Noname manuscript No.
(will be inserted by the editor)

Learning Discrete-valued Bayesian Networks from Mixed Data

First Author · Second Author · Third Author

Received: date / Accepted: date

Abstract Insert your abstract here. Include keywords, PACS and mathematical subject classification numbers as needed.

Keywords Discretization · Bayesian Network · Continuous Variable

1 Introduction

Bayesian networks (Pearl 1988; Koller and Friedman 2009) are an increasingly popular method for modeling uncertainty and causality in science and engineering. They provide an efficient factorization of the joint probability distribution over a set of random variables. Bayesian networks first emerged from artificial intelligence research and have been applied to a wide variety of problems, ranging from decision-making systems (Kochenderfer 2015) to medical diagnoses. In most cases, we assume that all random variables in Bayesian networks are discrete, since many algorithms on Bayesian networks are unable to deal with continuous variables efficiently. However, the assumption that all variables are discrete is often too restrictive. For example, in the decision-making system of autonomous cars, it is imperative to deal with continuous variables such as position and velocity.

There are two methods around this assumption. The first one is to model conditional probability density of each continuous variable by specific families of parametric distributions, then redesign algorithms on Bayesian networks based on these parameters. One successful example is belief propagation in

F. Author
first address
Tel.: +123-45-678910
Fax: +123-45-678910
E-mail: fauthor@example.com

S. Author
second address

Gaussian graphical models (Weiss and Freeman 2001). Nevertheless, for other shapes of particles (Ihler and McAllester 2009) or non-linear functions, these redesigned algorithms are computationally expensive and do not perform well.

The second method around continuous variables is to discretize them. Discretization that learns from data has been developed well and discussed in the fields of machine learning and statistics for many years (Dougherty et al. 1995; Kerber 1992; Holte 1993; Fayyad and Irani 1993) . Most of these discretization methods are designed for classification problems. They search the best discretization policy of a continuous attribute by considering its interation with the class variable of interest. However, these discretization methods do not apply to continuous variables in Bayesian networks. In Bayesian networks, interations and dependencies between variables are determined by graph structure. Therefore, if a method discretizes continuous variables according to graph structure, instead of assigning one variable as a class variable, then it would be more appropriate. There is some reseach on discretizing continuous variables in naive Bayesian networks and tree augmented networks (Friedman et al. 1997), but only a few discretization methods on general Bayesian netowrk have been proposed (Friedman and Goldszmidt 1996; Kozlov and Koller 1997; Monti and Cooper 1998; Stech and Jaakkola 2007).

The discretization technique proposed by Friedman and Goldszmidt (1996) is the most well-known one among these few methods. It is based on minimum description length principle (MDL): optimal discretization policy minimizes the description length of Bayesian network and data information. If there is only one continuous variable in a Bayesian network and other variables are discrete, this MDL discretization method takes running time $O(N^3 + n_c {v_{max}}^{(n_c^p)_{max}} \cdot N^2 + {v_{max}}^{n_p} \cdot N^2)$, where N is the number of data instances for learning the discretization, n_p and n_c are the numbers of parent and children variables for the continuous variable, v_{max} is the largest cardinality over all variables in the Markov blanket, and $(n_c^p)_{max}$ is the largest number of parent variables of the continuous variable's children. If there are multiple continuous variables in the network, the method iterates over all continuous variables until convergence. In our test on real-world data, the iterations converge in a few cycles. However, the number of intervals after discretization is usually too small.

In this paper, we propose a new discretization method for continuous variables in Bayesian networks by learning from mixed data. This method is a generalized version of Boullé (2006) and Lustgarten et al. (2011), which are both discretization methods of a continuous attribute with one class variable. We begin our method with the assumption that only one variable in a given Bayesian network is continuous and other variables are discrete. Furthermore we assume that the network structure is known in advance. Under this situation, we look for the most probable discretization policy M given the data D on other discrete variables. That is to say, the desired policy is $\arg_M P(M \mid D)$. With Bayes' rule, $P(M \mid D)$ can be rewritten as $P(M) \cdot P(D \mid M)/P(D)$, which is proportional to $P(M) \cdot P(D \mid M)$. Usually, $P(D \mid M)$ increases as the number of discretized intervals increases, since more intervals can pro-

vide more accuracy. $P(M)$, on the other hand, is designed to decrease as the number of intervals rises. As a result, we can automatically determine the number of intervals after discretization by maximizing $P(M) \cdot P(D \mid M)$. In addition, the $P(D \mid M)$ can be factorized according to Bayesian network structure. With the proposed priors, we are able to restrict the number of discretized intervals so that it will not exceed the largest cardinality of variables in the Markov blanket by too much. This is important, since for most algorithms on Bayesian networks, their running times exponentially depend on the cardinality of variables. Another advantage of our method is the running time for learning. If there is only one continuous variable, the running time is $O(n_c v_{max} \cdot N^2 + {v_{max}}^{n_p} \cdot N^2)$, which is significantly shorter than the running time of MDL principle discretization (Friedman and Goldszmidt 1996). With the approximation proposed in this paper, we can further reduce running time to $O(v_{max}(n_p + n_c)N^2)$.

For a Bayesian network with multiple continuous variables, we apply the discretization method discussed above iteratively. In the beginning, we prediscretize each continuous variable by equal-width method. The number of intervals for equal-width discretization is equal to largest cardinality of discrete variables. Then we iterate the one-variable discretization on each continuous variable in the following order: from the variable with highest topological order (leaves) to the variable with lowest topological order (root). Each time we finish a discretization procedure on one variable, we store the discretization result for later iterations. Experiments on real-world data show that with the same iteration order, our discretization method provides better discretization results than MDL principle method in terms of likelihood. The MDL principle method is easily stuck at local minima and also discretizes variables into too few intervals.

