserved. This suggests that some key ideas underlying UL algorithms might be adapted for learning BN structures. Clustering is one such popular framework. In [14], for instance, multivariate discretization is performed by clustering but, unfortunately, independences between random variables are only considered given a latent variable. This limits considerably the range of applications of the method because numerous continuous variables require the latent one to have a large domain size in order to get good quality discretizations. This approach is therefore limited to small datasets and, by not exploiting the BN structure, it is best suited as a BN learning preprocess. Finally, by relying on entropy, its effectiveness for BN learning is certainly not optimal. However, here, we advocate to exploit clustering methods for discretization w.r.t. BN learning.

More precisely, we propose a new clustering-based approach for multivariate discretization that takes into account the conditional dependences among variables discovered during learning. By exploiting clustering rather than entropy, it avoids the shortcomings induced by the latter and, by taking into account the dependences between random variables, it significantly increases the quality of the discretization compared to state-of-the-art clustering approaches.

The rest of the paper is organized as follows. Section 2 recalls BN learning and discretizations. Then, in Section 3, we describe our approach and justify its correctness. Its effectiveness is highlighted through experiments in Section 4. Finally, some concluding remarks are given in Section 5.

## 2 Basics on BN Structure Learning and Discretization

Uppercase (resp. lowercase) letters $X, Z, x, z$, represent random variables and their instantiations respectively. Boldface letters represent sets.

**Definition 1.** *A (discrete) BN is a pair* $(\mathcal{G}, \boldsymbol{\theta})$ *where* $\mathcal{G} = (\mathbf{X}, \mathbf{A})$ *is a directed acyclic graph (DAG),* $\mathbf{X} = \{X_1, ..., X_n\}$ *represents a set of discrete random variables*[1]*,* $\mathbf{A}$ *is a set of arcs, and* $\boldsymbol{\theta} = \{P(X_i|\mathbf{Pa}(X_i))\}_{i=1}^{n}$ *is the set of the conditional probability distributions (CPT) of the variables* $X_i$ *in* $\mathcal{G}$ *given their parents* $\mathbf{Pa}(X_i)$ *in* $\mathcal{G}$*. The BN encodes the joint probability over* $\mathbf{X}$ *as:*

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\mathbf{Pa}(X_i)). \tag{1}$$

To avoid ambiguities between continuous variables and their discretized counterparts, letters, when superscripted by "$\circ$", e.g., $\mathring{X}, \mathring{x}$, represent variables and their instantiations prior to discretization, else they are discretized (for discrete variables, $X = \mathring{X}$ and $x = \mathring{x}$). In the rest of the paper, $n$ always denotes the number of variables in the BN, and we assume that $\mathring{X}_1, \ldots, \mathring{X}_l$ are discrete whereas $\mathring{X}_{l+1}, \ldots, \mathring{X}_n$ are continuous. $\mathring{\mathcal{D}}$ and $\mathcal{D}$ denote the input databases before and after discretization respectively and are assumed to be complete, i.e., they do not contain any missing data. $N$ refers to their number of records.

Given $\mathring{\mathcal{D}} = \{\mathring{\mathbf{x}}^{(1)}, \mathring{\mathbf{x}}^{(2)}, \ldots, \mathring{\mathbf{x}}^{(N)}\}$, BN learning consists of finding DAG $\mathcal{G}$ that most likely accounts for the observed data in $\mathring{\mathcal{D}}$. When all variables are discrete, i.e., $\mathcal{D} = \mathring{\mathcal{D}}$, there exist many efficient algorithms in the literature for solving this task. Those can be divided into 3 classes [10]: i) the search-based approaches that look for the structure optimizing a score (BD, BDeu, BIC, AIC, K2, *etc.*); ii) the constraint-based approaches that exploit statistical independence tests ($\chi^2$, $G^2$, *etc.*) to find the best structure $\mathcal{G}$; iii) the hybrid methods that exploit a combination of both. In the rest of the paper, we will focus on search-based approaches because our closest competitors, [13,6], belong to this class.

Basically, these algorithms start with a structure $\mathcal{G}_0$ (often empty). Then, at each step, they look in the neighborhood of the current structure for another

[1] By abuse of notation, we use interchangeably $X_i \in \mathbf{X}$ to denote a node in the BN and its corresponding random variable.

structure, say $\mathcal{G}$, that increases the likelihood of structure $\mathcal{G}$ given observations $\mathcal{D}$, i.e., $P(\mathcal{G}|\mathcal{D})$. The neighborhood is often defined as the set of graphs that differ from the current one only by one atomic graphical modification (arc addition, arc deletion, arc reversal). $P(\mathcal{G}|\mathcal{D})$ is computed locally through the aforementioned scores, their differences stemming essentially from different *a priori* hypotheses. More precisely, assuming a uniform prior on all structures $\mathcal{G}$, we have that:

$$P(\mathcal{G}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{G})P(\mathcal{G})}{P(\mathcal{D})} \propto P(\mathcal{D}|\mathcal{G}) = \int_{\boldsymbol{\theta}} P(\mathcal{D}|\mathcal{G},\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\mathcal{G})d\boldsymbol{\theta}, \tag{2}$$

where $\boldsymbol{\theta}$ is the set of parameters of the CPTs of a (discrete) BN with structure $\mathcal{G}$. Different hypotheses on prior $\pi$ and on $\boldsymbol{\theta}$ result in the different scores (see, e.g., [18] for the hypotheses for the BIC score used later).

