TEST (2022) 31:299–327
https://doi.org/10.1007/s11749-022-00812-3

INVITED PAPER

Hybrid semiparametric Bayesian networks

David Atienza[1] · **Pedro Larrañaga**[1] · **Concha Bielza**[1]

Received: 8 September 2021 / Accepted: 19 March 2022 / Published online: 13 June 2022

Abstract

This paper presents a new class of Bayesian networks called hybrid semiparametric Bayesian networks, which can model hybrid data (discrete and continuous data) by mixing parametric and nonparametric estimation models. The parametric estimation models can represent a conditional linear Gaussian relationship between variables, while the nonparametric estimation model can represent other types of relationships, such as non-Gaussian and nonlinear relationships. This new class of Bayesian networks generalizes the conditional linear Gaussian Bayesian networks, including them as a special case. In addition, we describe a learning procedure for the structure and the parameters of our proposed type of Bayesian network. This learning procedure finds the best combination of parametric and nonparametric models automatically from data. This requires the definition of a cross-validated score. We also detail how new data can be sampled from a hybrid semiparametric Bayesian network, which in turn can be useful to solve other related tasks, such as inference. Furthermore, we intuitively relate our proposal with adaptive kernel density estimation models. The experimental results show that hybrid semiparametric Bayesian networks are a valuable contribution when dealing with data that do not meet the parametric assumptions that are expected for other models, such as conditional linear Gaussian Bayesian networks. We include experiments with synthetic data and real-world data from the UCI repository which demonstrate the good performance and the ability to extract useful information about the relationship between the variables in the model.

This invited paper is discussed in the comments available at: https://doi.org/10.1007/s11749-022-00815-0, https://doi.org/10.1007/s11749-022-00816-z, https://doi.org/10.1007/s11749-022-00817-y, https://doi.org/10.1007/s11749-022-00818-x.

✉ David Atienza
datienza@fi.upm.es

Pedro Larrañaga
pedro.larranaga@fi.upm.es

Concha Bielza
mcbielza@fi.upm.es

[1] Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Madrid, Spain

Keywords Bayesian networks · Semiparametric model · Kernel density estimation · Hybrid data

Mathematics Subject Classification 68T05 · 68T10

1 Introduction

Nowadays, the large availability of real-world data in multiple fields has driven the attention to extract useful information using machine learning. Bayesian networks (Pearl 1988; Koller and Friedman 2009; Maathuis et al. 2018) are probabilistic graphical models that can represent a multivariate probability distribution in an efficient way by factorizing the probability distribution, taking advantage of the conditional independences of the distribution. Furthermore, these conditional independences of the distribution can be represented using a directed acyclic graph.

Bayesian networks have been applied in many different domains: Bielza and Larrañaga (2014b), Mascaro et al. (2014), and Codetta-Raiteri and Portinale (2015). Bayesian networks are a convenient model when there is uncertainty in the data because they are grounded on the sound framework of probability theory. Bayesian networks are very expressive models since they provide both quantitative and qualitative information. The quantitative information is related to the ability of Bayesian networks to answer different types of queries about the probability of occurrence of a given event. The qualitative information can be extracted reading the graph of the Bayesian network, so useful information about the real-world domain can be recovered. For this reason, many different Bayesian network learning algorithms have been developed in the literature (Scutari et al. 2019). Also, many different types of machine learning problems have been addressed using Bayesian networks: classification (Bielza and Larrañaga 2014a), clustering (Luengo-Sanchez et al. 2019), anomaly detection (Mascaro et al. 2014), etc.

To model the uncertainty, Bayesian networks usually make parametric assumptions, i.e., they assume that the data come from a specific distribution or class of distributions. We call this type of Bayesian networks as parametric Bayesian networks. The parametric assumptions have some advantages: the number of parameters is fixed and the estimation of those parameters is usually faster. Also, if the data truly meet the assumptions, there is usually a minimum variance unbiased estimator for the parameters which exhibits good performance (even though it is not guaranteed to be the estimator with minimum error). However, if the assumptions in the data are not met, the model may have an inherent bias that cannot be reduced even in the case of a large amount of available data. In that case, it is better to use a nonparametric estimation model that does not make assumptions about the data distribution. Nevertheless, a nonparametric estimation model can be computationally more expensive and usually offers worse error convergence results when the parametric assumption is met (Scott 2015). A Bayesian network constructed using nonparametric estimation models will be called nonparametric Bayesian network.

In the present paper, we propose a new class of Bayesian networks that mixes parametric and nonparametric estimation models to combine the advantages of both

Bayesian network types. We call this type semiparametric Bayesian networks. In addition, we include a learning algorithm that automatically detects which parts of the Bayesian networks are best represented with a parametric or nonparametric estimation model. This information can also be read in the graph, so more qualitative information can be extracted about the type of relation between the variables. Also, this type of Bayesian networks can model hybrid probability distributions, i.e., probability distributions that combine discrete and continuous random variables, which are the two most common data types in real applications.

The contributions of the paper are as follows: (1) a new class of Bayesian networks that models hybrid data and mixes parametric and nonparametric estimation models, (2) the adaptation of standard learning techniques to learn the structure and the parameters of hybrid semiparametric Bayesian networks, (3) a procedure to sample new data from this new class of Bayesian network, (4) an intuitive connection between hybrid semiparametric Bayesian networks and the adaptive kernel density estimation models, and (5) the availability of the source code of the proposed framework.

The paper is organized as follows. Section 2 introduces the needed concepts about Bayesian networks and nonparametric models by also reviewing previous works in the literature. In Sect. 3, the hybrid semiparametric Bayesian network class is described, along with a learning algorithm for the structure and parameters of the network. Section 4 provides experimental results obtained by testing hybrid semiparametric Bayesian networks in synthetic and real-world data. Section 5 concludes the paper and provides future work proposals.

2 Background

We denote with capital letters, e.g., X, a random variable, while using the boldface version to represent random variable vectors, e.g., $\mathbf{X}$. A subscript is used to index an element or set of elements, e.g., X_i is the i-th element of $\mathbf{X}$ and $\mathbf{X}_S$, with $S \subset \{1, \ldots, n\}$, selects the set S of indices from a vector $\mathbf{X}$ of n variables. The instantiation of random variables is denoted using lowercase letters, e.g., x, x_i or $\mathbf{x}_S$. The vector of continuous random variables are denoted using $\mathbf{Y}$. The vector of discrete random variables will be denoted using $\mathbf{Z}$. The number of discrete variables is d and the number of continuous variables is c, so $n = d + c$. The domain of the discrete variable Z_i is denoted $\mathbf{\Omega}_i$, and the domain of the discrete variable $\mathbf{Z}_S$ is $\mathbf{\Omega}_S = \times_{i \in S} \mathbf{\Omega}_i$. The union of continuous and discrete random variables is denoted with $\mathbf{X} = (\mathbf{Y}, \mathbf{Z})$. A dataset is denoted with $\mathcal{D}$, which can also be indexed by indices, $\mathcal{D}_i$, and sets of indices, $\mathcal{D}_S$, to select a subset of $\mathcal{D}$ with only the indexed variables. Also, a subset of the data with the instances having a discrete configuration $\mathbf{z}$ for the variables $\mathbf{Z}$ is denoted $\mathcal{D}_{\downarrow \mathbf{z}}$.

2.1 Kernel density estimation

Kernel density estimation (KDE) (Scott 2015) is a nonparametric technique that estimates the underlying probability density function of some continuous data. The

multivariate kernel density estimation is defined as:

$$\hat{f}_{\text{KDE}}(\mathbf{y}) = \frac{1}{N\,|\mathbf{H}|^{1/2}} \sum_{i=1}^{N} K\left(\mathbf{H}^{-1/2}(\mathbf{y}-\mathbf{y}^i)\right) = \frac{1}{N}\sum_{i=1}^{N} K_{\mathbf{H}}\left(\mathbf{y}-\mathbf{y}^i\right), \quad (1)$$

where $\mathbf{y}^i = (y_1^i, \ldots, y_c^i)$ is the i-th instance of the training dataset $\mathcal{D}_{\mathbf{Y}} = \{\mathbf{y}^1, \ldots, \mathbf{y}^N\}$, which contains N instances. $\mathbf{H}$ is a $c \times c$ symmetric positive-definite matrix called bandwidth matrix, and $|\mathbf{H}|$ is its determinant. $K : \mathbb{R}^c \mapsto \mathbb{R}$ is a kernel function whose integral is 1. $K_{\mathbf{H}}(\mathbf{x}) = |\mathbf{H}|^{-1/2} K(\mathbf{H}^{-1/2}\mathbf{x})$ is the scaled version of the kernel function by the bandwidth matrix.

Usually, a Gaussian kernel, $K(\mathbf{x}) = (2\pi)^{-c/2} \exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{x}\right)$, is used because it is a distribution with good theoretical properties and some interesting derived results for KDEs. The scaled version of the kernel is equal to the multivariate Gaussian density with covariance matrix $\mathbf{H}$ and mean $\mathbf{y}^i$:

$$K_{\mathbf{H}}(\mathbf{y}-\mathbf{y}^i) = \frac{1}{(2\pi)^{c/2}\,|\mathbf{H}|^{1/2}} \exp\left[(\mathbf{y}-\mathbf{y}^i)^T\mathbf{H}^{-1}(\mathbf{y}-\mathbf{y}^i)\right].$$

Thus, a KDE model with a Gaussian kernel is equivalent to a Gaussian mixture model with equiprobable components where each component is located on each training instance. From this perspective, it is easy to understand the effect of $\mathbf{H}$ on the resulting density estimation $\hat{f}_{\text{KDE}}(\mathbf{y})$. If $|\mathbf{H}|$ is small, the space near the training instances has a high probability density and the space away from any training instance will have probability density close to 0. Conversely, if $|\mathbf{H}|$ is large, the probability density is spread over the entire space, resulting in a smooth density estimate.

The bandwidth matrix has an important impact in the performance of the KDE. For this reason, multiple techniques have been developed to select the bandwidth matrix automatically. Most of the research was dedicated to the bandwidth selection in the univariate case (Cao et al. 1994). In the multivariate case, bandwidth selection can be more challenging, although many univariate bandwidth selection techniques have been adapted to it (Chacón and Duong 2018). The bandwidth matrix can be constrained to have a specific structure, e.g., a diagonal bandwidth matrix. However, in this manuscript we will work with unconstrained bandwidth matrices. Many bandwidth selection techniques are based on minimizing the mean integrated squared error (MISE):

$$\text{MISE}\{\hat{f}_{\text{KDE}}\} = \mathbb{E}\int_{\mathbb{R}^c}\left[f(\mathbf{y}) - \hat{f}_{\text{KDE}}(\mathbf{y})\right]^2 d\mathbf{y}, \quad (2)$$

where f is the underlying probability density function that we are estimating, and the expectation is carried out with respect to the different training datasets that can be used to create the estimator $\hat{f}_{\text{KDE}}$. Usually, the MISE cannot be calculated directly because function f is unknown. The normal reference rule is a bandwidth selection technique that assumes that f is the Gaussian distribution. Then, the bandwidth selection that optimizes the MISE is:

$$\hat{\mathbf{H}} = \left(\frac{4}{N(c+2)}\right)^{2/(c+4)} \hat{\boldsymbol{\Sigma}}, \quad (3)$$

where $\hat{\mathbf{\Sigma}}$ is the sample covariance. The normal reference rule is known to oversmooth when the data distribution is not Gaussian. That is, the normal reference rule has an inherent bias toward larger bandwidth matrices when f is not Gaussian. An alternative is the unbiased cross-validation (UCV) criterion (Rudemo 1982; Bowman 1984), which performs a cross-validation on the data to estimate the MISE. Although UCV is an unbiased estimator, it exhibits a considerable variance in most cases. The plug-in selection method (Duong and Hazelton 2003) is based on minimizing the asymptotic MISE (AMISE), which is asymptotically equal to MISE (when $N \to \infty$). The plug-in method requires the estimation of some derivatives of f, which can be calculated taking the derivative of K in Eq. (1).