Finally, we can combine our new discretization method with the K2 structure learning alorithm (Cooper and Herskovits 1992). We first prediscretize all continuous variables before the K2 learning, since the K2 algorithm requires all variables to be discrete. Each time an edge is added in the K2 algorithm procedure, we rediscretize the continuous variables based on the updated Bayesian network and store the result for the next K2 algorithm iteration. By this principle, we are able to learn a discrete-valued Bayesian network from mixed data.

The rest of the paper is organized as follows: related works, including MDL principle discretization (Friedman and Goldszmidt 1996) and MODL discretization by Boullé (2006), are summarized in Section 2. We prepare the preliminaries and notations in Section 3. In Section 4, we introduce the new discretization method for the case that only one variable is continuous, including the derivation of objective function and the algorithms. Section 5 is a discussion about multiple continuous variables in a Bayesian network and how to discretize them. The combination of our proposed discretization and the K2 structure learning is also introduced. Finally, in Section 6, we apply the proposed discretization method to real-world data and show the result.

2 Related Work

In this section we review two related works: MDL principle discretization (Friedman and Goldszmidt 1996) and MODL discretization (Boullé 2006). The former is the most famous discretization method for continuous variables in Bayesian networks, and we will compare it with our proposed method in Section 6. The latter is a discretization method for one continuous attribute according to a target class. Our proposed method is a generalization of this method. Note that MODL discretization is a Bayesian approach. The asymptotical equivalence between MDL approach and Bayesian approach has been examined in (Vitényi and Li 2000).

2.1 MDL Principle Discretization

The MDL principle was first proposed by Rissanen (1978). It provides a way to compress data, that is to say, describe data with fewer number of symbols than the number of symbols needed to describe the data literally (Grünwald 2007). MDL chooses a model that trades off goodness-of-fit for the complexity of model. Therefore it has an advantage that automatically avoids overfitting. In the case of Bayesian networks, Friedman and Goldszmidt (1996) applied the MDL concept to determine the number of intervals of each continuous variable after discretization and also the positions of discretization boundaries. Here is the mechanism: MDL principle discretization selects a discretization policy that minimizes sum of description lengths of discretized Bayesian network and the necessary information that recovers original data from discretized data. If there is only one continuous variable and other variables are discrete, the objective function is

$$\begin{aligned}&\frac{1}{2}\log(N)\left\{\|\Pi_{X_i}\|(\|X_i^*\|-1)+\sum_{j,X_i\in\Pi_{X_i}}\|\Pi_{X_j^*}\|(\|X_j\|-1)\right\}+\log(\|X_i^*\|)\\&+(N_i-1)H\left(\frac{||X_i^*||-1}{N_i-1}\right)-N\cdot\left[I(X_i^*,\Pi_{X_i})+\sum_{j,X_i\in\Pi_{X_i}}I(X_j,\Pi_{X_j^*})\right],\end{aligned}\tag{1}$$

where function $I(\boldsymbol{A},\boldsymbol{B})=\sum_{\boldsymbol{a},\boldsymbol{b}}\hat{P}_D(\boldsymbol{a},\boldsymbol{b})\log\frac{\hat{P}_D(\boldsymbol{a},\boldsymbol{b})}{\hat{P}_D(\boldsymbol{a}),\hat{P}_D(\boldsymbol{b})}$ is the mutual information between two sets of variables $\boldsymbol{A}$ and $\boldsymbol{B}$, $||\boldsymbol{X}||$ means the cardinality of a variable set $\boldsymbol{X}$, X_i^* is the discretized version of continuous variable X_i and $H(p)=-p\log(p)-(1-p)\log(1-p)$. The optimal discretization policy can be found by dynamical programming. If there are multiple continuous variables in a Bayesian network, we apply the algorithm on one variable at a time and iterate over all continuous variables. While iterating, only one variable is treated as continuous and other continuous variables are discretized based on initial prediscretization or the discretization result of previous iteration.

For the above discretization processes, the network structure is known in advance. If the network structure is not given, we can alternate between structure learning and discretization learning. We start with some discretization, and learn a network structure given this discretization. Then, we rediscretize based on the learned network. This cycle continues until no improvement in objective function is made.

2.2 MODL Discretization

MODL discretization (Boullé 2006) is a Bayesian-approach discretization. It discretizes a continuous feature according to a class variable. The best discretization model is found by maximizing the probability $P(Model \mid Data)$ of the model given the data. Using Bayes' rule and since $P(Data)$ is a constant while varying the discretization model, this is equivalent to maximize:

$$P(Model) \cdot P(Data \mid Model). \tag{2}$$

Once the prior distribution of the discretization model is fixed, each term in Eq (3) can be evaluated. Usually, $P(Data \mid Model)$ increases as the number of intervals of discretization model increases, since more intervals means that the model contains more information and thus it is easier to reproduce the original data. On the other hand, $P(Model)$ decreases as the number of intervals increases. Thus, maximizing $P(Model) \cdot P(Data \mid Model)$ provides a trade-off to determine number of intervals after discretization.

The MODL method adapts dynamical programming to find the optimal discretization model, and it takes running time $O(N^3 + ||\text{Class}|| \cdot N^2)$, where N is number of data instances. Lustgarten et al. (2011) provide a different prior to evaluate $P(\text{Model})$ term. This prior could include the assumption of uniform prior probabilities over discretization in MODL as a special case. Additionally, it reduces runing time to $O(||\text{Class}|| \cdot N^2)$.