When database $\mathring{\mathcal{D}}$ contains continuous variables, those can be discretized. A discretization of a continuous variable $\mathring{X}$ is a function $f : \mathbb{R} \to \{0, \ldots, g\}$ defined by an increasing sequence of $g$ cut points $\{t_1, t_2, ..., t_g\}$ such that:

$$f(\mathring{x}) = \begin{cases} 0 & \text{if } \mathring{x} < t_1, \\ k & \text{if } t_k \leq \mathring{x} < t_{k+1}, \quad \text{for all } k \in \{1, \ldots, g-1\} \\ g & \text{if } \mathring{x} \geq t_g. \end{cases}$$

Let $\mathcal{F}$ be a set of discretization functions, one for each continuous variable. Then, given $\mathcal{F}$, if $\mathcal{D}$ denotes the (unique) database resulting from the discretization of $\mathring{\mathcal{D}}$ by $\mathcal{F}$., Eq. (2) becomes:

$$P(\mathcal{G}|\mathring{\mathcal{D}},\mathcal{F}) \propto P(\mathring{\mathcal{D}}|\mathcal{G},\mathcal{F}) = P(\mathcal{D}|\mathring{\mathcal{D}},\mathcal{G},\mathcal{F})P(\mathring{\mathcal{D}}|\mathcal{G},\mathcal{F}) = P(\mathring{\mathcal{D}}|\mathcal{D},\mathcal{G},\mathcal{F})P(\mathcal{D}|\mathcal{G},\mathcal{F}),$$

Assuming that all databases $\mathring{\mathcal{D}}$ compatible with $\mathcal{D}$ given $\mathcal{F}$ are equiprobable, we thus have that:

$$P(\mathcal{G}|\mathring{\mathcal{D}},\mathcal{F}) \propto P(\mathcal{D}|\mathcal{G},\mathcal{F}) = \int_{\boldsymbol{\theta}} P(\mathcal{D}|\mathcal{G},\mathcal{F},\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\mathcal{F},\mathcal{G})d\boldsymbol{\theta}. \tag{3}$$

BN structure learning therefore amounts to find structure $\mathcal{G}^*$ such that $\mathcal{G}^* = \text{Argmax}_{\mathcal{G}}\, P(\mathcal{G}|\mathcal{D},\mathcal{F})$. Note that $P(\mathcal{D}|\mathcal{G},\mathcal{F},\boldsymbol{\theta})$ corresponds to a classical score over discrete data. $\pi(\boldsymbol{\theta}|\mathcal{F},\mathcal{G})$ is the prior over the parameters of the BN given $\mathcal{F}$. Eq. (3) is precisely the one used when discretization is performed as a preprocess before learning.

When discretization is performed while learning, like in [6,13], both the structure and the discretization should be optimized simultaneously. In other words, the problem consists of computing $\text{Argmax}_{\mathcal{F},\mathcal{G}}\, P(\mathcal{G},\mathcal{F}|\mathring{\mathcal{D}})$, where finding the best discretization amounts to find the best set of cut points (including the best size for this set) for each continuous random variable. And we have that:

$$P(\mathcal{G},\mathcal{F}|\mathring{\mathcal{D}}) = P(\mathcal{G}|\mathcal{F},\mathring{\mathcal{D}})P(\mathcal{F}|\mathring{\mathcal{D}}) \propto P(\mathcal{F}|\mathring{\mathcal{D}}) \int_{\boldsymbol{\theta}} P(\mathcal{D}|\mathcal{G},\mathcal{F},\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\mathcal{F},\mathcal{G})d\boldsymbol{\theta}. \tag{4}$$

As can be seen, the resulting equation combines the classical score on the discretized data (the integral) with a score $P(\mathcal{F}|\mathring{\mathcal{D}})$ for the discretization algorithm itself. The logarithm of latter corresponds to what [6] and [13] call $DL_{\Lambda}(\Lambda) + DL_{\mathring{\mathcal{D}}\to\mathcal{D}}(\mathring{\mathcal{D}},\Lambda)$ and $\mathcal{S}_c(\Lambda;\mathring{\mathcal{D}})$ respectively.

**Input**: a database $\mathring{\mathcal{D}}$, an initial graph $\mathcal{G}$, a score function $sc$ on discrete variables
**Output**: the structure $\mathcal{G}$ of the Bayesian network
1 **repeat**
2 | Find the best discretization $\mathcal{F}$ given $\mathcal{G}$
3 | $\{X_{l+1}, \ldots, X_n\} \leftarrow$ discretize variables $\{\mathring{X}_{l+1}, \ldots, \mathring{X}_n\}$ given $\mathcal{F}$
4 | $\mathcal{G} \leftarrow \mathcal{G}$'s neighbor that maximizes scoring function $sc$ w.r.t. $\{X_1, \ldots, X_n\}$
5 **until** $\mathcal{G}$ *maximizes the score*;

**Algorithm 1:** Our structure learning architecture.

## 3 A New Multivariate Discretization-Learning Algorithm

As mentioned earlier, we believe that taking into account the conditional dependences between random variables is important to provide high-quality discretizations. Our approach thus follows Eq. (4) and our goal is to compute $\text{Argmax}_{\mathcal{F},\mathcal{G}} P(\mathcal{G}, \mathcal{F}|\mathring{\mathcal{D}})$. Optimizing jointly over $\mathcal{F}$ and $\mathcal{G}$ is too computationally intensive a task to be usable in practice. Fortunately, we can approximate it efficiently through a gradient descent, alternating optimizations over $\mathcal{F}$ given a fixed structure $\mathcal{G}$ and optimizations over $\mathcal{G}$ given a fixed discretization $\mathcal{F}$. This suggests the BN structure learning method described as Algo. 1.

Multivariate discretization is much more time consuming than univariate discretization. As such, Line 2 could thus incur a strong overhead to the learning algorithm because the discretization search space increases exponentially with the number of variables to discretize. To alleviate this problem without sacrificing too much in accuracy, we suggest a local search algorithm that iteratively fixes the discretizations of all the continuous variables but one and optimizes the discretization of the latter (given the other variables) until some stopping criterion is met. As such, discretizations being optimized one continuous variable at a time, the combinatorics and the computation time are significantly limited. Line 2 can thus be detailed as Algo. 2.