The bandwidth matrix is constant in $\hat{f}_{\text{KDE}}$, so the same amount of smoothing is applied for all the training instances. If the distribution of the data differs in different parts of the space, this could be suboptimal. For this reason, some works have proposed to use different bandwidth matrices in different parts of the space. This type of density estimator has been called adaptive kernel density estimator (AKDE). This idea has been implemented in two different ways. The bandwidth matrix could depend on a function of $\mathbf{y}$ (Loftsgaarden and Quesenberry 1965), and then:

$$\hat{f}_{\text{adaptive}_1}(\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} K_{\mathbf{H}(\mathbf{y})}\left(\mathbf{y} - \mathbf{y}^i\right), \tag{4}$$

where $\mathbf{H}(\mathbf{y})$ is a function that returns a bandwidth matrix. Alternatively, each training instance can have a different bandwidth matrix (Breiman et al. 1977):

$$\hat{f}_{\text{adaptive}_2}(\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} K_{\mathbf{H}(\mathbf{y}^i)}\left(\mathbf{y} - \mathbf{y}^i\right). \tag{5}$$

where $\mathbf{H}(\mathbf{y}^i)$ is a function that returns a bandwidth matrix for each training instance.

2.2 Bayesian networks

Formally, a Bayesian network is a tuple $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$, where $\mathcal{G} = (V, A)$ is a directed acyclic graph (DAG) with a set of nodes $V = \{1, \ldots, n\}$ and a set of arcs $A \subseteq V \times V$, and $\boldsymbol{\theta} = \{P\left(x_i \mid \mathbf{x}_{\text{Pa}(i)}\right), i = 1, \ldots, n\}$ is a set of parameters that contains a conditional probability distribution (CPD) for each random variable. A Bayesian network factorizes a joint probability distribution $P(\mathbf{x})$ of a vector of random variables $\mathbf{X} = (X_1, \ldots, X_n)$. The set of nodes V indexes the vector of random variables, so $\mathbf{X}_V = \mathbf{X}$. The form in which the joint probability distribution is factorized depends on the set of arcs A of $\mathcal{G}$:

$$P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i \mid \mathbf{x}_{\text{Pa}(i)}), \tag{6}$$

where $\text{Pa}(i)$ are the parent indices of node i in the graph $\mathcal{G}$. $P(x_i \mid \mathbf{x}_{\text{Pa}(i)})$ is the CPD of the random variable X_i given the random variables $\mathbf{X}_{\text{Pa}(i)}$ contained in $\boldsymbol{\theta}$.

The number of parameters of an unfactorized distribution $P(\mathbf{x})$ is usually larger than $O(n)$, e.g., it is quadratic, $O(n^2)$, for a multivariate Gaussian distribution, or exponential, $\prod_{i=1}^{n} |\mathbf{\Omega}_i|$, for a conditional probability table (CPT) of a categorical distribution. Since usually $|\text{Pa}(i)| \ll n$, the set $\boldsymbol{\theta}$ contains n CPDs with a very small number of parameters. Thus, the number of parameters needed to define a Bayesian network is usually much lower than the unfactorized representation of $P(\mathbf{x})$.

Moreover, the set of conditional independences defined by the Bayesian network can be directly read from the graph G using the d-separation criterion (Koller and Friedman 2009).

2.2.1 Parametric Bayesian networks

Most of the Bayesian networks defined in the literature are parametric, i.e., they are defined using parametric CPDs with a fixed number of parameters. The most common type of Bayesian networks are discrete Bayesian networks, that only can be used to model discrete domains. Usually, discrete Bayesian networks are defined using a CPT that specifies a conditional probability for each configuration of X_i and $\mathbf{X}_{\text{Pa}(i)}$. Other types of CPDs are available such as tree-structured CPDs (Friedman and Goldszmidt 1996), which can take advantage of the conditional independences in the CPD to reduce the needed number of parameters.

Gaussian Bayesian networks are a specific continuous Bayesian network type. A Gaussian Bayesian network uses a linear Gaussian (LG) CPD for each node of the Bayesian network. The LG CPD assumes that there is a linear relationship between X_i and $\mathbf{X}_{\text{Pa}(i)}$, and there is a Gaussian conditional distribution given $\mathbf{x}_{\text{Pa}(i)}$. A Gaussian Bayesian network assumes that the random variables are distributed according to a multivariate Gaussian distribution. Indeed, a Gaussian Bayesian network can be considered an alternative representation of a multivariate Gaussian distribution. Since the normality assumption can be inadequate to model non-Gaussian distributions, other types of Bayesian networks have been developed. Mixtures of Gaussian DAG models (MGDAG) (Thiesson et al. 1998) are analogous to the Gaussian mixture model but they use Gaussian Bayesian networks to represent the base components of the mixture. Tweedie Bayesian networks (TDB) (Masmoudi and Masmoudi 2019), restrict the CPDs to be a special class of distributions from the exponential family. The Tweedie class of distributions include the Gaussian, the inverse-Gaussian, the gamma, and the Poisson distributions.

The Bayesian networks that can model domains with discrete and continuous variables are called hybrid Bayesian networks. A conditional linear Gaussian Bayesian network (CLGBN), Lauritzen and Wermuth (1989), is a hybrid Bayesian network that assumes that the discrete random variables can be represented with a CPT, and the continuous random variables are assumed to be conditionally distributed with an LG CPD or a set of LG CPDs. A CLGBN restricts the structure of the network, so arcs from continuous random variables to discrete random variables are not allowed. Also, the assumption of an LG CPD implies, for each possible configuration of the discrete parents, a Gaussian conditional distribution. Some efforts have been made to increase the flexibility of the CLGBN model. The augmented CLGBN (Lerner et al. 2001) addresses the restriction on the continuous parents for discrete variables using a

softmax function as CPD for the discrete variables. Mixture of truncated exponentials (MTE) networks (Moral et al. 2001) describe piecewise-defined exponential functions that are used as CPD. The MTE networks overcome the normality assumption by dividing the domain in multiple hypercubes and assigning a different exponential function to each domain partition. This idea inspired the mixture of polynomials (MoP) networks (Shenoy and West 2011) and mixture of truncated basis functions (MoTBF) networks (Langseth et al. 2012), which use polynomials and basis functions, respectively, in each domain partition.

2.2.2 Nonparametric Bayesian networks

When the parametric assumptions are not met, the performance of parametric models may be greatly reduced. An alternative is the use of nonparametric models that do not make assumptions about the data distribution. A nonparametric model does not have a fixed number of parameters and usually the complexity of its representation grows with the amount of data available.

A nonparametric Bayesian network is a Bayesian network where all the CPDs of a Bayesian network are estimated with nonparametric methods. Hofmann and Tresp (1995) define a type of nonparametric Bayesian network where the CPDs are defined as the ratio of two KDE models. The use of KDE models was extended to create flexible Bayesian network classifiers. A flexible naive Bayes is described in John and Langley (1995), and other flexible Bayesian networks with fewer structure constraints are proposed in Pérez et al. (2009). Gonzalez et al. (2015) apply a flexible Bayesian network to detect anomalies in an industrial context.

Friedman and Nachman (2000) define Bayesian networks using Gaussian processes, which are also nonparametric methods. These type of Bayesian networks are designed to detect functional dependencies between the variables. Ickstadt et al. (2012) introduce a continuous nonparametric Bayesian network that uses an infinite mixture model to avoid the normality assumption. This Bayesian network is learned taking into account the uncertainty of the parameters and the graph of a Gaussian network and using Bayesian priors over the parameters.

2.2.3 Semiparametric Bayesian networks

A semiparametric model combines parametric and nonparametric components. They try to get the advantages of both types of models. Not much work has been done on semiparametric Bayesian networks. Boukabour and Masmoudi (2020) define a continuous semiparametric Bayesian network that rely on semiparametric regression. The semiparametric regression combines a linear regression component with a Nadaraya–Watson estimator (Nadaraya 1964; Watson 1964), that uses a diagonal bandwidth matrix for the nonparametric component. In this work, all the CPDs are semiparametric regressions. The graph structure of these networks is learned using a novel algorithm based on statistical hypothesis tests of conditional independence that requires to know the correct ancestral ordering for the nodes.

Atienza et al. (2022) proposed continuous semiparametric Bayesian networks, where a CPD can be parametric or nonparametric. The parametric CPDs are LG

CPDs and the nonparametric CPDs are the ratio of two KDE joint distributions using unconstrained bandwidth matrices. The learning algorithm automatically detects the type of CPD suitable for each node. These networks can be learned using any of the score-and-search and constraint-based algorithms already developed in the state of the art, such as greedy hill-climbing and PC (Spirtes et al. 2001).

To the best of our knowledge, no previous approach supported hybrid data using nonparametric/semiparametric Bayesian networks.

3 Hybrid semiparametric Bayesian networks

This section proposes the class of hybrid semiparametric Bayesian networks (HSPBN). First, we describe their representation in Sect. 3.1, which describes the CPDs of HSPBNs. In Section 3.2, we propose a learning process of HSPBNs. Then, we present a procedure to sample new data from an HSPBN in Sect. 3.3. Finally, Sect. 3.4 discusses the relation between HSPBNs and the adaptive KDE models [Eqs. (4) and (5)].

3.1 Representation

An HSPBN is a class of Bayesian network that can model discrete and continuous variables. The discrete variables are conditionally distributed as a categorical distribution. This distribution is usually represented by a CPT. Like CLGBNs, the graph is restricted so that a discrete variable cannot have a continuous variable as parent. The continuous variables can be represented using parametric or nonparametric CPDs. The parametric CPDs make the assumptions that, for each possible configuration of the discrete parents, the continuous variables are conditionally distributed as a Gaussian distribution and have a linear relationship with their continuous parents. When this parametric assumption is not met, the HSPBN model can have more flexibility by using a nonparametric CPD, which does not make any parametric assumption.

The HSPBN model includes the CLGBNs as a special case when all the continuous variables CPDs are parametric. However, the HSPBNs have the possibility of more flexible models when nonparametric CPDs are used for some variables.

Figure 1 presents an example of HSPBN. Discrete nodes can only have other discrete nodes as parents. For the continuous nodes, any combination of parent node types is allowed.

3.1.1 Parametric conditional probability distribution

The parametric CPDs in HSPBNs are conditional linear Gaussian (CLG) CPDs, which are also used by CLGBNs. The CLG CPD is composed of an LG CPD for each discrete configuration of its evidence.