3 Preliminaries

In this section we provide a brief review of Bayesian networks, including the factorization of joint probability distribution, sampling from a given network, and structure learning. These concepts will be used in later sections. In addition, the formal definition of a discretization policy on a continuous variable is also introduced.

3.1 Bayesian Network and Structure Learning

A Bayesian network B is defined by a pair (G, Θ), where $G = (X, E)$ is a directed acyclic graph whose nodes correspond to a set of random variable $X = \{X_1, X_2, \cdots, X_n\}$, and whose edges E represents probabilistic dependencies

among nodes. The graph structure G encodes the Markov property: each node X_i is independent of its non-descendants given its parents in G. The second component of B, namely Θ, contains a set of parameters that qualify the network. Elements of Θ take the form $\theta_{x_i|\Pi_{x_i}} = P(x_i \mid \Pi_{x_i})$ for each possible value x_i of X_i, and Π_{x_i} of Π_{X_i} (the set of parents of X_i in G). Applying Markov property, we can represent the multivariate joint distribution over X as

$$P_B(X_1, \cdots, X_n) = \prod_{i=1}^{n} P_B(X_i \mid \Pi_{X_i}) = \prod_{i=1}^{n} \theta_{x_i|\Pi_{x_i}} \tag{3}$$

For a given Bayesian network, we can generate its data instances by forward sampling, i.e., sampling variables one by one in a topological order (see Cormen et al. 2009, chap. 22). Given an instance of X_i's parent set, Π_{X_i}, values of X_i can be sampled according to the conditional probability table $P(X_i \mid \Pi_{X_i})$. Furthermore, this parent-child sampling order can be reversed if we know the marginal probability of child variable $P(X_i)$. By Bayes' rule,

$$P(X_i \mid \Pi_{X_i}) \cdot \prod_{j=1}^{Pa(X_i)} P(\{\Pi_{X_i}\}_j) = P(X_i) \cdot P(\Pi_{X_i} \mid X_i), \tag{4}$$

where $Pa(X_i)$ is the number of parents of X_i, and $\{\Pi_{X_i}\}_j$ is the jth parent of X_i. We can first sample X_i by $P(X_i)$, then sample all the parent variables simultaneously by $P(\Pi_{X_i} \mid X_i)$. That is to say, X_i becomes the starting point of sampling.

If we don't know the structure of the Bayesian network in advance, then we need to learn it from data. Roughly speaking, there are three approaches to learn a Bayesian network structure from data (see Koller and Friedman 2009, chap. 18): constraint-based structure learning, score-based structure learning, and Bayesian model averaging. Here we review the K2 structure learning algorithm (Cooper and Herskovits 1992), which is one of the most successful score-based structure learning methods. Similar to most structure learning algorithms, the K2 algorithm requires all variables to be discrete. The score of a learned network is defined as $\prod_i f(X_i, \Pi_{X_i})$, where

$$f(X_i, \Pi_{X_i}) = \prod_{j=1}^{||\Pi_{X_i}||} \frac{(||X_i|| - 1)!}{(N_{ij} + ||X_i|| - 1)!} \prod_{k=1}^{||X_i||} \alpha_{ijk}!. \tag{5}$$

$||X_i||$ is the cardinality of variable X_i. $||\Pi_{X_i}||$ is number of all possible instantiations of the parent variables of X_i, i.e., $|\Pi_{X_i}| = \prod_{Y \in \Pi_{X_i}} |Y|$. α_{ijk} is the number of instances in a data set that variable X_i is instantiated with its kth value, and the parent of X_i are instantiated with the jth value of Π_{X_i}. $N_{ij} = \sum_{k=1}^{|X_i|} \alpha_{ijk}$.

The K2 algorithm searches the network structure with the highest score, which can be interpreted as the most probable network with the given data. In addition, the K2 requires a topological order of variables to be known before the scoring can start. This constraint can prevent cycles from being introduced.

The searching for high-score network is an iterative process. There is no way to find the optimal network directly, since there are $2^{O(n^2)}$ possible structures, where n is number of variables. In order to compensate for this, we run the K2 algorithm many times, and each time we start with a different order of variables. The network with the highest score in iterations is the desired one.

3.2 Discretization Policy

A Discretization Policy $M_C =< t_1, t_2, \ldots, t_{k-1} >$ on a continuous variable C is a mapping from $\boldsymbol{R}$ to $\{1, 2, 3, \ldots, k\}$ such that

$$M_C(x) = \begin{cases} 1, & \text{if } x < t_1. \\ i, & \text{if } t_{i-1} \leq x < t_i. \\ k, & \text{if } t_k \leq x \end{cases} \tag{6}$$

That is to say, the policy discretizes the continuous variable C into k intervals. Furthermore, we assume that the discretization edge t_i can only take values on middle points of two consective values of data of C. By this assumption, we can obtain an integer representation of M_C as follows: we sort the data of continuous variable C in an increasing order, $\{c_1, c_2, \ldots, c_N\}$, and if $t_i = (c_{s_i} + c_{s_i+1})/2$ for $i = 1, 2, \ldots, k$, then $M_C =< t_1, t_2, \ldots, t_{k-1} >\equiv [n_1, n_2, ..n_k]$, where $n_1 = s_1$ and $n_i = s_i - s_{i-1}$ for $i = 2, \ldots, k$.

4 Discretize One Continuous Variable

In this section, we consider the case that only one variable in a Bayesian network is continuous and the others are discrete. Before we formulate the objective function, we introduce some notations that will make the following calculation easier.

4.1 Notations

Assume the continuous variable is X. It has n_p parent variables, $\Pi_X = \{P_1, P_2, P_3, \ldots, P_{n_p}\}$. It also has n_c child variables $\{C_1, C_2, \ldots, C_{n_c}\}$. Each child variable C_i of X has other parent variables, which forms a set $\boldsymbol{S_i}$. Before we do the discretization, we sort the data instances by its attribute of X.