### 3.1 Discretization Criterion

To implement Algo. 2, a discretization criterion to be optimized is needed. Basic ideas include trying to find cut points minimizing the discrepancy between the frequencies or the sizes of intervals $[t_k, t_{k+1})$. A more sophisticated approach

**Input**: a database $\mathring{\mathcal{D}}$, a graph $\mathcal{G}$, a scoring function $sc$ on discrete variables
**Output**: a discretization $\mathcal{F}$
1 **repeat**
2 | $i_0 \leftarrow$ Select an element in $\{l+1, \ldots, n\}$
3 | Discretize $\mathring{X}_{i_0}$ given $\mathcal{G}$ and $\{X_1, \ldots, X_{i_0-1}, X_{i_0+1}, \ldots, X_n\}$
4 **until** *stopping condition*;

**Algorithm 2:** One-variable discretization architecture.

consists of limiting as much as possible the quantity of information lost after discretization, or equivalently to maximize the quantity of information remaining after discretization. This naturally calls for maximizing an entropy. This is essentially what our closest competitors, [6,13], do.

But entropy may not be the most appropriate measure when dealing with BNs. Actually, consider a variable $A$ with domain $\{a_1, a_2, a_3\}$. Then, it is possible that, for some BN, $P(A = a_1) = \frac{1}{6}$, $P(A = a_2) = \frac{1}{3}$ and $P(A = a_3) = \frac{1}{2}$. With a sufficiently large database $\mathcal{D}$, the frequencies of observations of $a_1, a_2, a_3$ in $\mathcal{D}$ would certainly lead to estimate $P(A) \approx [\frac{1}{6}, \frac{1}{3}, \frac{1}{2}]$. Now, assume that the observations in $\mathcal{D}$ are noisy, say with a Gaussian noise with an *infinitely small* variance, as in Fig. 1. Then, after discretization, we shall expect to have 3 intervals with respective frequencies $\frac{1}{6}$, $\frac{1}{3}$ and $\frac{1}{2}$, i.e., intervals similar to $(-\infty, t_1)$, $[t_1, t_2)$ and $[t_2, +\infty)$ of Fig. 1. However, w.r.t. entropy, the best discretization corresponds to intervals $[-\infty, s_1)$, $[s_1, s_2)$ and $[s_2, +\infty)$ of Fig. 1 whose frequencies are all approximately equal to $\frac{1}{3}$ (entropy is maximal for equiprobable intervals). Therefore, whatever the infinitesimal noise added to data in $\mathcal{D}$, an entropy-based discretization produces a discretized variable $A$ with distribution $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ instead of $[\frac{1}{6}, \frac{1}{3}, \frac{1}{2}]$. This suggests that entropy is probably not the best criterion for discretizing continuous variables for BN learning.

Fig. 1 suggests that clustering would probably be more appropriate: here, one cluster/interval per Gaussian would provide a better discretization. *In this paper, we assume that, within every interval, each continuous random variable, say* $\mathring{X}_{i_0}$*, is distributed w.r.t. a truncated Gaussian.* Over its whole domain of definition, it is thus distributed as a mixture of truncated Gaussians, the weights of the latter being precisely the CPT of $X_{i_0}$ in the discrete BN. In particular, if $\mathring{X}_{i_0}$ has some parents, there are as many mixtures as the product of the domain sizes of the parents. The parameters of such a discretization scheme are therefore: i) a set of $g$ cut points (to define $g + 1$ intervals) and ii) a mean and a variance for each interval (to define its Gaussian). Fig. 1 actually illustrates the fact that the means of the Gaussians need not necessarily correspond to the middles of the intervals. For instance, the mean of the third Gaussian is $a_3$ whereas the third interval, $[t_3, +\infty)$, has no finite middle. Here, even finite interval middles, like that of $[t_1, t_2)$, do not correspond to the means of the Gaussians.

For each continuous random variable $\mathring{X}_{i_0}$, this joint optimization problem is really hard due to the normalization requirements that the integrals of the truncated exponential of each interval must sum to 1 (which cannot be expressed using closed-form formulas). Therefore, to alleviate the discretization computa-

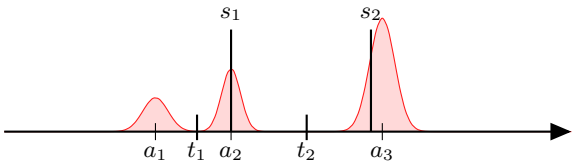


**Fig. 1.** Discretization: entropy v.s. clustering.

tional burden, we propose to approximate the computation of the cut points, means and variances using a two-step process: first, we approximate the density of the joint distribution of $\{X_1, \ldots, X_{i_0-1}, \mathring{X}_{i_0}, X_{i_0+1}, \ldots, X_n\}$ as a mixture of *untruncated* Gaussians and we determine by maximum likelihood the number of cut-points as well as the means and variances of the Gaussians. This can be easily done by an Expectation-Maximization (EM) approach. Then, in a second step, we compute the best cut points w.r.t. the Gaussians. As each Gaussian is associated with an interval, the parts of the Gaussian outside the interval can be considered as a loss of information and we will therefore look for cut points that minimize this loss. Now, let us delve into the details of the approach.