Definition 1 Linear Gaussian CPD. Let X_i be a continuous random variable and $\mathbf{Y} = (Y_1, \ldots, Y_r)$ a vector of continuous random variables. The conditional distribution of X_i given $\mathbf{Y}$ is said to be linear Gaussian if there are parameters $\beta_0, \ldots, \beta_r$ and σ_i^2

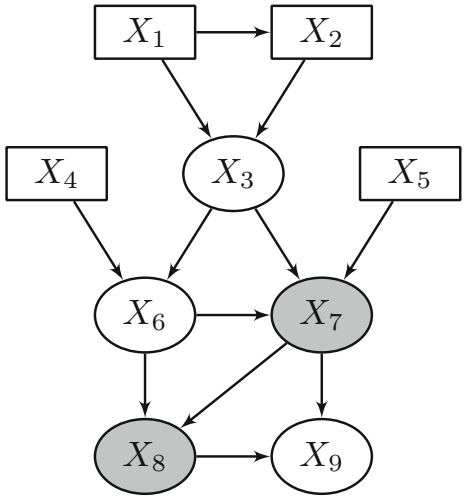

Fig. 1 Example representing the structure of an HSPBN. Discrete variable nodes are represented with rectangles and continuous variable nodes with ellipses. Parametric CPD nodes are represented with white nodes and nonparametric CPDs nodes are represented with gray shaded nodes

such that:

$$f_{\mathrm{LG}}(x_i \mid \mathbf{y}) = \mathcal{N}\left(\beta_0 + \sum_{j=1}^{r} \beta_j y_j, \sigma_i^2\right), \tag{7}$$

where the mean of the Gaussian distribution is computed as a linear regression, β_0 is the intercept of the linear regression, β_j $(j = 1, \ldots, r)$ is the regression coefficient corresponding to Y_j, and σ_i^2 is the variance of the conditional distribution. The LG CPD is derived from the assumption of a linear relationship between X_i and Y_j:

$$X_i = \beta_0 + \sum_{j=1}^{r} \beta_j Y_j + \epsilon, \quad \text{with} \quad \epsilon \sim \mathcal{N}(0, \sigma_i^2),$$

where $Y_1, \ldots, Y_r$ are independent of ϵ.

Note that if all the CPDs of a Bayesian network are LG, as in Gaussian Bayesian networks, the marginal and conditional distributions of each random variable in the Bayesian network are Gaussian. Also, the joint distribution is multivariate Gaussian.

Definition 2 Conditional Linear Gaussian CPD. Let X_i be a continuous random variable, $\mathbf{Y} = (Y_1, \ldots, Y_r)$ a vector of continuous random variables, and $\mathbf{Z} = (Z_1, \ldots, Z_p)$ a vector of discrete random variables. The conditional distribution of X_i given $\mathbf{Y}$ and $\mathbf{Z}$ is said to be conditional linear Gaussian if there are parameters $\beta_{\mathbf{z},0}, \ldots, \beta_{\mathbf{z},r}$ and $\sigma_{\mathbf{z},i}^2$ for each discrete configuration $\mathbf{z} \in \mathbf{\Omega}_{\{1,\ldots,p\}}$ such that:

$$f_{\mathrm{CLG}}(x_i \mid \mathbf{y}, \mathbf{z}) = \mathcal{N}\left(\beta_{\mathbf{z},0} + \sum_{j=1}^{r} \beta_{\mathbf{z},j} y_j, \sigma_{\mathbf{z},i}^2\right), \tag{8}$$

where the mean of the Gaussian distribution of each discrete configuration $\mathbf{z}$ is computed as a linear regression, $\beta_{\mathbf{z},0}$ is the intercept for the discrete configuration $\mathbf{z}$, $\beta_{\mathbf{z},j}$ $(j = 1, \ldots, r)$ is the regression coefficient corresponding to Y_j for the discrete configuration $\mathbf{z}$, and $\sigma^2_{\mathbf{z},i}$ is the variance of the conditional distribution for discrete configuration $\mathbf{z}$.

If all the CPDs of a Bayesian network are CLGs, as in CLGBNs, the conditional distribution of each random variable given their parents ($\mathbf{Y}$ and $\mathbf{Z}$) is Gaussian, but its marginal distribution $f(x_i)$ may be non-Gaussian. Indeed, the marginal distribution of each variable can be represented using a Gaussian mixture where each component of the mixture is constructed from each discrete configuration of $\mathbf{Z}$. Thus, the CLG CPDs allow to relax the normality assumptions, so only the conditional (and not the marginal) distribution given the discrete parents needs to be Gaussian.

Note that the LG CPD is a particular case of the CLG CPD, where there are no discrete conditioning parents.

3.1.2 Nonparametric conditional probability distribution

The nonparametric CPDs of an HSPBN are based on the conditional kernel density estimation (CKDE) described in Hofmann and Tresp (1995) and Atienza et al. (2022). The CKDE is defined as the ratio of two joint distributions estimated with KDE

Definition 3 Conditional Kernel Density Estimation. Let X_i be a continuous random variable and $\mathbf{Y} = (Y_1, \ldots, Y_r)$ a vector of continuous random variables. Then, the conditional distribution of X_i given $\mathbf{Y}$ is said to be conditional kernel density estimation if it is calculated from two KDE models [Eq. (1)] $\hat{f}_{\text{KDE}}(x_i, \mathbf{y})$ and $\hat{f}_{\text{KDE}}(\mathbf{y})$ such that:

$$\hat{f}_{\text{CKDE}}(x_i \mid \mathbf{y}) = \frac{\hat{f}_{\text{KDE}}(x_i, \mathbf{y})}{\hat{f}_{\text{KDE}}(\mathbf{y})} = \frac{\sum_{j=1}^{N} K_{\mathbf{H}}\left(\begin{bmatrix} x_i \\ \mathbf{y} \end{bmatrix} - \begin{bmatrix} x_i^j \\ \mathbf{y}^j \end{bmatrix}\right)}{\sum_{j=1}^{N} K_{\mathbf{H}_{-i}}\left(\mathbf{y} - \mathbf{y}^j\right)}, \tag{9}$$

where x_i^j and $\mathbf{y}^j$ are the values in the j-th training instance for the variables X_i and $\mathbf{Y}$, respectively. $\mathbf{H}$ and $\mathbf{H}_{-i}$ are the bandwidth matrices for the KDE models $\hat{f}_{\text{KDE}}(x_i, \mathbf{y})$ and $\hat{f}_{\text{KDE}}(\mathbf{y})$, respectively.

The CKDE CPD does not assume the marginal or conditional distribution of X_i. However, the CKDE CPD only supports continuous evidence. The hybrid conditional kernel density estimation (HCKDE) supports also discrete evidence by defining a CKDE for each discrete evidence configuration, inspired by the CLG CPD approach.

Definition 4 Hybrid Conditional Kernel Density Estimation. Let X_i be a continuous random variable, $\mathbf{Y} = (Y_1, \ldots, Y_r)$ a vector of continuous random variables, and $\mathbf{Z} = (Z_1, \ldots, Z_p)$ a vector of discrete random variables. The conditional distribution of X_i given $\mathbf{Y}$ and $\mathbf{Z}$ is said to be hybrid conditional kernel density estimation if for every discrete configuration $\mathbf{z} \in \mathbf{\Omega}_{\{1,\ldots,p\}}$, its conditional distribution is modeled by a

different CKDE CPD $\hat{f}_{\text{CKDE},\mathbf{z}}(x_i \mid \mathbf{y})$ such that:

$$\hat{f}_{\text{HCKDE}}(x_i \mid \mathbf{y}, \mathbf{z}) = \hat{f}_{\text{CKDE},\mathbf{z}}(x_i \mid \mathbf{y}) = \frac{\sum_{j=1:\mathbf{z}^j=\mathbf{z}}^{N} K_{\mathbf{H}(\mathbf{z})}\left(\begin{bmatrix} x_i \\ \mathbf{y} \end{bmatrix} - \begin{bmatrix} x_i^j \\ \mathbf{y}^j \end{bmatrix}\right)}{\sum_{j=1:\mathbf{z}^j=\mathbf{z}}^{N} K_{\mathbf{H}_{-i}(\mathbf{z})}\left(\mathbf{y} - \mathbf{y}^j\right)}, \quad (10)$$

where $\hat{f}_{\text{CKDE},\mathbf{z}}$ is a CKDE constructed with the instances with the discrete evidence $\mathbf{z}$, $\mathbf{H}(\mathbf{z})$ and $\mathbf{H}_{-i}(\mathbf{z})$ are bandwidth matrices that depend on the discrete evidence configuration $\mathbf{z}$.

An HCKDE CPD does not require assumptions about the marginal or conditional distribution of X_i. Note that this is a difference with respect to CLG, which assumes a conditional Gaussian distribution.

3.2 Learning

A Bayesian network can be constructed by taking advantage of knowledge from experts of the domain or automatically from data. In this work, we focus our attention on learning automatically from data because in several application fields large amount of data are available.

There are two parts on the learning process of a Bayesian network: parameter learning and structure learning. The parameter learning estimates the parameters of the set of CPDs, $\boldsymbol{\theta}$, for a given structure. The structure learning estimates the graph, $\mathcal{G}$, of the Bayesian network. For most Bayesian networks, the structure learning involves learning the arcs in the graph. In addition, HSPBNs contain extra information specifying the type of CPD for each node.

Usually, the learning process involves first performing structure learning to find the best Bayesian network structure. Then, parameter learning is applied on the best found structure.

3.2.1 Parameter learning

The parameter learning requires knowing the type of CPD and the parents of each node to estimate the parameters of each CPD, $P(x_i \mid \mathbf{x}_{\text{Pa}(i)})$. In this section, we describe the parameter learning process for each CPD type.

For many parametric CPDs a well-known parameter learning criterion is maximum likelihood criterion. The maximum likelihood estimate (MLE) selects the parameter that maximizes the likelihood of the data. For convenience, usually the log of the likelihood, log-likelihood, is used as the criterion because it returns the same estimations as the likelihood. The log-likelihood of some data $\mathcal{D}$ given a graph structure $\mathcal{G}$ and parameters $\boldsymbol{\theta}$ is:

$$\mathcal{L}(\mathcal{G}, \boldsymbol{\theta} : \mathcal{D}) = \sum_{i=1}^{n} \sum_{j=1}^{N} \log P\left(x_i^j \mid \mathbf{x}_{\text{Pa}(i)}^j\right), \quad (11)$$

which is derived from Eq. (6). Then, the MLE is:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\Theta} \mathcal{L}(\mathcal{G}, \boldsymbol{\theta} : \mathcal{D}), \tag{12}$$

where Θ is the parameter space, namely, the set of allowable parameters.

A categorical CPD is usually represented with a CPT where each conditional probability is defined by parameters $P(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)}) = \theta_{x_i|\mathbf{x}_{\mathrm{Pa}(i)}}$, with $\sum_{k\in\boldsymbol{\Omega}_i} \theta_{k|\mathbf{x}_{\mathrm{Pa}(i)}} = 1$. For a categorical CPD, the MLE are the observed frequency estimates of the discrete configurations in the data:

$$\hat{\theta}^{\mathrm{MLE}}_{x_i|\mathbf{x}_{\mathrm{Pa}(i)}} = \frac{N[x_i, \mathbf{x}_{\mathrm{Pa}(i)}]}{N[\mathbf{x}_{\mathrm{Pa}(i)}]}, \tag{13}$$

where $N[x_i, \mathbf{x}_{\mathrm{Pa}(i)}]$ is the number of instances in the data where $X_i = x_i$ and $\mathbf{X}_{\mathrm{Pa}(i)} = \mathbf{x}_{\mathrm{Pa}(i)}$. $N[\mathbf{x}_{\mathrm{Pa}(i)}]$ is defined similarly.