Let $D \cup D_X$ be a dataset of N instances which we plan to learn discretization from. D_X only contains the data of X. D contains the data of all variables except X. Assume $D \cup D_X$ has been sorted in the ascending order of D_X. In the following content, $D_{\boldsymbol{Y}}$ means the data instances of a set of variable $\boldsymbol{Y}$. Let $D_X = \{x_1, x_2, x_3, \ldots, x_N\}$. In order to make the discretization mechanism more understandable, we temporarily assume there is no repeated values in D_X, i.e., $x_1 < x_2 < \ldots < x_N$. We will remove this assumption later in the

algorithm section. Therefore, a discretization policy M on X can be written as $M = [n_1, n_2, \ldots, n_k]$, where k, n_1, n_2, $\ldots$, n_k are positive integers satisfying $N = \sum_{i=1}^{k} n_i$.

4.2 Priors and Objective Function

We have the following four priors for the discretization model M that enable us to evaluate $P(M)$ and $P(D \mid M)$:

1. For a discretization edge locating at $(x_i + x_{i+1})/2$ has probability
$$1 - \exp(-L \cdot \frac{x_{i+1} - x_i}{x_N - x_1}), \tag{7}$$
where L is the largest cardinality number of discrete variables in X's Markov blanket.
2. For a given interval of M, every distribution of Π_X is equiprobable.
3. For each pair of $(C_i, \boldsymbol{S_i})$ and a given interval of M, every distribution of C_i with a value of $\boldsymbol{S_i}$ is equiprobable.
4. The distributions of each C_i and Π_X in each interval of M with each value of $\boldsymbol{S_i}$ are independent from each other.

Owing to these priors, we are able to evaluate $P(M)$ and $P(D \mid M)$. For the former, we have

$$P(M) = \prod_{i=1}^{k} \{[1 - \exp(-L \cdot \frac{x_{s_i+1} - x_{s_i}}{x_N - x_1})] \cdot \exp(L \cdot \frac{x_{s_i} - x_{s_{i-1}+1}}{x_N - x_1})\}, \tag{8}$$

where $s_i = \sum_{j=1}^{k} n_j$ and $s_k = N$, $s_0 = 0$. Before evaluating $P(D \mid M)$, notice that

$$P(D \mid M) \propto P(D_{\Pi_x} \mid M) \cdot \prod_{i=1}^{n_c} P(D_{C_i} \mid M, D_{\boldsymbol{S_i}}). \tag{9}$$

The graph structure allows us to only consider the interations between X and other variables in its Markov blanket and thus leads to the factorization. For example, in Figure 1., the corresponding $P(D \mid M)$ is factorized as

$$P(D_{P_1,P_2,P_3} \mid M) \cdot P(D_{C_1} \mid M, D_{\boldsymbol{S_1}}) \cdot P(D_{C_2} \mid M, D_{S_2}). \tag{10}$$

The concept behind the factorization is similar to sampling in a Bayesian network. Once we have the distribution of X and sample X for it, then we can further sample Π_x. Since the samples of $P_1, P_2, \ldots, P_{n_p}$ given a value of X are not independent, we can not further factorize $P(D_{\Pi_x} \mid M)$ into $\prod_i P(D_{P_i} \mid M)$. Similarly, once we have samples of X and $\boldsymbol{S_i}$, we can sample C_i. Since the sampling processes of child variables are independent from each other, we have the product $\prod_{i=1}^{n_c} P(D_{C_i} \mid M, D_{\boldsymbol{S_i}})$.

For the following part, we evaluate $P(D_{\Pi_x} \mid M)$ and $P(D_{C_i} \mid M, D_{\boldsymbol{S_i}})$ based on discretization policy $M = [n_1, n_2, \ldots, n_k]$ and dataset D.

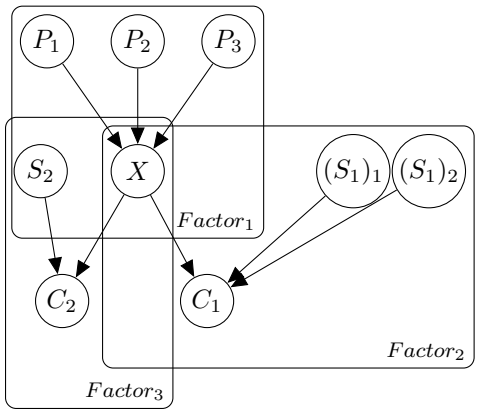

Fig. 1 Factorization of $P(D \mid M)$

4.2.1 Evaluate $P(D_{\Pi_x} \mid M)$

Assume that $J_P = \prod_{i=1}^{n_p} ||P_i||$. Then we have:

$$P(D_{\Pi_X} \mid M) = \prod_{i=1}^{k} \frac{1}{\binom{n_i+J_P-1}{J_P-1}} \frac{1}{\frac{n_i!}{n_{i,1}^{(p)}! n_{i,2}^{(p)}! \cdots n_{i,J_P}^{(p)}!}}, \tag{11}$$

where $n_{i,j}^{(p)}$ is the number of instances in ith discretized interval with jth value of Π_X. Note that $n_i = \sum_{j=1}^{||\Pi_x||} n_{i,j}^{(p)}$. The two factors on RHS comes from the second prior: all distributions of values of Π_X in a given interval are equiprobable. According to the forth prior, the distribution in each interval is independent, so we multiply all the two factors together.