### 3.2 Discretization Exploiting the BN Structure

For the first discretization step of $\mathring{X}_{i_0}$, we estimate the number $g$ of cut-points and the Gaussians' means and variances. Assume that structure $\mathcal{G}$ is fixed and that all the other variables are discrete. The density over all the variables, $p(\mathring{\mathbf{X}})$, is equal to $p(\mathring{X}_{i_0}|\mathbf{Pa}(\mathring{X}_{i_0}))\prod_{i\neq i_0} P(X_i|\mathbf{Pa}(X_i))$, where $p(\mathring{X}_{i_0}|\mathbf{Pa}(\mathring{X}_{i_0}))$ represents a mixture of Gaussians for each value of $\mathring{X}_{i_0}$'s parents (there are a finite number of values since all the variables but $\mathring{X}_{i_0}$ are discrete). $P(X_i|\mathbf{Pa}(X_i))$ should be the CPT of discrete variable $X_i$ but, unfortunately, it is not well defined if $\mathring{X}_{i_0} \in \mathbf{Pa}(X_i)$ because, in this case, $\mathbf{Pa}(X_i)$ has infinitely many values. This is a serious issue since this CPT is used in the computation of $P(\mathcal{D}|\mathcal{G}, \mathcal{F}, \boldsymbol{\theta})$ of Eq. (4). Fortunately, this problem can be overcome by enforcing that $\mathring{X}_{i_0}$ has no child while guaranteeing that the density remains unchanged. Actually, in [19], an arc reversal operator is provided that, when applied, never alters the density/probability distribution. More precisely, when reverting arc $X \to Y$, Shachter showed that if all the parents of $X$ are added to $Y$ and all the parents of $Y$ except $X$ are added to $X$, then the resulting BN encodes the same distribution. As an example of these transformations, reversing arc $X \to V$ of Fig. 2.(a) results in Fig. 2.(b) and, then, reversing arc $X \to W$ results in Fig. 2.(c).

Therefore, to enforce that $\mathring{X}_{i_0}$ has no child, if $\{i_1, \ldots, i_c\}$ denotes the set of indices of the children variables of $\mathring{X}_{i_0}$, sorted by a topological order of $\mathcal{G}$, then, by reversing sequentially all the arcs $X_{i_j} \to \mathring{X}_{i_0}$, $j = 1, \ldots, c$, we get:

$$p(\mathring{\mathbf{X}}) = p(\mathring{X}_{i_0}|\mathbf{Pa}(\mathring{X}_{i_0})) \times \prod_{i\neq\{i_0,\ldots,i_c\}} P(X_i|\mathbf{Pa}(X_i)) \times \prod_{j=1}^{c} P(X_{i_j}|\mathbf{Pa}(X_{i_j})),$$

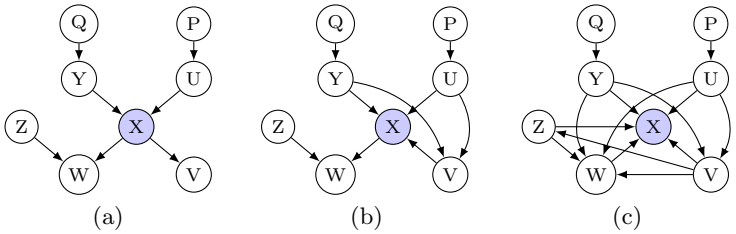


**Fig. 2.** Shachter's arc reversals.

$$p(\mathring{\mathbf{X}}) = p(\mathring{X}_{i_0}|\mathbf{MB}(\mathring{X}_{i_0})) \times \prod_{i \neq \{i_0,\ldots,i_c\}} P(X_i|\mathbf{Pa}(X_i)) \times \prod_{j=1}^{c} P(X_{i_j}|\bigcup_{h=1}^{j}(\mathbf{Pa}(X_{i_h}) \backslash \{\mathring{X}_{i_0}\}) \cup \mathbf{Pa}(\mathring{X}_{i_0})),$$

where $\mathbf{MB}(\mathring{X}_{i_0})$ is the Markov blanket of $\mathring{X}_{i_0}$ in $\mathcal{G}$:

**Definition 2.** *The Markov blanket of any node in $\mathcal{G}$ is the set of its parents, its children and the other parents of its children.*

Note that, in the last expression of $p(\mathring{\mathbf{X}})$, only the first term involves $\mathring{X}_{i_0}$, hence all the other CPTs are well defined (they are finite CPTs). As a side effect, only $p(\mathring{X}_{i_0}|\mathbf{MB}(\mathring{X}_{i_0}))$ needs be taken into account to discretize $\mathring{X}_{i_0}$ since none of the other terms is related to $\mathring{X}_{i_0}$. *It shall be noted here that these arc reversals are applied only for determining the parameters of the discretization, i.e., the set of cut points, means and variances of the Gaussians, they are never used to learn the BN structure.* Now, let us see how the parameters of the mixture of Gaussians $p(\mathring{X}_{i_0}|\mathbf{MB}(\mathring{X}_{i_0}))$ maximizing the likelihood of dataset $\mathring{\mathcal{D}}$ can be easily estimated using an EM algorithm.

### 3.3 Parameter Estimation by an EM Algorithm

Let $q_{i_0}$ represent the (finite) number of values of $\mathbf{MB}(\mathring{X}_{i_0})$. For simplicity, we will denote by $\{1,\ldots,q_{i_0}\}$ the set of values of the joint discrete random variable $\mathbf{MB}(\mathring{X}_{i_0})$. Let $g$ denote the number of cut points in the discretization and let $\{\mathcal{N}(\mu_k,\sigma_k) : k \in \{0,\ldots,g\}\}$ be the corresponding set of Gaussians. Then:

$$p(\mathring{X}_{i_0} = \mathring{x}_{i_0}|\mathbf{MB}(\mathring{X}_{i_0}) = j) = \sum_{k=0}^{g} \pi_{jk} f(\mathring{x}_{i_0}|\boldsymbol{\theta}_k) \quad \forall\ j \in \{1,\ldots,q_{i_0}\},$$