In practice, the MLE may be inconvenient for short samples, specially when $N[x_i, \mathbf{x}_{\mathrm{Pa}(i)}] = 0$, if the ground truth probability $P(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$ is not 0. In that case, the probability of an instance with $X_i = x_i$ and $\mathbf{X}_{\mathrm{Pa}(i)} = \mathbf{x}_{\mathrm{Pa}(i)}$ is equal to 0 and the log-likelihood is not defined. Moreover, the MLE is not defined when $N[\mathbf{x}_{\mathrm{Pa}(i)}] = 0$. This problem can be addressed using a Bayesian prior for the categorical CPD. In this paper, we use a uniform Bayesian Dirichlet equivalent (BDeu) prior (Heckerman et al. 1995), which assigns a uniform Dirichlet prior distribution (which is the conjugate prior of the categorical distribution) to each categorical variable in the Bayesian network:

$$\hat{\theta}^{\mathrm{BDeu}}_{x_i|\mathbf{x}_{\mathrm{Pa}(i)}} = \frac{\alpha / \left|\boldsymbol{\Omega}_{\{i\}\cup\mathrm{Pa}(i)}\right| + N[x_i, \mathbf{x}_{\mathrm{Pa}(i)}]}{\alpha / \left|\boldsymbol{\Omega}_{\mathrm{Pa}(i)}\right| + N[\mathbf{x}_{\mathrm{Pa}(i)}]}. \tag{14}$$

where $\alpha > 0$ is the equivalent sample size of the prior. In the limiting case $\alpha = 0$ the BDeu prior is not defined, but when α is close to 0 the BDeu estimation is close to the MLE. In this paper, we use $\alpha = 1$.

The CLG CPD for X_i given $\mathbf{X}_{\mathrm{Pa}(i)}$ requires computing the set of coefficients and variances, $\boldsymbol{\theta}_{X_i|\mathbf{X}_{\mathrm{Pa}(i)}} = \{\boldsymbol{\beta}_{\mathbf{z}}, \sigma^2_{\mathbf{z},i} \mid \mathbf{z} \in \boldsymbol{\Omega}_{\mathrm{Pa}_{\mathbf{Z}}(i)}\}$, where the vector $\boldsymbol{\beta}_{\mathbf{z}} = (\beta_{\mathbf{z},0}, \beta_{\mathbf{z},1}, \ldots, \beta_{\mathbf{z},r})$ combines the intercept and the linear regression coefficients for discrete configuration $\mathbf{z}$, and $\mathrm{Pa}_{\mathbf{Z}}(i)$ are the discrete parent indices (similarly, we will use $\mathrm{Pa}_{\mathbf{Y}}(i)$ to denote the continuous parent indices). The MLE for the CLG is computed calculating the MLE of each LG CPD. The MLE of an LG CPD can be found using ordinary least squares under the assumptions in Definition 1 (Fox 1997). Thus, the parameters of a CLG can be found applying ordinary least squares for each discrete configuration $\mathbf{z} \in \boldsymbol{\Omega}_{\mathrm{Pa}_{\mathbf{Z}}(i)}$ on the subset of data $\mathcal{D}_{\downarrow\mathbf{z}}$.

The HCKDE CPD for X_i given $\mathbf{X}_{\mathrm{Pa}(i)}$ needs to estimate a CKDE model, $\hat{f}_{\mathrm{CKDE},\mathbf{z}}$, for each discrete parent configuration $\mathbf{z} \in \boldsymbol{\Omega}_{\mathrm{Pa}_{\mathbf{Z}}(i)}$. A CKDE CPD is composed of two KDE models $\hat{f}_{\mathrm{KDE}}(x_i, \mathbf{x}_{\mathrm{Pa}(i)})$ and $\hat{f}_{\mathrm{KDE}}(\mathbf{x}_{\mathrm{Pa}(i)})$. These two KDE models are closely related because the CKDE models need to integrate to 1:

$$\int_{-\infty}^{\infty} \hat{f}_{\mathrm{CKDE}}(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)}) \mathrm{d}x_i = 1, \quad \forall \mathbf{x}_{\mathrm{Pa}(i)},$$

which implies from Eq. (9) that:

$$\int_{-\infty}^{\infty} \sum_{j=1}^{N} K_{\mathbf{H}} \left(\begin{bmatrix} x_i \\ \mathbf{x}_{\mathrm{Pa}(i)} \end{bmatrix} - \begin{bmatrix} x_i^j \\ \mathbf{x}_{\mathrm{Pa}(i)}^j \end{bmatrix} \right) \mathrm{d}x_i = \sum_{j=1}^{N} K_{\mathbf{H}_{-i}} \left(\mathbf{x}_{\mathrm{Pa}(i)} - \mathbf{x}_{\mathrm{Pa}(i)}^j \right).$$

Using Fubini's theorem to switch the integral and the summation we have:

$$\sum_{j=1}^{N} \int_{-\infty}^{\infty} K_{\mathbf{H}} \left(\begin{bmatrix} x_i \\ \mathbf{x}_{\mathrm{Pa}(i)} \end{bmatrix} - \begin{bmatrix} x_i^j \\ \mathbf{x}_{\mathrm{Pa}(i)}^j \end{bmatrix} \right) \mathrm{d}x_i = \sum_{j=1}^{N} K_{\mathbf{H}_{-i}} \left(\mathbf{x}_{\mathrm{Pa}(i)} - \mathbf{x}_{\mathrm{Pa}(i)}^j \right).$$

If $K_{\mathbf{H}}$ is a Gaussian multivariate kernel, the integral can be easily computed. If the $\mathbf{H}$ matrix is defined by blocks:

$$\mathbf{H} = \begin{bmatrix} a & \mathbf{b}^T \\ \mathbf{b} & \mathbf{C} \end{bmatrix}, \tag{15}$$

the integral of $K_{\mathbf{H}}$ is a kernel $K_{\mathbf{C}}$ verifying:

$$\sum_{j=1}^{N} K_{\mathbf{C}} \left(\mathbf{x}_{\mathrm{Pa}(i)} - \mathbf{x}_{\mathrm{Pa}(i)}^j \right) = \sum_{j=1}^{N} K_{\mathbf{H}_{-i}} \left(\mathbf{x}_{\mathrm{Pa}(i)} - \mathbf{x}_{\mathrm{Pa}(i)}^j \right). \tag{16}$$

The expression in Eq. (16) is true for any dataset $\mathcal{D}$ if and only if $\mathbf{H}_{-i} = \mathbf{C}$. Thus, to select the bandwidths of the CKDE, we only need to select $\mathbf{H}$ and assign $\mathbf{H}_{-i} = \mathbf{C}$. The bandwidth $\mathbf{H}$ can be selected with any of the bandwidth selection methods of Sect. 2.1.

Algorithm 1 summarizes the learning process of an HCKDE CPD. For every discrete configuration of the parent variables $\mathbf{z}$, a new CKDE must be learned, which also requires learning two KDE models. The HCKDE CPD is the combination of all the trained CKDE CPDs.

Algorithm 1 HCKDE parameter learning

Require: Training data $\mathcal{D}$, variable X_i, evidence variables $\mathbf{X}_{\mathrm{Pa}(i)}$
1: **for** $\mathbf{z}$ in $\mathbf{\Omega}_{\mathrm{Pa}_{\mathbf{Z}}(i)}$ **do**
2: Estimate bandwidth $\hat{\mathbf{H}}$ from $\mathcal{D}_{\{i\}\cup\mathrm{Pa}_{\mathbf{Y}}(i),\downarrow\mathbf{z}}$
3: $\hat{\mathbf{H}}_{-i} \leftarrow \hat{\mathbf{C}}$
4: $\hat{f}_{\mathrm{KDE},\mathbf{z}}\left(x_i, \mathbf{x}_{\mathrm{Pa}_{\mathbf{Y}}(i)}\right) \leftarrow$ KDE constructed with $\hat{\mathbf{H}}$ and $\mathcal{D}_{\{i\}\cup\mathrm{Pa}_{\mathbf{Y}}(i),\downarrow\mathbf{z}}$
5: $\hat{f}_{\mathrm{KDE},\mathbf{z}}\left(\mathbf{x}_{\mathrm{Pa}_{\mathbf{Y}}(i)}\right) \leftarrow$ KDE constructed with $\hat{\mathbf{H}}_{-i}$ and $\mathcal{D}_{\mathrm{Pa}_{\mathbf{Y}}(i),\downarrow\mathbf{z}}$
6: $\hat{f}_{\mathrm{CKDE},\mathbf{z}}\left(x_i \mid \mathbf{x}_{\mathrm{Pa}_{\mathbf{Y}}(i)}\right) \leftarrow$ CKDE constructed with $\hat{f}_{\mathrm{KDE},\mathbf{z}}\left(x_i, \mathbf{x}_{\mathrm{Pa}_{\mathbf{Y}}(i)}\right)$ and $\hat{f}_{\mathrm{KDE},\mathbf{z}}\left(\mathbf{x}_{\mathrm{Pa}_{\mathbf{Y}}(i)}\right)$
7: **end for**
8: **return** $\hat{f}_{\mathrm{HCKDE}}\left(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)}\right)$ created with all the $\hat{f}_{\mathrm{CKDE},\mathbf{z}}\left(x_i \mid \mathbf{x}_{\mathrm{Pa}_{\mathbf{Y}}(i)}\right)$

3.2.2 Structure learning

The structure learning of an HSPBN is based on the learning process described in Atienza et al. (2022). In this work, we will use a greedy hill-climbing algorithm to learn the structure of the HSPBN. The greedy hill-climbing algorithm consists of a score function and a set of operators. The score function can evaluate the goodness of a given graph structure. The set of operators is a set of small, local changes that modify a given structure and returns a new structure. The greedy hill-climbing algorithm successively applies the operator that most improves the score of the structure until a local minimum is found. Thus, the set of operators determines the possible search paths for the best structures.

For most Bayesian network types, the set of operators used in a greedy hill-climbing algorithm is composed of the possible changes in a single arc: an arc addition, an arc removal, or an arc reversal. In an HSPBN, some arc operators (some arc additions and arc reversals) are forbidden, since a discrete variable cannot have a continuous variable as parent. Furthermore, an HSPBN also requires estimating the type of CPD for each node. The type of CPD for the nodes can be added as additional information in the graph, where a node type is assigned to each node. To learn the node types automatically, we include a set of operators that can change the node type of a single node. We denote a node type change for variable X_i as TYPE- CHANGE(i). For example, if the node type of X_i is a CLG node type, the TYPE- CHANGE(i) operator changes the node type of X_i to HCKDE. Conversely, if the node type of X_i is an HCKDE, then the node type is changed to CLG. For a discrete variable, the node type is not mutable, so the operators TYPE- CHANGE(i) are not included for this type of variables. The TYPE- CHANGE(i) operator allows to explore structures with different node types, which in turn enables the detection of the best node type for each node.

The score function is an important component of the greedy hill-climbing algorithm. The score function should return higher values for structures that improve their fit to the data. Usually, the score function is used to evaluate the delta score of each operator, i.e., the increase or reduction in score when an operator is applied. A valid scoring function is the maximum log-likelihood function, $\max \mathcal{L}(\mathcal{G}, \boldsymbol{\theta} : \mathcal{D})$, which is usually calculated as the log-likelihood function with the MLE parameters, $\mathcal{L}(\mathcal{G}, \hat{\boldsymbol{\theta}}^{\text{MLE}} : \mathcal{D})$. However, the use of the maximum log-likelihood score is not recommended because an arc addition never reduces the score (Koller and Friedman 2009), which usually leads to completely connected networks. A common alternative is the use of the BIC (Bayesian Information Criterion) score, that includes a penalization to the complexity of the network in the maximum log-likelihood function:

$$\mathcal{S}_{\text{BIC}}(\mathcal{D}, \mathcal{G}) = \mathcal{L}(\mathcal{G}, \hat{\boldsymbol{\theta}}^{\text{MLE}} : \mathcal{D}) - \frac{\log N}{2} \text{Dim}(\mathcal{G}), \tag{17}$$

where $\text{Dim}(\mathcal{G})$ counts the number of free parameters in the Bayesian network. Neither the maximum log-likelihood nor the BIC score functions are valid candidates to learn an HSPBN because the calculation of the log-likelihood function when the network contains KDE models overestimates the goodness of the Bayesian network. This is because a KDE model stores the data $\mathcal{D}$ to fit the KDE models. Then, when the log-

likelihood of an instance is being calculated, it is guaranteed that this same instance is also a training instance in Eq. (1). This causes an overestimation in the goodness of the model because there would be N $K_{\mathbf{H}}(\mathbf{0})$ terms in the log-likelihood calculation (the maximum of $K_{\mathbf{H}}$ is assumed to be at $K_{\mathbf{H}}(\mathbf{0})$).