4.2.2 Evaluate $P(D_{C_i} \mid M, D_{\boldsymbol{S_i}})$

Assume for each pair of $(C_j, \boldsymbol{S_j})$, we have $||C_j|| = J_j$ and $||\boldsymbol{S_j}|| = L_j = \prod_{v \in \boldsymbol{S_j}} ||v||$. Therefore,

$$P(D_{C_j} \mid M, D_{S_j}) = \prod_{i=1}^{k} \prod_{l=1}^{L_j} \frac{1}{\binom{n_{i,l}+J_j-1}{J_j-1}} \frac{1}{\frac{n_{i,l}!}{n_{i,1,l}^{(j)}! n_{i,2,l}^{(j)}! \cdots n_{i,J_j,l}^{(j)}!}}, \tag{12}$$

where $n_{i,m,l}^{(j)}$ is the number of instances in ith interval with mth value of C_j and lth value of $\boldsymbol{S_j}$, $n_{i,l} = \sum_{m=1}^{J_j} n_{i,m,l}^{(j)}$, and $n_i = \sum_{l=1}^{L_j} n_{i,l}$. The two factors on RHS comes from the third prior: all distribution of values of C_j in a given interval and with a given value of S_j are equiprobable. According to the forth prior, these distributions are independent from each other, therefore we

multiply all factors. If $S_j = \emptyset$, then Equation 13. is equivalent to

$$P(D_{C_j} \mid M) = \prod_{i=1}^{k} \frac{1}{\binom{n_i+J_j-1}{J_j-1}} \frac{1}{\frac{n_i!}{n_{i,1,\emptyset}^{(j)}! n_{i,2,\emptyset}^{(j)}! \cdots n_{i,J_j,\emptyset}^{(j)}!}}, \tag{13}$$

where $n_{i,m,\emptyset}^{(j)}$ is the number of instances in ith interval with mth value of C_j, and $\sum_{m=1}^{J_j} n_{i,m,\emptyset}^{(j)} = n_i$.

With Equation 8,9,11,12, and 13, now we are able to write down the objective function. Instead of maximizing $P(M) \cdot P(D|M)$, we minimize its log-inverse for convenience. The objective function is

$$\begin{aligned}
&\sum_{i=1}^{k-1} -\log(1 - \exp(-L \cdot \frac{x_{s_i+1} - x_{s_i}}{x_N - x_1})) + \sum_{i=1}^{k} L \cdot \frac{x_{s_i} - x_{s_{i-1}+1}}{x_N - x_1} + \\
&\sum_{j=1}^{n_c} \sum_{i=1}^{k} \sum_{l=1}^{L_j} \left[\log \binom{n_{i,l} + J_j - 1}{J_j - 1} + \log \left(\frac{n_{i,l}!}{n_{i,1,l}^{(j)}! n_{i,2,l}^{(j)}! \cdots n_{i,J_j,l}^{(j)}!} \right) \right] + \\
&\sum_{i=1}^{k} \left[\log \binom{n_i + J_P - 1}{J_P - 1} + \log \left(\frac{n_i!}{n_{i,1}^{(p)}! n_{i,2}^{(p)}! \cdots n_{i,J_p}^{(p)}!} \right) \right].
\end{aligned} \tag{14}$$

All parameters and variables in the objective function are explained in previous subsections.

4.3 Algorithm

Once the objective function is established, the next problem is how to find a discretization model that minimizes the objective function. Note that the objective function is cumulative on intervals. Therefore, if a partition of continuous variable X into k intervals with lengths $n_1, n_2, \ldots, n_k$ is an optimal discretization policy, then $\{n_2, n_3, \ldots, n_k\}$ is optimal for the subproblem, i.e., the last $N - n_1$ instances of the dataset which has been sorted in ascending order of D_X. With this property, we can adapt dynamical programming to solve this optimization problem.

If there is no repeated value in D_X, all middle points of two consecutive values can a discretization edge. If there exists repeated values in D_X, only the middle points of two consecutive and different values can be a discretization edge. Assume there are M unique values in D_X, then $D'_X = \{x'_1, x'_2, \ldots, x'_M\}$ is the set of unique values of D_X in ascending order. Since D_X is also sorted, let $b = \{b_0, b_1, b_2, \ldots, b_M\}$ be an increasing sequence of integers such that $b_0 = 0$ and $x_{b_{i-1}+1} = x_{b_{i-1}+2} = \ldots = x_{b_i} = x'_i$. By this definition, the allowable discretization positions are $d_i = (x_{b_i} + x_{b_i+1})/2$ for all $i = 1, 2, \ldots, M-1$.

Before doing the dynamical programming, in order to save runtime, we first calculate the following function $h(u, v)$ for a interval I_q starting from x_u to x_v with all u,v satisfying $u \leq v$:

$$h(u,v) = \log \binom{n_q + J_P - 1}{J_P - 1} + \log \left(\frac{n_q!}{n_{q,1}^{(p)}! n_{q,2}^{(p)}! \cdots n_{q,J_p}^{(p)}!} \right) + \sum_{j=1}^{n_c} \sum_{l=1}^{L_j} \left[\log \binom{n_{q,l} + J_j - 1}{J_j - 1} + \log \left(\frac{n_{q,l}!}{n_{q,1,l}^{(j)}! n_{q,2,l}^{(j)}! \cdots n_{q,J_j,l}^{(j)}!} \right) \right] \tag{15}$$

The evaluation of function $h(u, v)$ for all $u \leq v$ is summerized in Algorithm 4 in Appendix. The calculation can be done in $O(n_c v'_{max} \cdot N^2 + {v'_{max}}^{n_p} \cdot N^2)$, where n_c and n_p are the numbers of child and parent variables, respectively, and v'_{max} is the largest cardinality of variables that directly connects to X. Notice that due to repeated values in D_X, for some pairs of u and v, the value of $h(u, v)$ might depend on the sorting method of D_X and D. However, this does not influence the optimization result, since these pairs of u and v will not form valid intervals.