where $f(\cdot|\boldsymbol{\theta}_k)$ represents the density of the normal distribution of parameters $\boldsymbol{\theta}_k = (\mu_k,\sigma_k)$, and $\pi_{jk}$ represents the weights of the mixture (with the constraint that $\pi_{jk} \geq 0$ for all $j,k$ and $\sum_{k=0}^{g} \pi_{jk} = 1$ for all $j$). Remember that each value of $\mathbf{MB}(\mathring{X}_{i_0})$ induces its own set of weights $\{\pi_{j0},\ldots,\pi_{jg}\}$. Now, we propose to estimate parameters $\boldsymbol{\theta}_k$ from $\mathring{\mathcal{D}}$ by maximum likelihood. For this, EM is well-known to efficiently provide good approximations [3] (due to the mixture, direct maximum likelihood is actually hard to estimate). Assuming that data in $\mathring{\mathcal{D}}$ are i.i.d., the log-likelihood of $\mathring{\mathcal{D}}$ given $\boldsymbol{\Theta} = \bigcup_{k=0}^{g}(\bigcup_{j=1}^{q_{i_0}}\{\pi_{jk}\} \cup \{\boldsymbol{\theta}_k\})$ is equal to:

$$\mathcal{L}(\mathring{\mathcal{D}}|\boldsymbol{\Theta}) = \sum_{m=1}^{N} \log p(\mathring{X}_{i_0} = \mathring{x}_{i_0}^{(m)}|\mathbf{MB}(\mathring{x}_{i_0})^{(m)}, \boldsymbol{\Theta}),$$

where $\mathring{x}_{i_0}^{(m)}$ represents the observed value of $\mathring{X}_{i_0}$ in the $m$th record of $\mathring{\mathcal{D}}$. Thus:

$$\mathcal{L}(\mathring{\mathcal{D}}|\boldsymbol{\Theta}) = \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} \log\left[\sum_{k=0}^{g} \pi_{jk} f(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_k)\right]. \tag{5}$$

To solve $\text{Argmax}_{\boldsymbol{\Theta}} \mathcal{L}(\mathring{\mathcal{D}}|\boldsymbol{\Theta})$, EM [3] iteratively alternates expectations (E-step) and maximizations (M-step) until convergence toward a local maximum which is guaranteed to correspond to the Argmax we look for due to the concavity of the log-likelihood function. In this paper, we just need to apply the standard EM, considering for weights $\pi_{jk}$ only the records in the database that correspond to $\mathbf{MB}(\mathring{x}_{i_0})^{(m)} = j$. More precisely, for each record of $\mathring{\mathcal{D}}$, let $Z^{(m)}$ be a random variable whose domain is $\{0, \ldots, g\}$, and such that $Z^{(m)} = k$ if and only if observation $\mathring{x}_{i_0}^{(m)}$ has been generated from the $k$th Gaussian. Let $Q_m^t(Z^{(m)}) = P(Z^{(m)}|\mathring{x}_{i_0}^{(m)}, \boldsymbol{\Theta}^t)$, i.e., $Q_m^t(Z^{(m)})$ represents the distribution that, at the $t$th step of the algorithm, $\mathring{x}_{i_0}^{(m)}$ is believed to have been generated by such and such Gaussian. Then, EM is described in Algo. 3.

In the EM algorithm, only the M-step can be computationally intensive. Fortunately, here, we can derive in closed-form the optimal values of Line 4:

**Proposition 1.** *At the E-step, probability* $Q_m^{t+1}(k) = \dfrac{\pi_{jk}^t f(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_k^t)}{\sum_{k'=0}^{g} \pi_{jk'}^t f(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_{k'}^t)}$, *where* $\pi_{jk}^t$ *and* $\boldsymbol{\theta}_k^t$ *are weights, means and variances in* $\boldsymbol{\Theta}^t$. *The optimal parameters of the M-step are respectively:*

$$\pi_{jk}^{t+1} = \frac{\sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} Q_m^{t+1}(k)}{\sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} \sum_{k'=0}^{g} Q_m^{t+1}(k')},$$

$$\mu_k^{t+1} = \frac{\sum_{m=1}^{N} Q_m^{t+1}(k) \mathring{x}_{i_0}^{(m)}}{\sum_{m=1}^{N} Q_m^{t+1}(k)} \qquad \sigma_k^{t+1} = \sqrt{\frac{\sum_{m=1}^{N} Q_m^{t+1}(k)(\mathring{x}_{i_0}^{(m)} - \mu_k^{t+1})^2}{\sum_{m=1}^{N} Q_m^{t+1}(k)}}.$$

Using Algo. 3 with the formulas of Proposition 1, it is thus possible to determine the means and variances of the Gaussians. However, our ultimate goal is not to compute them but to exploit them to discretize variable $\mathring{X}_{i_0}$, i.e., to determine the best cut points $t_1, \ldots, t_g$. Let us see how this task can be performed.

**Input**: a database $\mathring{\mathcal{D}}$, a number $g$ of cut points
**Output**: an optimal set of parameters $\boldsymbol{\Theta}$
1 Select (randomly) an initial value $\boldsymbol{\Theta}^0$
2 **repeat**
  `// E-step (expectation)`
3  $Q_m^{t+1}(Z^{(m)}) \leftarrow P(Z^{(m)}|\mathring{x}_{i_0}^{(m)}, \boldsymbol{\Theta}^t) \quad \forall\, m \in \{1, \ldots, N\}$
  `// M-step (maximization)`
4  $\boldsymbol{\Theta}^{t+1} \leftarrow \underset{\boldsymbol{\Theta}}{\text{Argmax}} \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} \sum_{k=0}^{g} Q_m^{t+1}(k) \log \left[ \dfrac{\pi_{jk} f(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_k)}{Q_m^{t+1}(k)} \right]$
5 **until** *convergence*;