From these observations, it is reasonable that the data used to fit the CPDs must be different than the data to evaluate the goodness of the model. Applying a k-fold cross-validation to the data can split the data to achieve this objective, while ensuring all the data are used. A k-fold cross-validation splits the data into k disjoint subsets. We will denote $\mathcal{I}^i$ the instance indices for the i-th fold, so $\mathcal{D}_{\downarrow \mathcal{I}^i}$ is the i-th fold data. The indices not in the i-th fold will be denoted $\mathcal{I}^{-i} = \bigcup_{m=1:m\neq i}^{k} \mathcal{I}^m$. The set of indices is represented as $\mathcal{I} = \{\mathcal{I}^1, \ldots, \mathcal{I}^k\}$. A k-fold cross-validated likelihood score is then:

$$\mathcal{S}^k_{\mathrm{CV}}(\mathcal{D}, \mathcal{G}) = \sum_{m=1}^{k} \mathcal{L}(\mathcal{G}, \boldsymbol{\theta}^{\mathcal{I}^{-m}} : \mathcal{D}_{\downarrow \mathcal{I}^m}), \tag{18}$$

where $\boldsymbol{\theta}^{\mathcal{I}^{-m}}$ are the parameters trained with the data $\mathcal{D}_{\downarrow \mathcal{I}^{-m}}$ using the parameter learning procedure described in Sect. 3.2.1. The $\mathcal{S}^k_{\mathrm{CV}}$ is a valid score to train HSPBNs because the log-likelihood is calculated over data that were not seen while estimating the parameters. In addition, the score $\mathcal{S}^k_{\mathrm{CV}}$ can decrease by the addition of some arcs. This event takes place when the arc addition does not improve the model performance in unseen data, thus avoiding the overfitting. Therefore, the score can control the complexity of the model automatically. $\mathcal{S}^k_{\mathrm{CV}}$ can be understood as an estimator of the expected log-likelihood of the model, i.e., the expected log-likelihood for the new and unseen data.

During the progress of the greedy hill-climbing algorithm, several candidate structures need to be evaluated. For this reason, to accomplish a fast execution of the algorithm we need to reduce as much as possible the number of evaluations of the score function. The score function may have a property called decomposability that notably reduces the number of needed score evaluations. A decomposable score can be calculated as the sum of local score terms for each node:

$$\mathcal{S}_{\mathrm{decomposable}} = \sum_{i=1}^{n} \mathcal{S}_{\mathrm{local}}(X_i \mid \mathbf{X}_{\mathrm{Pa}(i)}). \tag{19}$$

A decomposable score reduces the number of evaluations because the operators change the local score of a node (arc addition, arc removal, node type change) or two nodes (arc reversal). This ensures that after applying an operator, it is only necessary to update the delta score for the operators of the affected nodes. The delta score of the other operators can be cached to avoid calling the score function. The maximum log-likelihood and the BIC scores are known to be decomposable [see Eq. (19), which sums a term for each node].

The $\mathcal{S}^k_{\mathrm{CV}}$ score is also decomposable. This is obvious because $\mathcal{S}^k_{\mathrm{CV}}$ is the sum of k log-likelihood functions, which are in turn also decomposable. However, to keep the delta scores cached it is important to fix a set of k-fold cross-validation indices $\mathcal{I}$ in

advance and never change it. Otherwise, the data used to estimate the parameters of each fold, $\boldsymbol{\theta}^{\mathcal{I}^{-m}}$, and the data fold, $\mathcal{D}_{\downarrow\mathcal{I}^m}$, would change. Therefore, the score $\mathcal{S}^k_{\text{CV}}$ with two different sets of indices $\mathcal{I}$ and $\mathcal{I}'$ will be slightly different, thus breaking the capability to take advantage of the decomposability of the score.

Using the $\mathcal{S}^k_{\text{CV}}$ score with a fixed set of indices $\mathcal{I}$ makes it decomposable and allows to cache the scores. However, fixing a set of indices $\mathcal{I}$ introduces a bias on the score, because the final structure will be optimized only for the set of indices $\mathcal{I}$, but it may be suboptimal for other sets of indices $\mathcal{I}'$. This is especially relevant when evaluating operators with small positive delta scores. This is because, for the chosen $\mathcal{I}$, the delta score can be a low positive number, while for other set of indices $\mathcal{I}'$ the delta score can be a negative number. This effect can be understood as overfitting the set of indices $\mathcal{I}$. To solve this problem, we randomly split the data into two disjoint subsets: a training dataset and a validation dataset, $\mathcal{D} = \mathcal{D}^{\text{train}} \cup \mathcal{D}^{\text{val}}$. The operators will be evaluated using the $\mathcal{S}^k_{\text{CV}}$ score with the training dataset $\mathcal{D}^{\text{train}}$. The validation dataset, $\mathcal{D}^{\text{val}}$, is used to control if the score $\mathcal{S}^k_{\text{CV}}$ is overfitting $\mathcal{I}$ using a validated log-likelihood:

$$\mathcal{S}_{\text{validation}}(\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}}, \mathcal{G}) = \mathcal{L}(\mathcal{G}, \boldsymbol{\theta}^{\text{train}} : \mathcal{D}^{\text{val}}). \tag{20}$$

where $\boldsymbol{\theta}^{\text{train}}$ are the parameters learned using the dataset $\mathcal{D}^{\text{train}}$.

If the structure overfits $\mathcal{I}$ with a given operator, the $\mathcal{S}^k_{\text{CV}}$ delta score will be positive, but the $\mathcal{S}_{\text{validation}}$ delta score will be negative because the operator deteriorates the generalization ability of the model. However, if the delta of both scores is positive, the operator is clearly useful to increase the performance of the model. The split of the data into train and validation data is a technique called early-stopping criterion (Prechelt 2012) or engineering criterion (Heckerman and Chickering 1997), and has also been used to train other types of machine learning models.

The greedy hill-climbing algorithm selects and applies the best operator found on each iteration until no operator provides an improvement. The greedy hill-climbing usually ends up finding a local optimum. By definition, the local optimum structure does not contain in the neighborhood of the search space a better structure, i.e., there is no operator that improves the current structure. In our greedy hill-climbing implementation, we try to relax this restriction. Therefore, we allow the structure not to improve the score for a maximum of λ iterations. If λ iterations have been executed without improving the score, the best structure found so far is returned. λ is a parameter of the algorithm called patience. If λ is greater than 0, we allow exploration beyond the local neighborhood trying to escape the local optimum. To improve the exploration, we implemented a tabu search (Glover and Laguna 1993) that is executed while trying to escape from local optimum. The tabu search forbids applying operators that reverse recently applied ones, e.g., an arc addition is not allowed if the same arc removal was executed recently. If the algorithm manages to escape the local optimum, the tabu search is disabled. Thus, when exploration of the search space is not needed, the search is dedicated to finding the optimum as fast as possible.

Algorithm 2 details the implementation of our greedy hill-climbing strategy combined with tabu search. The algorithm starts by doing some basic initializations (lines 1–4), splitting the data into training and validation sets (line 5), and assigning a set

of k-fold cross-validation indices $\mathcal{I}$ (lines 6–7). Then, the algorithm starts optimizing the structure in the main loop (lines 8–28). First, the algorithm finds the best available operator (lines 10–18). We include the requirement that the delta score $\mathcal{S}^k_{\text{CV}}(\mathcal{D}^{\text{train}}, \mathcal{G}_{\text{candidate}}) - \mathcal{S}^k_{\text{CV}}(\mathcal{D}^{\text{train}}, \mathcal{G})$ must be greater than a threshold $\epsilon \geq 0$. In this work, we always use $\epsilon = 0$ because it guarantees that the selected operator improves the $\mathcal{S}^k_{\text{CV}}$ score. Then, the algorithm checks that the best operator improves the validation score $\mathcal{S}_{\text{validation}}$ (line 19). If it improves the validation score (lines 20–22), the best structure so far has been found, the tabu search is disabled and the patience counter, p, is reset to 0. Otherwise, the tabu search set is updated and the patience counter is increased (lines 24–25). At the end of each iteration, the cached scores are updated taking into account the applied operator o_{new}, so only the delta score of the affected nodes is updated.

Algorithm 2 Greedy hill-climbing for HSPBNs

Require: Training data $\mathcal{D}$, starting structure $\mathcal{G}_0$, the set of operators $\mathcal{O}$, patience $\lambda \in \mathbb{N}$, the number of folds $k \geq 2$ (and $k < N$), minimum delta $\epsilon \geq 0$

1: $\mathcal{G}_{\text{best}} \leftarrow \mathcal{G}_0$
2: $\mathcal{G}_{\text{new}} \leftarrow \mathcal{G}_0$
3: $p \leftarrow 0$
4: Tabu $\leftarrow \emptyset$
5: $\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}} \leftarrow \text{Split}(\mathcal{D})$
6: $\mathcal{I} \leftarrow$ Generate k sets of disjoint indices for $\mathcal{D}^{\text{train}}$
7: Assign $\mathcal{I}$ to $\mathcal{S}^k_{\text{CV}}$
8: **do**
9: $\quad \mathcal{G} \leftarrow \mathcal{G}_{\text{new}}$
10: $\quad$ **for** o in $\mathcal{O}$ **do**
11: $\quad\quad$ **if** o does not reverse $o' \in$ Tabu **then**
12: $\quad\quad\quad \mathcal{G}_{\text{candidate}} \leftarrow o(\mathcal{G})$
13: $\quad\quad\quad$ **if** $\mathcal{S}^k_{\text{CV}}(\mathcal{D}^{\text{train}}, \mathcal{G}_{\text{candidate}}) > \mathcal{S}^k_{\text{CV}}(\mathcal{D}^{\text{train}}, \mathcal{G}_{\text{new}})$ **and** $\mathcal{S}^k_{\text{CV}}(\mathcal{D}^{\text{train}}, \mathcal{G}_{\text{candidate}}) - \mathcal{S}^k_{\text{CV}}(\mathcal{D}^{\text{train}}, \mathcal{G}) > \epsilon$ **then**
14: $\quad\quad\quad\quad o_{\text{new}} \leftarrow o$
15: $\quad\quad\quad\quad \mathcal{G}_{\text{new}} \leftarrow \mathcal{G}_{\text{candidate}}$
16: $\quad\quad\quad$ **end if**
17: $\quad\quad$ **end if**
18: $\quad$ **end for**
19: $\quad$ **if** $\mathcal{S}_{\text{validation}}(\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}}, \mathcal{G}_{\text{new}}) > \mathcal{S}_{\text{validation}}(\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}}, \mathcal{G}_{\text{best}})$ **then**
20: $\quad\quad \mathcal{G}_{\text{best}} \leftarrow \mathcal{G}_{\text{new}}$
21: $\quad\quad$ Tabu $\leftarrow \emptyset$
22: $\quad\quad p \leftarrow 0$
23: $\quad$ **else**
24: $\quad\quad$ Tabu $\leftarrow$ Tabu $\cup\, o_{\text{new}}$
25: $\quad\quad p \leftarrow p + 1$
26: $\quad$ **end if**
27: $\quad$ UPDATE_SCORE_CACHE($\mathcal{G}, o_{\text{new}}$)
28: **while** $p < \lambda$
29: **return** $\mathcal{G}_{\text{best}}$

3.3 Sampling from nonparametric conditional probability distribution

Bayesian networks are known to be generative models, which model a probability distribution $P(\mathbf{x})$. One of the advantages of a generative model is the ability to sample new data from the model. In the case of Bayesian networks, there are also some inference algorithms that are based on sampling, such as likelihood weighting and Markov chain Monte Carlo techniques (Darwiche 2009), e.g., Gibbs sampling. In this section, we detail how new data can be sampled from the CKDE and HCKDE CPDs.

