Conditional Density Approximations with Mixtures of Polynomials

Gherardo Varando[a,*], Pedro L. López-Cruz[a], Thomas D. Nielsen[b], Pedro Larrañaga[a], Concha Bielza[a]

[a] *Computational Intelligence Group, Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Spain*
[b] *Department of Computer Science, Aalborg University, Denmark*

Abstract

Mixtures of polynomials (MoPs) are a non-parametric density estimation technique especially designed for hybrid Bayesian networks with continuous and discrete variables. Algorithms to learn one- and multi-dimensional (marginal) MoPs from data have recently been proposed. In this paper we introduce two methods for learning MoP approximations of conditional densities from data. Both approaches are based on learning MoP approximations of the joint density and the marginal density of the conditioning variables, but they differ as to how the MoP approximation of the quotient of the two densities is found. We illustrate and study the methods using data sampled from known parametric distributions, and we demonstrate their applicability by learning models based on real neuroscience data. Finally, we compare the performance of the proposed methods with an approach for learning mixtures of truncated basis functions (MoTBFs). The empirical results show that the proposed methods generally yield models that are comparable to or significantly better than those found using the MoTBF-based method.

Keywords: Hybrid Bayesian networks, conditional density estimation, mixtures of polynomials, mixtures of truncated basis functions

1. Introduction

Mixtures of polynomials (MoPs) [1,2], mixtures of truncated basis functions (MoTBFs) [3], and mixtures of truncated exponentials (MTEs) [4] have recently been proposed as non-parametric density estimation techniques for hybrid Bayesian networks (BNs) that include both continuous and discrete random variables (MoTBF include MTEs and MoPs as special cases, and at a slight loss of precision we will sometimes simplify the presentation by simply referring to this joint collection of potentials as MoTBF potentials and MoTBF networks by extension). These classes of densities are closed under multiplication and marginalization, and they therefore support exact inference

*Corresponding author
Email addresses: gherardo.varando@upm.es (Gherardo Varando), pedro.lcruz@upm.es (Pedro L. López-Cruz), tdn@cs.aau.dk (Thomas D. Nielsen), pedro.larranaga@fi.upm.es (Pedro Larrañaga), mcbielza@fi.upm.es (Concha Bielza)

schemes over Bayesian networks without deterministic conditionals, based on the Shenoy-Shafer architecture[5,6]. Furthermore, the densities are integrable in closed form, thereby avoiding any structural constraints on the model, unlike, e.g., conditional linear Gaussian (CLG) networks.

Typically, an MoTBF network is constructed by either making an MoTBF-translation of the densities in an existing hybrid network or by automatically learning the MoTBF densities from data. Methods for translating standard statistical density functions have been explored, e.g. in Cobb *et al.*[7] and include regular discretization as a special case. For learning MoTBF densities, research has mainly been directed towards learning univariate densities from data. Moral *et al.*[8] and Romero *et al.*[9] used iterative least squares estimation procedures to obtain MTE-potentials based on, respectively, an empirical histogram and a kernel-based density representation of the data. Although least squares estimation procedures may provide potentials with good generalization properties, there is no guarantee that the estimated parameter values will be close to the maximum likelihood values. This shortcoming has motivated alternative learning schemes that perform direct maximum likelihood estimation. For example, Langseth *et al.*[10] consider optimizing the likelihood function using numerical methods, whereas Langseth *et al.*[11,12] use a kernel-density estimate of the data as a proxy for learning the maximum likelihood parameters, and López-Cruz *et al.*[13] present a maximum likelihood-based learning method relying on B-spline interpolation.