**Algorithm 3:** The EM algorithm.

### 3.4 Determination of the Cut Points

As mentioned at the end of Subsection 3.1, each Gaussian $\mathcal{N}(\mu_k, \sigma_k)$ is associated with an interval $[t_k, t_{k+1})$[2] and the parts of the Gaussian outside the interval can be considered as a loss of information. The optimal set of cut points $\widehat{\mathcal{T}} = \{\hat{t}_1, \ldots, \hat{t}_g\}$ is thus that which minimizes this loss. In other words, it is equal to:

$$\widehat{\mathcal{T}} = \underset{\{t_1,\ldots,t_g\}}{\text{Argmin}} \sum_{k=1}^{g} \int_{t_k}^{+\infty} f(x|\boldsymbol{\theta}_{k-1})dx + \int_{-\infty}^{t_k} f(x|\boldsymbol{\theta}_k)dx,$$

where $\boldsymbol{\theta}_k$ represents pairs $(\mu_k, \sigma_k)$. As each Gaussian $\mathcal{N}(\mu_k, \sigma_k)$ is associated with interval $[t_k, t_{k+1})$, we can assume that $\hat{t}_k \in [\mu_{k-1}, \mu_k)$, for all $k$. Therefore:

$$\widehat{\mathcal{T}} = \left\{ \underset{t_k \in [\mu_{k-1},\mu_k)}{\text{Argmin}} \int_{t_k}^{+\infty} f(x|\boldsymbol{\theta}_{k-1})dx + \int_{-\infty}^{t_k} f(x|\boldsymbol{\theta}_k)dx : k \in \{1, \ldots, g\} \right\}. \quad (6)$$

All the $\hat{t}_k$ can thus be determined independently. In addition, as shown below, their values are the solution of a quadratic equation:

**Proposition 2.** *Let $u(t_k)$ represent the sum of the integrals in Eq. (6). Let $\alpha_k$ be a solution (if any) within interval $(\mu_{k-1}, \mu_k)$ of the quadratic equation in $t_k$:*

$$t_k^2 \left( \frac{1}{\sigma_{k-1}^2} - \frac{1}{\sigma_k^2} \right) + 2t_k \left( \frac{\mu_k}{\sigma_k^2} - \frac{\mu_{k-1}}{\sigma_{k-1}^2} \right) + \left( \frac{\mu_{k-1}^2}{\sigma_{k-1}^2} - \frac{\mu_k^2}{\sigma_k^2} - 2log\frac{\sigma_k}{\sigma_{k-1}} \right) = 0. \quad (7)$$

*Then $\hat{t}_k$ is, among $\{\mu_{k-1}, \mu_k, \alpha_k\}$, the element with the highest value of $u(\cdot)$ (which can be quickly approximated using a table of the Normal distribution).*

**Proof.** Let $g(\cdot)$ and $h(\cdot)$ be two functions such that $\partial g(x)/\partial x = f(x|\boldsymbol{\theta}_{k-1})$ and $\partial h(x)/\partial x = f(x|\boldsymbol{\theta}_k)$. Then:

$$\begin{aligned} \hat{t}_k &= \underset{t_k \in [\mu_{k-1},\mu_k)}{\text{Argmin}} \; u(t_k) = \underset{t_k \in [\mu_{k-1},\mu_k)}{\text{Argmin}} \int_{t_k}^{+\infty} \frac{\partial g(x)}{\partial x} dx + \int_{-\infty}^{t_k} \frac{\partial h(x)}{\partial x} dx \\ &= \underset{t_k \in [\mu_{k-1},\mu_k)}{\text{Argmin}} \; -g(t_k) + h(t_k) + \lim_{t \to +\infty} [g(t) - h(-t)]. \end{aligned}$$

Let us relax the optimization problem and try to find the Argmin over $\mathbb{R}$. Then the min is obtained when $\partial u(t_k)/\partial t_k = 0$ or, equivalently, when $\partial(-g(t_k) + h(t_k))/\partial t_k = -f(t_k|\boldsymbol{\theta}_{k-1}) + f(t_k|\boldsymbol{\theta}_k) = 0$. Since $f(\cdot|\boldsymbol{\theta})$ represents the density of the Normal distribution of parameters $\boldsymbol{\theta}$, this is equivalent to:

$$-\frac{1}{\sqrt{2\pi}\sigma_{k-1}} \exp\left( -\frac{1}{2} \left( \frac{t_k - \mu_{k-1}}{\sigma_{k-1}} \right)^2 \right) + \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left( -\frac{1}{2} \left( \frac{t_k - \mu_k}{\sigma_k} \right)^2 \right) = 0,$$

or, equivalently:

[2] Without loss of generality, we consider here that the the $\mu_k$'s resulting from the EM algorithm are sorted by increasing order.

$$\frac{\sigma_k}{\sigma_{k-1}} = \frac{\exp\left[-\frac{1}{2}\left(\frac{t_k-\mu_k}{\sigma_k}\right)^2\right]}{\exp\left[-\frac{1}{2}\left(\frac{t_k-\mu_{k-1}}{\sigma_{k-1}}\right)^2\right]} = \exp\left[\frac{1}{2}\left(\frac{t_k-\mu_{k-1}}{\sigma_{k-1}}\right)^2 - \frac{1}{2}\left(\frac{t_k-\mu_k}{\sigma_k}\right)^2\right],$$

which, by a log transformation, is equivalent to:

$$2\log\frac{\sigma_k}{\sigma_{k-1}} = \frac{t_k^2}{\sigma_{k-1}^2} - \frac{2\mu_{k-1}t_k}{\sigma_{k-1}^2} + \frac{\mu_{k-1}^2}{\sigma_{k-1}^2} - \frac{t_k^2}{\sigma_k^2} + \frac{2\mu_k t_k}{\sigma_k^2} - \frac{\mu_k^2}{\sigma_k^2}.$$

This corresponds precisely to Eq. (7). So, to summarize, if the optimal solution lies inside interval $(\mu_{k-1}, \mu_k)$, then it satisfies Eq. (7). Otherwise, either $u(t_k)$ is strictly increasing or strictly decreasing within $(\mu_{k-1}, \mu_k)$, which implies that the optimal solution for $\hat{t}_k$ is either $\mu_{k-1}$ or $\mu_k$, which completes the proof. ■