Now we are able to solve the optimization problem with the objective function in Equation 14. The dynamical programming procedure is shown in Algorithm 1. We have three inputs: D_X, the data of continuous variable X in the ascending order, D, the data of other variables and sorted according to D_X, and G, the network structure. The runtime of Algorithm 1 is also $O(n_c v'_{max} \cdot N^2 + {v'_{max}}^{n_p} \cdot N^2)$, since the dynamical programming procedure does not increase the time complexity beyond the calculation of $h(u, v)$. There are other methods that lead to suboptimal results but runs faster than dynamical programming method. Please refer to (Boullé 2006).

4.4 Approximation

Notice that Algorithm1 leads to the non-polynomial runtime $O({v'_{max}}^{n_p} \cdot N^2)$ in the discretization process. Therefore, we have an empirical approximation to the objective function that can reduce runtime and still preserve the quality of discretization. The approximation is as follows: for the dominator of the last factor in Equation 11, we have

$$\frac{n_i!}{{n^{(p)}}_{i,1}! n_{i,2}^{(p)}! \cdots n_{i,J_P}^{(p)}!} \approx \prod_{r=1}^{n_p} \frac{n_i!}{n_{i,1}^{(p_r)}! n_{i,2}^{(p_r)}! \cdots n_{i,J_{p_r}}^{(p_r)}!}, \tag{16}$$

where $J_{p_r} = ||P_r||$ and $n_{i,j}^{(p_r)}!$ is the number of instances in ith interval with jth value of P_r. Applying this approximation, the approximated objective function

Algorithm 1 Discretization of one continuous variable

function DISCRETIZE(D_X, D, G)

$N \leftarrow$ the number of instances
$N' \leftarrow$ the number of non-repeated values of D_X
$H \leftarrow$ an $N \times N$ matrix such that $H[u, v] = h(u, v)$ as Algorithm 4 in Appendix
$b \leftarrow$ the increasing sequence of M integers defined in the previous discussion
$L \leftarrow$ the largest cardinality over all discrete variables in the Markov blanket
$S[u] \leftarrow$ the optimal objective value of a subproblem with instances from 1 to u
$M[u] \leftarrow$ the optimal discretization of a subproblem with instances from 1 to u
$W[u] \leftarrow -\log(1 - \exp(-L \cdot \frac{x_{b(u)+1} - x_{b(u)}}{x_N - x_1}))$ for $u = 1, 2, \ldots, N' - 1$ and $L(N') \leftarrow 0$

for $v = 1$ to N' **do**
  if $v = 1$ **then**
    $S[v] = g(1, b[v]) + L[v]$
    $M[v] = \{(x_{b[v]} + x_{b[v]+1})/2\}$
  else
    $s \leftarrow \infty$ and *boundary* $\leftarrow \infty$
    for $u = 1$ to v **do**
      $s' \leftarrow S[u] + g(b[u] + 1, b[v]) + L \cdot \frac{x_{b[v]} - x_{b[u]+1}}{x_N - x_1} + W[v]$
      if $s' < s$ **then**
        $s \leftarrow s'$
        boundary $\leftarrow (x_{b[u]} + x_{b[u]+1})/2$
    $S[v] \leftarrow s$
    $M[v] \leftarrow M[u] \cup \{boundary\}$
return M

is

$$\sum_{i=1}^{k-1} -\log(1 - \exp(-L \cdot \frac{x_{s_i+1} - x_{s_i}}{x_N - x_1})) + \sum_{i=1}^{k} L \cdot \frac{x_{s_i} - x_{s_{i-1}+1}}{x_N - x_1} + \sum_{j=1}^{n_c} \sum_{i=1}^{k} \sum_{l=1}^{L_j} \left[\log \binom{n_{i,l} + J_j - 1}{J_j - 1} + \log \left(\frac{n_{i,l}!}{n_{i,1,l}^{(j)}! n_{i,2,l}^{(j)}! \cdots n_{i,J_j,l}^{(j)}!} \right) \right] + \sum_{i=1}^{k} \left[\log \binom{n_i + J_P - 1}{J_P - 1} + \sum_{q=1}^{n_p} \log \left(\frac{n_i!}{n_{i,1}^{(p_q)}! n_{i,2}^{(p_q)}! \cdots n_{i,J_{p_q}}^{(p_q)}!} \right) \right]. \quad (17)$$

As we will see in Section 6, the approximated algorithm is more sensitive to the distribution of other variables' values and usually leads to slightly more discretization edges. With the approximation, we can reduce runtime of Algorithm 1 from $O(n_c v'_{max} \cdot N^2 + {v'_{max}}^{n_p} \cdot N^2)$ to $O(v'_{max}(n_c + n_p) \cdot N^2)$. The rigorous proof of the approximation have not been proposed.