Let X_i be an HCKDE node, and $\hat{f}_{\text{HCKDE}}(x_i \mid \mathbf{x}_{\text{Pa}(i)})$ be its conditional distribution given its parents. By definition, Eq. (10), its conditional distribution is equal to $\hat{f}_{\text{CKDE},\mathbf{x}_{\text{Pa}_\mathbf{Z}(i)}}(x_i \mid \mathbf{x}_{\text{Pa}_\mathbf{Y}(i)})$. Thus, sampling from an HCKDE requires sampling from the corresponding CKDE CPD.

Assuming K is the Gaussian multivariate kernel, the CKDE CPD can be expressed in the following way:

$$\begin{aligned}\hat{f}_{\text{CKDE}}(x_i \mid \mathbf{y}) &= \frac{\sum_{j=1}^{N} \mathcal{N}\left(\begin{bmatrix} x_i \\ \mathbf{y} \end{bmatrix}; \begin{bmatrix} x_i^j \\ \mathbf{y}^j \end{bmatrix}, \mathbf{H}\right)}{\sum_{j=1}^{N} \mathcal{N}\left(\mathbf{y}; \mathbf{y}^j, \mathbf{H}_{-i}\right)} \\ &= \frac{\sum_{j=1}^{N} \mathcal{N}\left(\mathbf{y}; \mathbf{y}^j; \mathbf{H}_{-i}\right) \mathcal{N}(x_i; \mu_i^{\text{cond, j}}, h_i^{\text{cond}})}{\sum_{j=1}^{N} \mathcal{N}\left(\mathbf{y}; \mathbf{y}^j, \mathbf{H}_{-i}\right)} \\ &= \sum_{j=1}^{N} w_j \mathcal{N}\left(x_i; \mu_i^{\text{cond, j}}, h_i^{\text{cond}}\right),\end{aligned} \tag{21}$$

with

$$w_j = \frac{\mathcal{N}\left(\mathbf{y}; \mathbf{y}^j; \mathbf{H}_{-i}\right)}{\sum_{k=1}^{N} \mathcal{N}\left(\mathbf{y}; \mathbf{y}^k, \mathbf{H}_{-i}\right)}, \tag{22}$$

where $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the Gaussian probability density function with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, evaluated at $\mathbf{x}$. The second equality holds because the joint distribution $f(x_i, \mathbf{y})$ is expressed as $f(x_i \mid \mathbf{y}) f(\mathbf{y})$. For the Gaussian distribution, the marginal, $f(\mathbf{y})$, and the conditional, $f(x_i \mid \mathbf{y})$, distributions are easy to find. The parameters $\mu_i^{\text{cond, j}}$ and $h_i^{\text{cond, j}}$ are:

$$\begin{aligned}\mu_i^{\text{cond, j}} &= x_i^j + \mathbf{b}^T \mathbf{C}^{-1} (\mathbf{y} - \mathbf{y}^j) \\ h_i^{\text{cond}} &= a - \mathbf{b}^T \mathbf{C}^{-1} \mathbf{b},\end{aligned} \tag{23}$$

where a, $\mathbf{b}$ and $\mathbf{C}$ are the bandwidth matrix blocks defined in Eq. (15).

Sampling from a CKDE is easier from the expression in Eq. (21). Algorithm 3 details the sampling procedure from a CKDE. First, the weights $\mathbf{w} = (w_1, \ldots, w_N)$ are calculated (line 1). Then, a sample is generated from a categorical distribution with the weights $\mathbf{w}$ as parameters (line 2). Finally, the $\mu_i^{\text{cond, j}}$ and h_i^{cond} parameters are calculated (line 3), and a new sample it is obtained sampling a Gaussian with mean $\mu_i^{\text{cond, j}}$ and variance h_i^{cond} (line 4).

Algorithm 3 Sampling new data from a CKDE CPD

Require: CKDE CPD $\hat{f}(x_i \mid \mathbf{y})$, evidence instantiation $\mathbf{y}$
1: $\mathbf{w} \leftarrow$ Compute vector of w_j (Eq. (22))
2: $j \sim$ Categorical($\mathbf{w}$) // Sample from a categorical distribution with parameters $\mathbf{w}$
3: $\mu_i^{\text{cond, j}}, h_i^{\text{cond}} \leftarrow$ Compute conditional mean and variance (Eq. (23))
4: $x \sim \mathcal{N}(\mu_i^{\text{cond, j}}, h_i^{\text{cond}})$ // Sample Gaussian with mean $\mu_i^{\text{cond, j}}$ and variance h_i^{cond}
5: **return** x

3.4 Relation with adaptive KDE

In the HCKDE CPD [Eq. (10)], the bandwidth matrix depends on the discrete configuration $\mathbf{z}$. This bandwidth matrix expression is similar to the bandwidth matrices used in adaptive KDEs [Eqs. (4) and (5)]. Furthermore, we can consider an HCKDE CPD to be a type of adaptive KDE. The main difference with the adaptive KDEs defined in the state of the art is that in an HCKDE the bandwidth matrix depends on discrete variables instead of continuous variables. In addition, since the HCKDE CPD is part of a Bayesian network, it can automatically learn which discrete variables are most useful for splitting the data and apply a different amount of smoothing at each split. When the Bayesian network does not select any discrete variable as a parent of the HCKDE, the resulting CPD is a CKDE, which is not adaptive. Thus, the HSPBN learning algorithm automatically chooses between using standard KDEs or adaptive KDEs.

3.5 Asymptotic analysis

In this section, we analyze the asymptotic time and space complexity of the proposed approach. Let $J = N - N/k$ be the number of train instances in the cross-validation performed by $\mathcal{S}_{\text{CV}}^k$. Atienza et al. (2022) show that the asymptotic time complexity of $\mathcal{S}_{\text{CV}}^k$ for a variable X_i represented with a CKDE CPD is $O(NJ|\text{Pa}(i)|^2)$, or more loosely expressed $O(N^2|\text{Pa}(i)|^2)$. Furthermore, it is shown that the asymptotic time complexity of $\mathcal{S}_{\text{CV}}^k$ for an LG CPD is $O(kJ|\text{Pa}(i)|^2)$. Therefore, the time complexity of the nonparametric models scales much faster with respect to the sample size. Note, however, that a CLG or a HCKDE CPD is composed of an LG or CKDE CPD for each discrete configuration of its parents, so the data is partitioned. This makes the asymptotic time complexity of a CLG or HCKDE CPD always less than or equal to the asymptotic time complexity of an LG or CKDE CPD with the same set of continuous parents, respectively. Since the CKDE time complexity scales worse than that of the LG, the relative time complexity gains can be much larger by using discrete parents in the HCKDE case.

In terms of spatial complexity, the LG CPD requires storing the $\beta_\bullet$ coefficients and the variance σ_i^2. This has a spatial complexity of $O(|\text{Pa}(i)| + 2)$. The CKDE CPD requires storing the data and a bandwidth matrix. Therefore, the spatial complexity of a CKDE CPD is $O(N(|\text{Pa}(i)| + 1) + (|\text{Pa}(i)| + 1)^2)$. This shows that, as expected, a parametric model is much more concise than a nonparametric one. Let $Z = |\mathbf{\Omega}_{\text{Pa}_\mathbf{Z}(i)}|$ be the number of discrete configurations for the discrete variables of a CLG or HCKDE

CPD. Then, the spatial complexity of CLG is equal to $O(Z(|\text{Pa}_{\mathbf{Y}}(i)| + 2))$ since it is necessary to store the parameters of each LG. In the case of HCKDE, the data is partitioned into multiple CKDE models, and a different bandwidth matrix is needed for each CKDE. Therefore, the spatial complexity of a HCDKE CPD is equal to $O(N(|\text{Pa}_{\mathbf{Y}}(i)| + 1) + Z(|\text{Pa}_{\mathbf{Y}}(i)| + 1)^2)$. Most of the time, since $N \gg Z(|\text{Pa}_{\mathbf{Y}}(i)| + 1)$ the spatial complexity is dominated by the $N(|\text{Pa}_{\mathbf{Y}}(i)| + 1)$ term, so there is no major difference between the spatial complexity of CKDE and HCKDE CPDs. Note, however, that Z increases exponentially with the number of discrete parents $|\text{Pa}_{\mathbf{Z}}(i)|$, so this may change for HCKDE CPDs with large number of discrete parents.

4 Experiments

In this section, we experimentally check the capabilities of HSPBNs to obtain a good fit to the data. We will perform two types of experiments: experiments with synthetic data, where we know the data distribution and the expected HSPBN model, and real-world data from the UCI repository, where the underlying model that generated the data is unknown.

The experiments were conducted using our PyBNesian library https://github.com/davenza/PyBNesian.git, and the source code is available at https://github.com/davenza/HSPBN-Experiments.git

4.1 Synthetic data

In this section, we compare the results of CLGBN models and HSPBN models learned with the procedure described in Algorithm 2. To test the ability of the HSPBNs to capture conditional linear relationships (with a CLG CPD) and also conditional nonlinear relationships (with an HCKDE CPD), we will generate synthetic data. The synthetic data are sampled from synthetic Bayesian networks, whose structure and parameters are randomly created with the following procedure:

1. Generate random structure.
   (a) Add an arc between each pair of variables with probability 0.25.
   (b) Select a node type (CLG or HCKDE) for each continuous node with equal probability.
2. Generate random parameters. Sample a new CPD for each variable X_i.
   (a) For discrete nodes. $P(x_i \mid \mathbf{x}_{\text{Pa}(i)}) \sim \text{Dir}(3 \cdot \mathbf{1}_{|\Omega_i|})$, where $\mathbf{1}_p$ is a vector of ones with p dimensions and $\text{Dir}(\cdot)$ is the Dirichlet distribution.
   (b) For CLG nodes. Sample the LG CPD parameters for each discrete configuration of their discrete parents.
   (c) For HCKDE nodes. Assign a mixture of LG CPDs to each discrete configuration of their discrete parents with the following procedure:
      i. Choose the number of components of the mixture k, i.e., the number of LG CPDs, so $k \sim \text{Categorical}([2, 3, 4]; [0.4, 0.3, 0.3])$
      ii. Sample the prior probability of the mixture from $\text{Dir}(3 \cdot \mathbf{1}_k)$.

iii. Sample the LG CPD parameters for each component of the mixture.

Note that we assign mixtures of LG CPDs to each HCKDE node instead of a standard HCKDE CPD as described in Definition 4. The purpose of using mixtures of LG CPDs is to sample conditional non-Gaussian data. Therefore, we expect our learning process to detect that data sampled from mixtures of LG CPDs are best represented using HCKDEs. Furthermore, this decision is motivated because the construction of HCKDE CPDs requires the generation of training data to be assigned to the HCKDEs, which is the problem we are solving in the first place.