### 3.5 Score and Number of Cut Points

To complete the description of the algorithm, there remains to determine the number of cut points. Of course, the higher the number of cut points, the higher the likelihood but the lower the compactness of the representation. To reach of good trade-off, we simply propose to exploit the penalty functions included into the score used for the evaluation of different BN structures (see Line 5 of Algo. 1). Here, we used the BIC score [18], which can be locally expressed as:

$$BIC(\mathring{X}_{i_0}|\mathbf{MB}(\mathring{X}_{i_0})) = \mathcal{L}(\mathring{\mathcal{D}}|\mathbf{\Theta}) - \frac{|\mathbf{\Theta}|}{2}\log(N) \tag{8}$$

where $\mathcal{L}(\mathring{\mathcal{D}}|\mathbf{\Theta})$ is the log-likelihood with the parameters estimated by EM, given the current structure $\mathcal{G}$. $|\mathbf{\Theta}|$ represents the number of parameters, i.e., $|\mathbf{\Theta}| = q_{i_0} \times g + 2 \times (g+1)$: the 1st and 2nd terms correspond to the number of parameters $\pi_{jk}$ and of $(\mu_k, \sigma_k)$ needed to encode the conditional distributions (recall that there are $g+1$ Gaussians and $q_{i_0}$ represents the domain size of $\mathbf{MB}(\mathring{X}_{i_0})$). Now, the best number of cut points is simply that which optimizes Eq. (8).

## 4 Experimentations

In this section, we highlight the effectiveness of our method, hereafter denoted MGCD (for Mixture of Gaussians Clustering-based Discretization), by comparing it with the algorithms provided in [17] and [6], hereafter called Ruichu and Friedman respectively. Step 4 of Algo. 1 was performed using a simple *Tabu* search method. For the comparisons, three criteria have been taken into account: i) the quality of the structure learnt by the algorithm (which strongly depends on that of the discretization); ii) the computation time and iii) the quality of the learnt CPT parameters, which has been evaluated by their prediction power on the values taken by some variables given observations.

For the first two criteria, we randomly generated discrete BNs following the guidelines given in [7]. Those contained from 10 to 30 nodes and from 12 to 56

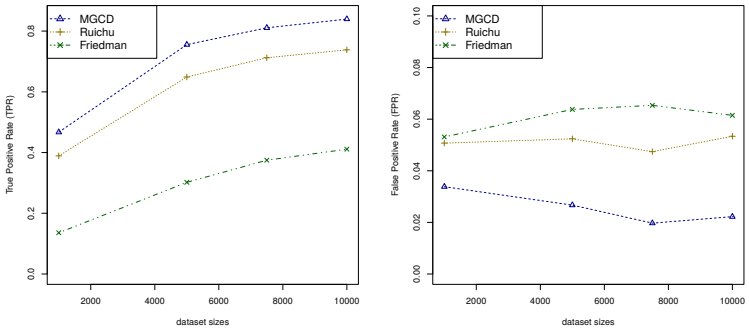


**Fig. 3.** Averages of the TPR (left) and FPR (right) metrics for BNs with 10 to 30 nodes in function of the sample sizes.

arcs. Each node had at most 6 parents and its domain size was randomly chosen between 2 and 5. The CPTs of these BNs represented the $\pi_{jk}$ of the preceding section. From these BNs, we generated continuous datasets containing from 1000 to 10000 records as follows: for each random variable $X_i$, we mapped its finite set of values into a set of consecutive intervals $\{[t_{k-1}, t_k)\}_{k=1}^{|X_i|}$ of arbitrary lengths. Then, we assigned a truncated Gaussian to each interval, the parameters $(\mu_k, \sigma_k)$ of which were randomly chosen. Finally, to generate a continuous record, we first generated a discrete record from the discrete BN using a logic sampling algorithm. Then, this record was mapped into a continuous one by sampling from the truncated exponentials. Overall, 350 continuous datasets were generated.

To compare them, the BN structures produced by Ruichu, Friedman and MGCD were converted into their Markov equivalence class, i.e., into a partially directed DAG (CPDAG). Such a transformation increases the quality of comparisons since two BNs encode the same distribution iff they belong to the same equivalence class. The CPDAGs were then compared w.r.t. their true and false positive rate metrics (TPR and FPR). TPR (resp. FPR) represents the percentage of arcs/edges belonging to the learnt CPDAG that also exist (resp. do not exist) in the original CPDAG. Both metrics describe how well the dependences between variables are preserved by learning/discretization. Fig. 3 shows the average TPR and FPR over the 350 generated databases. As can be seen, MGCD outperforms the others for all dataset sizes: MGCD's TPR is about 10% higher than Ruichu and 40% higher than Friedman, and MGCD's FPR is between 20% and 40% lower than the other methods. MGCD's performance w.r.t. Ruichu's can be explained by that fact that, unlike Ruichu's, it fully takes into account the conditional dependences between all the random variables. Its performance w.r.t. Friedman's can be explained by our choice of exploiting clustering rather than an entropy-based approach. Table 1 provides computation time ratios (other method's runtime / MGCD's runtime). As can be seen, our method slightly outperforms Ruichu's (but is 10% better in terms of TPR and more than 20% better in terms of FPR) and it significantly outperforms Friedman's (about 3 times faster) while at the same time being 40% higher in terms of TPR.

| Approaches / Dataset sizes | 1000 | 5000 | 7500 | 10000 |
|---|---|---|---|---|
| Friedman | 2.762444 | 3.350782 | 3.404958 | 3.361540 |
| Ruichu | 0.8872389 | 1.1402535 | 1.1334637 | 1.1032982 |

**Table 1.** Runtime ratio comparisons between discretization approaches.

Finally, we compared the discretizations w.r.t. the quality of the produced CPTs. To do so, we generated from two classical BNs, Child and Sachs, 100 continuous databases using the same process as above except that: i) the distributions inside intervals were uniform instead of Gaussians (to penalize our approach since data do not fit its hypotheses), and ii) some small sets of variables were kept discrete and served as multilabel targets. Databases were split into a learning (2/3) and a test (1/3) part. For each record in the latter, we computed the distribution (learnt by each of the 3 algorithms on the learning database) of each target given some observations on their Markov blanket and we estimated the value of the target by sampling it from the learnt distribution. The percentages of correct predictions are shown in Table 2. As we can see, our algorithm outperforms the other algorithms, especially Ruichu's, which fails to have correct predictions due to its univariate discretization not taking into account the conditional dependencies among random variables. Friedman's results are closer to ours but recall that it is about 3 times slower than ours.