With the approximated objective function, Equation 17, we now have a more clear way to see that how child variables and parent variables contribute to the objective function differently. For example, in the left graph of Figure

2, the corresponding sqaure brackets in Equation 17 are

$$\sum_{i=1}^{k}\left\{\left[\log\binom{n_i+J_{C_1}-1}{J_{C_1}-1}+\log\left(\frac{n_i!}{n_{i,1,\emptyset}^{(1)}!\cdots n_{i,J_{C_1},\emptyset}^{(1)}!}\right)+\right.\right.$$
$$\left.\log\binom{n_i+J_{C_2}-1}{J_{C_2}-1}+\log\left(\frac{n_i!}{n_{i,1,\emptyset}^{(2)}!\cdots n_{i,J_{C_1},\emptyset}^{(2)}!}\right)\right]+$$
$$\left.\left[\log\binom{n_i+J_P-1}{J_P-1}+\log\left(\frac{n_i!}{n_{i,1}^{(p_1)}!\cdots n_{i,J_{p_1}}^{(p_1)}!}\right)+\frac{n_i!}{n_{i,1}^{(p_2}!\cdots n_{i,J_{p_2}}^{(p_2)}!}\right]\right\}$$
.
(18)

In Equation18, each child variable carries two terms, as shown in the first square bracket, and two parent variables only carry three terms, as shown in the second square bracket. Therefore, in this case, child variables are more determinative than parent variables even though the numbers of child variables and parent variables are equal. However, if C_2 has the other parent variable S_2, as shown in the right graph of Figure 2, the importances of C_2 will be debiliated, since the information from C_2 is now adulterated by S_2. These argument shows that our proposed method, either before or after approximation, indeed involves graph structures to discretize continuous variables.

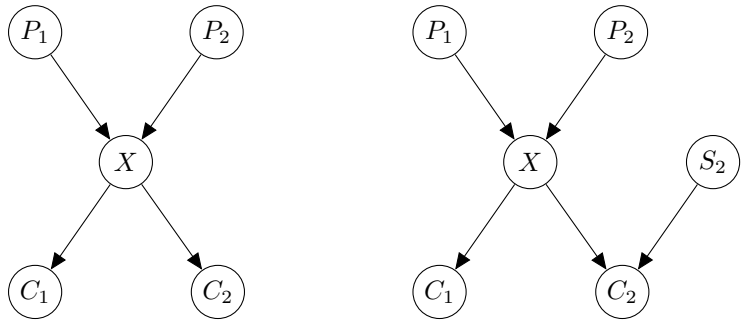

Fig. 2 Two example networks

5 Discretizing Multiple Continuous Variable with Structure Learning

5.1 Discretization of Multiple Continuous Variables

If there are multiple continuous variables in a Bayesian network, we iterate the one-variable discretization method discussed above for all continuous variables.

While discretizing a continuous variable, other continuous variables must be discretized, either by a prediscretization that was done before the first iteration or by the discretization result of the latest iteration. The prediscretization can be done by equal-width discretization, which is defined as follows:

$$M_X = \{min_X, min_X + \delta, min_X + 2\delta, \ldots, \delta, max_X\}, \tag{19}$$

where min_X and max_X are minimal and maximal values of D_X, respectively, $\delta = (max_X - min_X)/k$, and k is the desired number of intervals after equal-width discretization. k is same for all continuous variables and is set to be median value of all discrete variables' cardinalities. After prediscretization, we iterate the one-variable discretization on each continuous variable in the following order: from the continuous variable with highest topological order (leaves) to the continuous variable with lowest topological order (root). We call one such series of discretizations as a cycle. The advange of the order is that, for the first cycle of iteration, we can use less number of unsupervised discretization result. For example, in the right graph of Figure 2, assume S_2 is the only discrete variable. If we begin the iteration with P_1, then the discretization of P_1 will involve the prediscretization results of both P_2 and X. However, if we begin the iteration with C_1, then the discretization of C_1 will only involve the prediscretization result of X.

When the number of intervals after discretization and the positions of discretization edges converge, we stop the iterations and output the discretization policy on each continuous variable. Since the convergence is not guaranteed, we also set up a maximal number of cycles to prevent infinite iterations. In our test on real-world data, the iteration results usually converge within few cycles. Even if it does not converge, 10 cycles usually produce good enough discretization result. The pseudocode of multi-variable discretization is shown in Algorithm 2, where we require four inputs: D'', the mixed data of all variables that we plan to learn from, G, the network structure, C, the set of all continuous variables in a reverse topological order, and u_{cycle}, the upper bound of times of cycles.

5.2 Discretization of Continuous Variables While Structure Learning

In many situations the network structure is not known in advance, then we need to learn it from data. We use an alternative approach to combine our proposed discretization method with the K2 structure learning algorithm (Cooper and Herskovits 1992). That is to say, we start with some prediscretizations on continuous variables and run the K2 algorithm given the discretized data. Each time an edge is added by the K2 algorithm, we rediscretize the continuous variables based on the current network. Then the new discretization policies update the discretized data of all continuous variables, and we continue the next K2 step with the updated discretized data. This cycle is repeated until no edge is added by the K2 algorithm. The detail of the procedure is shown in Algorithm 3. Here we have five inputs: D', the mixed data we plan to learn

Algorithm 2 Discretization of multiple continuous variables

function DISCRETIZE(D'', G, C, u_{cycle})

$M[i] \leftarrow$ the discretization policy of ith variable
$n \leftarrow$ the number of variables in Bayesian network
$D_i^* \leftarrow M[i](D_i)$, the discretized D_i by $M[i]$
$D^* \leftarrow$ the discretized data for all variables, where $D_i^* \leftarrow D_i''$ if $i \notin C$
$k \leftarrow median\{||v||, v \notin C\}$

for $i = 1$ to n **do**
  if $i \in C$ **then**
    $M[i] \leftarrow$ equal-width discretization with k intervals
    $D_i^* \leftarrow M[i](D_i'')$

$cycle \leftarrow 0$
while M is not converged **and** $cycle \leq u_{cycle}$ **do**
  $cycle \leftarrow cycle + 1$
  for $j = 1$ to $||C||$ **do**
    $D^*_{\setminus C(j)} \leftarrow D^*$ without $D^*_{C(j)}$.
    $M[C(j)] \leftarrow$ DISCRETIZE($D_{C(j)}$,$D^*_{\setminus C(j)}$,graph)
    $D^*_{C(j)} \leftarrow M[C(j)](D''_{C(j)})$
return M

from, C, the set of all continuous variables, *order*, the variable order which is required by the $K2$, u_{parent}, the upper bound of number of parents which is also required by the $K2$, and u_{cycle}, the upper bound of times of cycles. Notice that in the usual structure learning from discrete data, one usually run the K2 algorithm many times with different orders, and choose the structure with the highest K2 score. We do the same thing: run Algorithm 3 many times, each time with a different order, and pick the discretized Bayesian network with the highest K2 score.