The above procedure also requires computing the parameters of LG CPDs. The parameters are sampled with the following approach:

- The intercept $\beta_0 \sim \mathcal{N}(0, 4)$.
- Each coefficient β_j $(j > 0) \sim$ Categorical$([1, -1]; [0.5, 0.5]) \cdot$ Unif$([1, 5])$, where Unif$([a, b])$ is the continuous uniform distribution with support in the range $[a, b]$. This ensures that the coefficients are positive or negative with equal probability. Moreover, the absolute value is always greater than 1, preventing conditional independences in the parameters (if $\beta_j = 0$).
- The variance $\sigma^2 \sim 0.1 + \chi^2_1$, where χ^2_l is the chi-squared distribution with l degrees of freedom. The sum of 0.1 ensures that the variance is sufficiently greater than 0, so there are no precision errors.

To compare the performance of the models, we generated 100 different synthetic Bayesian networks with 4 discrete variables and 4 continuous variables. The cardinality of the discrete variables is fixed, so there are two variables with cardinality 2, one variable with cardinality 3 and one variable with cardinality 4. For each Bayesian network, we sampled training datasets with different numbers of instances: 200, 2000, and 10,000. Then, for each dataset, we learned the following models: CLGBNs with the BIC (CLGBN-BIC) and the cross-validated score (CLGBN-VL) in Eq. (18), HSPBNs where the initial node type for all nodes is parametric (HSPBN-CLG), i.e., the CPDs for the continuous nodes of the starting model are CLGs, and HSPBNs where the initial node type for all nodes is nonparametric (HSPBN-HCKDE), i.e., the CPDs for the continuous nodes of the starting model are HCKDEs. In all cases, the starting graph ($\mathcal{G}_0$ in Algorithm 2) had no arcs. We used $\lambda = 0$ and $\lambda = 15$ for all the learned models. During the structure learning, the scores in Eqs. (18) and (20) of the HCKDE CPDs are calculated estimating the bandwidth matrix using the normal reference rule because is the fastest criterion available. Then, when the final structure is learned, we tried the normal reference rule, the UCV criterion, and the plug-in estimation method to estimate the bandwidth matrices.

To evaluate the performance of the models, we sampled another test dataset of 1000 instances from each synthetic Bayesian network that was not previously seen by the learned models, i.e., a new dataset different to $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{val}}$. This test dataset can estimate the expected performance of the models on the new and unseen data by calculating the log-likelihood of the test dataset. In addition, we also compare the structural accuracy of the graphs (how good the learned graph compares to the synthetic Bayesian network graph). The structural accuracy can be measured using the Hamming distance (HMD), which measures the number of arcs (ignoring the arc directions) present in one graph but not in the other. The HMD scores equally for

arcs that have the same or different orientations in both graphs. An alternative is the structural Hamming distance (SHD) (Tsamardinos et al. 2006), which measures the number of arcs additions, removals, or reversals required to transform a graph into the other. In addition, to evaluate the ability of the learning algorithm to find the best node type for each node, we include a type Hamming distance (THMD). The THMD measures the number of nodes with different node types in both graphs. For all the structural accuracy criteria, the lower the value, the better. Moreover, we measured the runtime needed to learn each model.

The results of this comparison are shown in Table 1. The log-likelihood, HMD, SHD, and THMD values are the mean value of the 100 datasets sampled from each synthetic Bayesian network. Table 1 shows the results when the bandwidth matrices are estimated using the normal reference rule. In general, the table shows that HSPBNs have better log-likelihood results as well as structural accuracy (except for the training datasets of 200 instances) than CLGBNs. As expected, the log-likelihood improves with the increase in the training dataset instances. This is most noticeable in the case of the HSPBNs. For small datasets, the difference between CLGBNs and HSPBNs is small, but for larger datasets, the difference is so significant that an HSPBN trained with 2000 instances can be as good or even better than a CLGBN trained with 10,000 instances. In the case of the THMD, we can see that the accuracy improves significantly when the number of instances increases. With respect to the runtimes, the CLGBNs exhibit much better result. This is caused by the greater complexity of the score function for HSPBNs, since the evaluation of KDE models is needed. Also, it seems that the runtimes scales much faster for HSPBNs.

We also tested the UCV and the plug-in estimation methods. We found experimentally that these two estimation methods are especially sensitive to the existence of outliers in the data. This caused bad results for the small datasets (especially for 200 instances) because some instances that are sampled with low probabilities can be seen as outliers in small datasets. The small datasets only contain a few instances in the low-probability space zones. Although we do not include these results in the manuscript, we highlight that the plug-in estimation method returned better results than UCV. Thus, for the large dataset of 10,000 instances, the plug-in estimation method returned a log-likelihood of -9622.92, -9624.18, -9626.71 and -9624.95 for the HSPBN-CLG with $\lambda = 0$, the HSPBN-CLG with $\lambda = 15$, the HSPBN-HCKDE with $\lambda = 0$ and the HSPBN-HCKDE with $\lambda = 15$, respectively. These results are even better than the results obtained by the normal reference rule in Table 1. In addition, the plug-in estimation method for the models trained with 2000 instances returned a log-likelihood of -9790.38, -9786.88, -9783.01 and -9764.87 for the HSPBN-CLG with $\lambda = 0$, the HSPBN-CLG with $\lambda = 15$, the HSPBN-HCKDE with $\lambda = 0$ and the HSPBN-HCKDE with $\lambda = 15$, respectively. These values are similar to the reported results with the normal reference rule with 10,000 instances. With these results, we can conclude that the UCV and plug-in estimation methods are only useful if there is a large amount of data.

Table 1 Results of learning from synthetic data sampled from synthetic Bayesian networks

Instances	Model	Log-likelihood	HMD	SHD	THMD	Time (s)
	Max possible value		32	32	4	–
	Ground truth	−9479.79	0	0	0	–
200	CLGBN-BIC $\lambda = 0$	−10957.95	**2.24**	**2.86**	–	2.67×10^{-3}
	CLGBN-BIC $\lambda = 15$	−10957.95	**2.24**	**2.86**	–	$\mathbf{2.43 \times 10^{-3}}$
	CLGBN-VL $\lambda = 0$	−11488.09	3.14	3.55	–	7.59×10^{-2}
	CLGBN-VL $\lambda = 15$	−11677.50	2.94	3.44	–	1.44×10^{-1}
	HSPBN-CLG $\lambda = 0$	−10994.30	3.15	3.61	1.19	1.14
	HSPBN-CLG $\lambda = 15$	−11626.07	2.63	3.22	**0.87**	1.44
	HSPBN-HCKDE $\lambda = 0$	−10787.45	2.63	3.10	1.45	2.35
	HSPBN-HCKDE $\lambda = 15$	**−10743.28**	2.43	2.97	1.15	2.57
2000	CLGBN-BIC $\lambda = 0$	−10428.11	0.87	1.59	–	5.20×10^{-3}
	CLGBN-BIC $\lambda = 15$	−10428.11	0.87	1.59	–	$\mathbf{4.62 \times 10^{-3}}$
	CLGBN-VL $\lambda = 0$	−10458.61	1.35	2.09	–	1.77×10^{-1}
	CLGBN-VL $\lambda = 15$	−10419.15	1.27	1.99	–	2.22×10^{-1}
	HSPBN-CLG $\lambda = 0$	−9920.09	1.05	1.79	0.3	7.10
	HSPBN-CLG $\lambda = 15$	−9914.59	1.03	1.77	**0.28**	7.19
	HSPBN-HCKDE $\lambda = 0$	−9918.34	0.72	1.35	0.71	10.01
	HSPBN-HCKDE $\lambda = 15$	**−9901.44**	**0.62**	**1.27**	0.46	10.46

Table 1 continued

Instances	Model	Log-likelihood	HMD	SHD	THMD	Time (s)
10000	CLGBN-BIC $\lambda = 0$	−10402.88	0.95	1.64	–	1.02×10^{-2}
	CLGBN-BIC $\lambda = 15$	−10402.88	0.95	1.64	–	$\mathbf{7.99 \times 10^{-3}}$
	CLGBN-VL $\lambda = 0$	−10402.20	1.07	1.94	–	2.42×10^{-1}
	CLGBN-VL $\lambda = 15$	−10402.16	1.07	1.87	–	3.02×10^{-1}
	HSPBN-CLG $\lambda = 0$	−9786.49	0.94	1.77	**0.19**	20.00
	HSPBN-CLG $\lambda = 15$	−9785.84	0.94	1.75	**0.19**	20.71
	HSPBN-HCKDE $\lambda = 0$	−9783.78	**0.67**	**1.56**	0.43	28.97
	HSPBN-HCKDE $\lambda = 15$	**−9783.30**	0.75	1.64	0.29	29.50

HMD stands for the Hamming distance, SHD denotes the structural Hamming distance, and THMD corresponds to the node type Hamming distance computed between the learned model and the ground truth model. The runtime of the learning process (Time) is reported in seconds.
The best result for each metric is highlighted boldface for each dataset size

Table 2 Datasets from the UCI repository

Dataset	N	n	c	d
Abalone	4177	9	8	1
Adult	45,222	14	5	9
Australian Statlog	690	15	6	9
Cover Type	11,340	55	10	45
Credit Approval	653	16	6	10
German Statlog	1000	21	7	14
KDD Cup 1999	10,000	29	23	6
Liver Disorders	341	7	6	1
Thyroid-Hypothyroid	2000	25	6	19
Thyroid-Sick	1947	30	7	23

4.2 UCI Data

In this section, we test the ability of the HSPBNs to fit to real-world data analyzing datasets from the UCI repository (Dua and Graff 2017). The characteristics of all the UCI datasets are shown in Table 2. For the KDD Cup 1999, we selected a subset of the data because it is a very large dataset (almost 5 million instances) and the experiments would have been too time-consuming. Thus, we selected the first 10,000 instances of the dataset. For all the datasets, we removed the constant columns (as they are not interesting to be analyzed with a Bayesian network). The number of columns indicated in Table 2 is the result obtained after this preprocessing step.

For the UCI datasets, we do not know the data distribution or the model that generated the data, as these datasets are usually extracted from the real world. For this reason, we cannot estimate the log-likelihood using a new test dataset and we cannot calculate the structural accuracy. To estimate the goodness of our model, we perform a tenfold cross-validation. On each fold, we learn a model on the training subset and we estimate the goodness of the model calculating the log-likelihood of the test subset on the learned model. We tested the same configurations of models as in the synthetic experiments with $\lambda = 0$, $\lambda = 5$ and $\lambda = 15$. In this case, we found that the UCV and the plug-in estimation methods are again sensitive to outliers. For this reason, in some datasets they returned very poor results, while in other datasets they offered competitive results. Therefore, we omit the presentation of results with these bandwidth selection techniques and we focus our attention on the results using the normal reference rule.