## 5 Conclusion

We have proposed a new multivariate discretization algorithm designed for BN structure learning, taking into account the dependences among variables acquired during learning. Our experiments highlighted its efficiency and effectiveness compared to state-of-the-art algorithms, but more experiments are of course needed to better assess the strengths and the shortcoming of our proposed approach. For future work, we plan to improve our algorithm, notably by directly working with truncated Gaussians instead of the current approximation by mixture of Gaussians. But such an improvement is not trivial due to the fact that, in this case, no closed-form solution exists for determining the cut points.

| datasets | sizes | 30 % Markov blanket | | | 60% Markov blanket | | | 100 % Markov blanket | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | MGCD | Ruichu | Friedman | MGCD | Ruichu | Friedman | MGCD | Ruichu | Friedman |
| Child | 1000 | **60.90** | 59.30 | 58.99 | **62.76** | 60.90 | 59.96 | **67.53** | 65.34 | 63.33 |
| | 2000 | **61.59** | 56.62 | 58.05 | **62.41** | 59.24 | 60.55 | **67.29** | 64.71 | 63.03 |
| | 5000 | **64.88** | 62.29 | 60.05 | **66.07** | 62.95 | 61.94 | **69.42** | 65.39 | 63.82 |
| | 10000 | **65.81** | 62.48 | 61.75 | **67.44** | 63.85 | 63.51 | **70.49** | 66.92 | 65.79 |
| Sachs | 1000 | 56.63 | 54.78 | **57.59** | 57.22 | 55.04 | **58.74** | **65.67** | 61.06 | 64.65 |
| | 2000 | **56.96** | 56.16 | 54.02 | **59.72** | 57.58 | 56.64 | **65.80** | 62.24 | 60.22 |
| | 5000 | **57.69** | 55.00 | 55.15 | **59.80** | 57.96 | 56.38 | **65.51** | 64.15 | 64.47 |
| | 10000 | **60.35** | 57.50 | 57.33 | **61.67** | 58.26 | 59.22 | **70.04** | 65.74 | 64.61 |

**Table 2.** Prediction accuracy rates for discrete target variables in the Child and Sachs standard BNs (`http://www.bnlearn.com/bnrepository/`) w.r.t. the percentage of observed variables in the Markov blanket.

## References


1. Boullé, M.: MODL: a Bayes optimal discretization method for continuous attributes. Machine learning 65(1), 131–165 (2006)
2. Boullé, M.: Khiops: A statistical discretization method of continuous attributes. Machine Learning 55(1), 53–69 (2004)
3. Dempster, A.P., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, pp. 1–38 (1977)
4. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: proc. of ICML'95. pp. 194–202 (1995)
5. Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proc. of IJCAI'93. pp. 1022–1029 (1993)
6. Friedman, N., Goldszmidt, M.: Discretizing continuous attributes while learning Bayesian networks. In: proc. of ICML'96. pp. 157–165 (1996)
7. Ide, J.S., Cozman, F.G., Ramos, F.T.: Generating random Bayesian networks with constraints on induced width. In: Proc. of ECAI'04. pp. 323–327 (2004)
8. Jiang, S., Li, X., Zheng, Q., Wang, L.: Approximate equal frequency discretization method. In: proc. of GCIS'09. pp. 514–518 (2009)
9. Kerber, R.: ChiMerge: Discretization of numeric attributes. In: Proc. of AAAI'92. pp. 123–128 (1992)
10. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
11. Kwedlo, W., Krętowski, M.: An evolutionary algorithm using multivariate discretization for decision rule induction. In: proc. of PKDD'99. pp. 392–397 (1999)
12. Lauritzen, S., Wermuth, N.: Graphical models for associations between variables, some of which are qualitative and some quantitative. Annals of Statistics 17(1), 31–57 (1989)
13. Monti, S., Cooper, G.: A multivariate discretization method for learning Bayesian networks from mixed data. In: Proc. of UAI'98. pp. 404–413 (1998)
14. Monti, S., Cooper, G.: A latent variable model for multivariate discretization. In: proc. of AIS'99. pp. 249–254 (1999)
15. Moral, S., Rumi, R., Salmeron, A.: Mixtures of truncated exponentials in hybrid Bayesian networks. In: Proc. of ECSQARU'01. Lecture Notes in Artificial Intelligence, vol. 2143, pp. 156–167 (2001)
16. Ratanamahatana, C.: CloNI: Clustering of sqrt(n)-interval discretization. In: proc. of Int. Conf. on Data Mining & Comm Tech. (2003)
17. Ruichu, C., Zhifeng, H., Wen, W., Lijuan, W.: Regularized Gaussian mixture model based discretization for gene expression data association mining. Applied intelligence 39(3), 607–613 (2013)
18. Schwarz, G.: Estimating the dimension of a model. Annals of Statistics 6(2), 461–464 (1978)
19. Shachter, R.: Evaluating influence diagrams. Operations Research 34(6), 871–882 (1986)
20. Shenoy, P., West, J.: Inference in hybrid Bayesian networks using mixtures of polynomials. International Journal of Approximate Reasoning 52(5), 641–657 (2011)
21. Song, D., Ek, C., Huebner, K., Kragic, D.: Multivariate discretization for Bayesian network structure learning in robot grasping. In: proc. of ICRA'11. pp. 1944–1950 (2011)
22. Zighed, D., Rabaséda, S., Rakotomalala, R.: FUSINTER: a method for discretization of continuous attributes. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6(03), 307–326 (1998)