6 Experiments

References

M. Boullé. Modl: A bayes optimal discretization method for continuous attributes. *Machine Learning*, pages pp131–165, 2006.

G. F. Cooper and E. Herskovits. A bayes method for the induction of probabilistic network from data. *Machine Learning*, 9:pp.309–147, 1992.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stain. *Introduction to algorithms*. The MIT Press, 3 edition, 2009.

J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *12th International Conference on Maching Learning (ICML)*. Morgan Kaufman Publishers, San Francisco, CA, 1995.

Algorithm 3 Learning a discrete-valued Bayesian network

function LEARN_DBN(D'', C, $order$, u_{parent}, C, u_{cycle})

$n \leftarrow$ the number of variables in Bayesian network
$k \leftarrow median\{||v||, v \notin C\}$
$M[i] \leftarrow$ the discretization policy of ith variable
$D_i^* \leftarrow M[i](D_i)$, the discretized D_i by $M[i]$
$D^* \leftarrow$ the discretized data for all variables, where $D_i^* \leftarrow D_i''$ if $i \notin C$
$G \leftarrow$ the initial graph (no edges between nodes)

for $i = 1$ to n **do**
  if $i \in C$ **then**
    $M[i] \leftarrow$ equal-width discretization with k intervals
    $D_i^* \leftarrow M[i](D_i'')$

for $i = 1$ to n **do**
  $P_{old} \leftarrow f(X_i^*, \Pi_{X_i^*})$: Equation 5
  OKToProceed $\leftarrow$ **true**
  while O **do**KToProceed **and** $||\Pi_{X_i^*}|| < u_{parent}$
    $Y \leftarrow$ an element from the set $order[1:i] \backslash \Pi_X$
    $P_{new} \leftarrow f(X_i^*, \Pi_{X_i^*} \cup Y)$
    if $P_{new} > P_{old}$ **then**
      $P_{old} \leftarrow P_{new}$
      $\Pi_{X_i} \leftarrow \Pi_{X_i} \cup Y$
      $M \leftarrow$ DISCRETIZE(D'', G, C, u_{cycle}): Algorithm 2
      $D* = M(D'')$
    else
      OKToProceed $\leftarrow$ **false**
return G, M

U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *13th International Joint Conference on Artificial Intelligence*, pages pp.1022–1027. Morgan Kaufman, 1993.

N. Friedman and M. Goldszmidt. Discretizing continuous attributes while learning bayesian networks. In *13th International Conference on Maching Learning (ICML)*. Morgan Kaufman Publishers, San Francisco, CA, 1996.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:pp.131–163, 1997.

P. D. Grünwald. *Minimum description length principle*. The MIT Press, 2007.

R.C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:pp.63–90, 1993.

A. Ihler and D. McAllester. Particle belief propagation. In *Artificial Intelligence and Statistics*, 2009.

R. Kerber. Chimerge: Discretization of numeric attributes. In *10th National Conference on Artificial Intelligence*, pages pp.123–128. MIT Press, 1992.

M. J. Kochenderfer. *Decision making under uncertainty*. MIT Lincoln Laboratory Series, 2015.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009.

A.V. Kozlov and D. Koller. Nonuniform dynamic discretization in hybrid networks. In *13th International Conference on Uncertainty in Artificial Intelligence (UAI)*, 1997.

J. L. Lustgarten, S. Viswewaran, Gopalakrishnan V., and G. F. Cooper. Application of an efficient bayesian discretization method to biomedical data. *BMC Bioinformatics*, 2011.

S. Monti and G.F. Cooper. A multivariate discretization method for learning bayesian networks from mixed data. In *14th International Conference on Uncertainty in Artificial Intelligence (UAI)*, 1998.

J. Pearl. Probabilistic reasoning in intelligent systems: networks of plausible inference. *Morgan Kaufman Publishers, Inc.*, 1988.

J. Rissanen. Modeling by shortest data description. *Automatica*, pages pp.465–471, 1978.

H. Stech and T. Jaakkola. Predictive discretization during model selection. In *11th International Conference on Artificial Intelligence and Statistics*, 2007.

P. M. B. Vitényi and M. Li. Minimum description length induction, bayesianism, and kolmogorov complexity. *IEEE Transactions on Information Theory*, 2000.

Y. Weiss and W. T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, 2001.

Appendices

Algorithm 4 Calculation of function $h(u, v)$ for all $u \leq v$

Initialize H as an $N \times N$ matrix that all elements are 0.
$count_p$ is an $N \times N \times ||\Pi_X||$ matrix such that $count_p[u, v, w]$ is the number of instances from x_u to x_v with wth value of Π_X. This matrix can be calculated in $O(||\Pi_X|| \cdot N^2)$
for $u = 1$ to N **do**
    for $v = u$ to N **do**
        $H(u, v) \leftarrow H(u, v) + \log((v - u + J_p)!) - \log((J_p - 1)!)$
        for $w = 1$ to $||\Pi_X||$ **do**
            $H(u, v) \leftarrow H(u, v) - \log(count_p(u, v, w)!)$
for $j = 1$ to n_c **do**
    XXX