We calculated the mean log-likelihood of the test subset in the tenfold cross-validation. Then, to find statistically significant differences between all the algorithms, we performed a Friedman test with $\alpha = 0.05$ and a Bergmann–Hommel post hoc procedure to detect the pairwise significant differences (García and Herrera 2008). The results are shown in Fig. 2 using a critical difference diagram (Demšar 2006). The critical difference diagram shows the mean rank of each model over all the datasets. The black horizontal bars indicate the groups of models whose mean rank difference is not statistically significant. Therefore, we can conclude that there is no statisti-

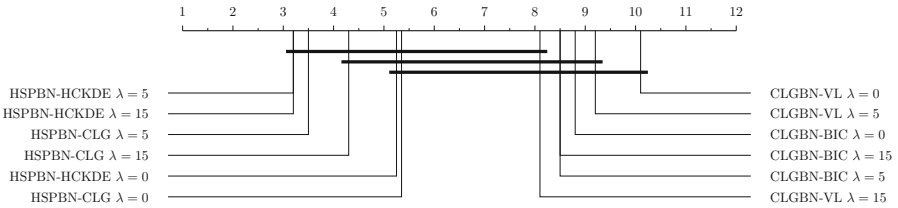

Fig. 2 Critical difference diagram for the mean rank of each algorithm in the UCI datasets

cally significant difference between all the HSPBNs configurations and CLGBN-VL $\lambda = 15$. However, the statistical test shows that there is a statistically significant mean rank difference between HSPBN-HCKDE $\lambda = 5$, HSPBN-HCKDE $\lambda = 15$, HSPBN-CLG $\lambda = 5$ and the remaining CLGBN models. Furthermore, we found that a patience greater than 0 is positive because the models trained with $\lambda = 5$ and $\lambda = 15$ obtain better results than the $\lambda = 0$ ones. This is reasonable because the algorithm takes more time to optimize the model trying to escape from local optima. These results demonstrate the usefulness and applicability of the HSPBN models to real-world data. Therefore, this result is not surprising because real-world data may contain non-Gaussian and nonlinear relationships between the variables. HSPBNs models can adapt better to this kind of relationships, obtaining a better fit to the data.

It is noticeable that HSPBN-CLG $\lambda = 5$ obtains a better ranking than HSPBN-CLG $\lambda = 15$. This may seem unreasonable, so we performed a careful manual review on the differences between these models. We found that this effect is caused by a very low log-likelihood for a small set of test instances. These test instances are given a low log-likelihood in the continuous variables because they have rare discrete configurations for their discrete parents, so their respective LG or CKDE model is learned with just a few training instances (we found examples with less than 5 training instances) and do not have a good fit to the data. This effect does not happen so often for $\lambda = 5$ because the learning process ends earlier, so the conParametersKDEtinuous variables have fewer discrete parents and their respective LGs and CKDEs are trained with a significant training set. In this sense, it would be interesting to research new regularization terms that alleviate this phenomenon.

5 Conclusion

This paper presented a new class of Bayesian networks called hybrid semiparametric Bayesian networks. To the best of our knowledge, this is the first type of semiparametric Bayesian network with the ability to model hybrid data containing both discrete and continuous data. In addition, for the continuous data the type of relationships between variables are explicitly represented in the graph. If there is a conditional linear Gaussian relationship, it can be defined using a parametric model (a CLG CPD). If the conditional distribution is non-Gaussian or there are nonlinear relationships, usually this can be better represented using a nonparametric model (an HCKDE CPD).

We described a learning procedure that estimates the parameters and the structure of HSPBN. This procedure is based on other standard techniques in the state of the art. Following this work, we would like to adapt other standard types of algorithms to learn HSPBNs in the future, such as constraint-based learning (e.g., the PC algorithm, Spirtes et al. 2001). In addition, we introduced a procedure to sample new data from an HSPBN and showed its relation with the adaptive KDE models.

The experimental work showed that the HSPBN model can improve the results of the CLGBN models. This is not surprising because the class of CLGBNs is included in the class of HSPBNs.

In the future, we would like to work on semiparametric Bayesian network classifiers. Also, other types of CPDs can be incorporated. Note that learning an HSPBN only requires to be able to compute a cross-validated log-likelihood. Therefore, other parametric CPDs can be used, e.g., other exponential family distributions or mixtures models. In addition, the discrete variables could be modeled using a softmax function as in the augmented CLGBNs (Lerner et al. 2001), so the arcs between continuous variables and discrete variables can be allowed. Moreover, other types of nonparametric models can be investigated to create HSPBNs. An interesting instance is the ratio density estimation (Sugiyama et al. 2012), which may improve the results even further. Furthermore, tools like BUGS (Lunn et al. 2009) and Stan (Carpenter et al. 2017) could be used to perform inference using Markov chain Monte Carlo. Finally, we probably could include a regularization term into the score function to deal with rare discrete configurations of the discrete parents of continuous variables.

Acknowledgements This work has been partially supported by the Ministry of Education, Culture and Sport through the grant FPU16/00921, by the Spanish Ministry of Science and Innovation through the PID2019-109247GB-I00 and RTC2019-006871-7 projects, and by the BBVA Foundation (2019 Call) through the "Score-based nonstationary temporal Bayesian networks. Applications in climate and neuroscience" (BAYES-CLIMA-NEURO) project.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

References

Atienza D, Bielza C, Larrañaga P (2022) Semiparametric Bayesian networks. Inf Sci 584:564–582

Bielza C, Larrañaga P (2014a) Discrete Bayesian network classifiers: a survey. ACM Comput Surv 47(1):Article 5

Bielza C, Larrañaga P (2014b) Bayesian networks in neuroscience: a survey. Front Comput Neurosci 8:Article 131

Boukabour S, Masmoudi A (2020) Semiparametric Bayesian networks for continuous data. Commun Stat Theory Methods 1–23

Bowman AW (1984) An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2):353–360
Breiman L, Meisel W, Purcell E (1977) Variable kernel estimates of multivariate densities. Technometrics 19(2):135–144
Cao R, Cuevas A, González Manteiga W (1994) A comparative study of several smoothing methods in density estimation. Comput Stat Data Anal 17(2):153–176
Carpenter B, Gelman A, Hoffman MD, Lee D, Goodrich B, Betancourt M, Brubaker M, Guo J, Li P, Riddell A (2017) Stan: a probabilistic programming language. J Stat Softw 76(1):1–32
Chacón JE, Duong T (2018) Multivariate kernel smoothing and its applications. Chapman and Hall/CRC, London
Codetta-Raiteri D, Portinale L (2015) Dynamic Bayesian networks for fault detection, identification, and recovery in autonomous spacecraft. IEEE Trans Syst Man Cybern Syst 45(1):13–24
Darwiche A (2009) Modeling and reasoning with Bayesian networks. Cambridge University Press, Cambridge
Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
Dua D, Graff C (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml
Duong T, Hazelton M (2003) Plug-in bandwidth matrices for bivariate kernel density estimation. J Nonparametric Stat 15(1):17–30
Fox J (1997) Applied regression analysis, linear models, and related methods. SAGE Publications, Beverly Hills
Friedman N, Goldszmidt M (1996) Learning Bayesian networks with local structure. In: Proceedings of the twelfth international conference on uncertainty in artificial intelligence. Morgan Kaufmann Publishers, Los Altos, pp 252–262
Friedman N, Nachman I (2000) Gaussian process networks. In: Proceedings of the Sixteenth conference on uncertainty in artificial intelligence. Morgan Kaufmann Publishers, Los Altos, pp 211–219
García S, Herrera F (2008) An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. J Mach Learn Res 9:2677–2694
Glover F, Laguna M (1993) Tabu Search. John Wiley & Sons, London
Gonzalez R, Huang B, Lau E (2015) Process monitoring using kernel density estimation and Bayesian networking with an industrial case study. ISA Trans 58:330–347
Heckerman D, Chickering DM (1997) A comparison of scientific and engineering criteria for Bayesian model selection. In: Proceedings of the sixth international workshop on artificial intelligence and statistics, vol R1, pp 275–282
Heckerman D, Geiger D, Chickering DM (1995) Learning Bayesian networks: the combination of knowledge and statistical data. Mach Learn 20(3):197–243
Hofmann R, Tresp V (1995) Discovering structure in continuous variables using Bayesian networks. In: Proceedings of advances in neural information processing systems, vol 8. MIT Press, Cambridge, pp 500–506
Ickstadt K, Bornkamp B, Grzegorczyk M, Wieczorek J, Rahuman Sheriff M, Grecco HE, Zamir E (2012) Nonparametric Bayesian networks. In: Bayesian statistics, vol 9. Oxford University Press, Oxford, pp 1–40
John GH, Langley P (1995) Estimating continuous distributions in Bayesian classifiers. In: Proceedings of the eleventh conference on uncertainty in artificial intelligence. Morgan Kaufmann Publishers, Los Altos, pp 338–345
Koller D, Friedman N (2009) Probabilistic graphical models: principles and techniques. The MIT Press, Cambridge
Langseth H, Nielsen TD, Rumí R, Salmerón A (2012) Mixtures of truncated basis functions. Int J Approx Reason 53(2):212–227
Lauritzen SL, Wermuth N (1989) Graphical models for associations between variables, some of which are qualitative and some quantitative. Ann Stat 17(1):31–57
Lerner U, Segal E, Koller D (2001) Exact inference in networks with discrete children of continuous parents. In: Proceedings of the seventeenth conference on uncertainty in artificial intelligence. Morgan Kaufmann Publishers, Los Altos, pp 319–328
Loftsgaarden DO, Quesenberry CP (1965) A nonparametric estimate of a multivariate density function. Ann Math Stat 36(3):1049–1051
Luengo-Sanchez S, Larrañaga P, Bielza C (2019) A directional-linear Bayesian network and its application for clustering and simulation of neural somas. IEEE Access 7:69907–69921

Lunn D, Spiegelhalter D, Thomas A, Best N (2009) The BUGS project: evolution, critique and future directions. Stat Med 28(25):3049–3067
Maathuis M, Drton M, Lauritzen S, Wainwright M (2018) Handbook of graphical models, 1st edn. CRC Press, Boca Raton
Mascaro S, Nicholson AE, Korb KB (2014) Anomaly detection in vessel tracks using Bayesian networks. Int J Approx Reason 55(1):84–98
Masmoudi K, Masmoudi A (2019) A new class of continuous Bayesian networks. Int J Approx Reason 109:125–138
Moral S, Rumí R, Salmerón A (2001) Mixtures of truncated exponentials in hybrid Bayesian networks. Symbolic and quantitative approaches to reasoning with uncertainty. Springer, Berlin, pp 156–167
Nadaraya EA (1964) On estimating regression. Theory Probab Appl 9(1):141–142
Pearl J (1988) Probabilistic reasoning in intelligent systems. Morgan Kaufmann Publishers, Los Altos
Pérez A, Larrañaga P, Inza I (2009) Bayesian classifiers based on kernel density estimation: flexible classifiers. Int J Approx Reason 50(2):341–362
Prechelt L (2012) Early stopping—but when? Neural networks: tricks of the trade. Springer, Berlin, pp 53–67
Rudemo M (1982) Empirical choice of histograms and kernel density estimators. Scand J Stat 9(2):65–78
Scott DW (2015) Multivariate density estimation: theory, practice, and visualization, 2nd edn. Wiley, London
Scutari M, Graafland CE, Gutiérrez JM (2019) Who learns better Bayesian network structures: accuracy and speed of structure learning algorithms. Int J Approx Reason 115:235–253
Shenoy PP, West JC (2011) Inference in hybrid Bayesian networks using mixtures of polynomials. Int J Approx Reason 52(5):641–657
Spirtes P, Glymour C, Scheines R (2001) Causation, prediction, and search, 2nd edn. The MIT Press, Cambridge
Sugiyama M, Suzuki T, Kanamori T (2012) Density ratio estimation in machine learning. Cambridge University Press, Cambridge
Thiesson B, Meek C, Chickering DM, Heckerman D (1998) Learning mixtures of DAG models. In: Proceedings of the fourteenth conference on uncertainty in artificial intelligence. Morgan Kaufmann Publishers, Los Altos, pp 504–513
Tsamardinos I, Brown LE, Aliferis CF (2006) The max–min hill-climbing Bayesian network structure learning algorithm. Mach Learn 65(1):31–78
Watson GS (1964) Smooth regression analysis. Sankhya Indian J Stat Ser A (1961–2002) 26(4):359–372