In spite of the advances in learning univariate densities, methods for learning conditional densities have so far only receive limited attention. There are two immediate approaches for learning conditional MoTBF densities: 1) express the conditional density $f(x \mid \mathbf{y})$ as the quotient $f(x, \mathbf{y})/f(\mathbf{y})$ and learn an MoTBF representation $\varphi(x \mid \mathbf{y})$ by finding MoTBF representations of the two components in the quotient; 2) learn an MoTBF representation of $f(x \mid \mathbf{y})$ directly from the data. The problem with the first approach is that neither MoPs, MTEs, nor MoTBFs are closed under division, hence the resulting potential does not belong to the class of MoTBF-potentials. The second approach is hampered by the difficulty of ensuring that the learned MoTBF representation is a proper conditional density. In general, the learning problem can be considered an overspecified optimization problem, where we have an uncountable number of constraints (one for each value of the conditioning variables), but only a finite number of parameters[14]. Hence, directly learning $\varphi(x \mid \mathbf{y})$ from data is not immediately feasible. As a result of these difficulties, conditional MoTBFs are typically being obtained by simply discretizing the parent variables and learning a marginal density for each of the discretized regions of these variables. Thus, the estimation of a conditional density is equivalent to estimating a collection of marginal densities, where the correlation between the variable and the conditioning variables is captured by the discretization procedure only; each marginal density is a constant function over the region for which it is defined[11,14]. One exception to this approach is a recently proposed specification/translation method by Shenoy[2] who defines MoPs based on hyper-rhombuses which generalize the hyperrectangles underlying the traditional

MoP definition. However, this extension mainly addresses the need for modeling multi-dimensional linear deterministic conditionals as well as high-dimensional CLG-distributions.

In this paper, we present two new methods for learning conditional MoP densities, one is based on conditional sampling and the other on interpolation. Thus, our approaches differ from previous methods in several ways. Firstly, as opposed to Shenoy and West [1], Shenoy [2], and Langseth *et al.* [3], we learn conditional MoPs directly from data without any parametric assumptions. Secondly, we do not rely on a discretization of the conditioning variables to capture the correlation among the variables [11,14]. On the downside, the conditional MoPs being learned are not guaranteed to be proper conditional densities, hence the posterior distributions established during inference have to be normalized so that they integrate to one. We analyze the methods using data sampled from known parametric distributions as well as real-world neuro-science data. Finally, we compare the proposed methods with an algorithm for learning MoTBFs [11]. The results show that the proposed methods generally yield results that are either comparable to or significantly better than those obtained using the MoTBF-based method.

The results in this paper extend those published in López-Cruz *et al.* [15]. In comparison, the added contributions of the present paper include a new method for learning the structure defining parameters of the conditional MoP potentials. The empirical analysis is extended to also cover the new learning method and we expand on the scope of this analysis by including additional data sets (both synthetic and real-world).

The paper is organized as follows. Section 2 reviews MoPs. Section 3 details the two new approaches for learning conditional MoPs and provides an empirical study with artificial data sampled from known distributions. An experimental comparison with MoTBFs is shown in Section 4. Section 5 includes the application of the new methods to real neuroscience data. Section 6 ends with conclusions and outlines future work.

2. Preliminaries

In this section we review the one- and multi-dimensional MoP approximations of a probability density function and how they are learnt using B-spline interpolation.

2.1. Mixtures of Polynomials

Let X be a one-dimensional continuous random variable with probability density $f_X(x)$. Shenoy and West [1] defined a one-dimensional MoP approximation of $f_X(x)$ over a closed domain $\Omega_X = [\epsilon_X, \xi_X] \subset \mathbb{R}$ as an L_X-piece d_X-degree piecewise function of the form

$$\varphi_X(x) = \begin{cases} pol_{l_X}(x) & \text{for } x \in A_{l_X},\ l_X = 1, \ldots, L_X \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $pol_{l_X}(x) = b_{0,l_X} + b_{1,l_X}x + b_{2,l_X}x^2 + \cdots + b_{d_X,l_X}x^{d_X}$ is a polynomial function with degree d_X (and order $r_X = d_X + 1$), $b_{0,l_X}, \ldots, b_{d_X,l_X}$ are constants and $A_1, \ldots, A_{L_X}$ are disjoint intervals in Ω_X, which do not depend on x and with $\Omega_X = \cup_{l_X=1}^{L_X} A_{l_X}$.

Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a multi-dimensional continuous random variable with probability density $f_{\mathbf{X}}(\mathbf{x})$. A multi-dimensional MoP approximation[1] of $f_{\mathbf{X}}(\mathbf{x})$ over a closed domain $\Omega_{\mathbf{X}} = [\epsilon_1, \xi_1] \times \cdots \times [\epsilon_n, \xi_n] \subset \mathbb{R}^n$ is an L-piece d-degree piecewise function of the form

$$\varphi_{\mathbf{X}}(\mathbf{x}) = \begin{cases} pol_l(\mathbf{x}) & \text{for } \mathbf{x} \in A_l,\ l = 1, \ldots, L, \\ 0 & \text{otherwise,} \end{cases}$$

where $pol_l(\mathbf{x})$ is a multivariate polynomial function with degree d (and order $r = d + 1$) and $A_1, \ldots, A_L$ are disjoint hyperrectangles in $\Omega_{\mathbf{X}}$, which do not depend on $\mathbf{x}$ and with $\Omega_{\mathbf{X}} = \cup_{l=1}^{L} A_l$. d is defined as the maximum degree of any multivariate monomial for all $l = 1, ..., L$.

If $\varphi_{\mathbf{X}}(\mathbf{x}) \geq 0$ and $\int_{\Omega_{\mathbf{X}}} \varphi_{\mathbf{X}}(\mathbf{x})d\mathbf{x} = 1$, then $\varphi_{\mathbf{X}}$ is said to be a *density*. We say that $\varphi_{X_1|\mathbf{X}'}(x_1|\mathbf{x}')$ is a conditional density for X_1 given $\mathbf{x}' = (x_2, \ldots, x_n)$ if $\varphi_{X_1|\mathbf{X}'}(x_1|\mathbf{x}') \geq 0$ and $\int_{\epsilon_1}^{\xi_1} \varphi_{\mathbf{X}}(x_1, \mathbf{x}')dx_1 = 1$ for all $\mathbf{x}' \in \Omega_{\mathbf{X}'} = [\epsilon_2, \xi_2] \times \cdots \times [\epsilon_n, \xi_n]$.

Example. The following $\varphi_{\mathbf{X}}(x_1, x_2, x_3)$ is an example of an MoP approximation with $L = 4$ pieces and degree $d = 7$ defined for $\mathbf{X} = (X_1, X_2, X_3)$ in the closed domain $\Omega_{\mathbf{X}} = [-4, 4] \times [-4, 4] \times [-4, 4] \subset \mathbb{R}^3$:

$$\varphi_{\mathbf{X}}(x_1, x_2, x_3) = \begin{cases} ax_1^2x_2x_3^2 & \text{for } -4 \leq x_1 \leq 0, -4 \leq x_2 \leq 0, -4 \leq x_3 \leq 4, \\ bx_1^4x_2x_3^2 + cx_3^3 & \text{for } -4 \leq x_1 \leq 0, 0 < x_2 \leq 4, -4 \leq x_3 \leq 4, \\ dx_1^5x_3 & \text{for } 0 < x_1 \leq 4, -4 \leq x_2 \leq 0, -4 \leq x_3 \leq 4, \\ ex_2^2x_3^3 & \text{for } 0 < x_1 \leq 4, 0 < x_2 \leq 4, -4 \leq x_3 \leq 4, \\ 0 & \text{otherwise,} \end{cases}$$

where $a, b, c, d, e \in \mathbb{R}$.

2.2. Learning MoPs Using B-Spline Interpolation

Shenoy and West found MoP approximations of known parametric univariate probability density functions $f_X(x)$ by using two methods: (a) computing the Taylor series expansion[1] around the middle point of each subinterval A_{l_X} in the MoP, and (b) estimating $pol_{l_X}(x)$ as the Lagrange interpolation[2] over the Chebyshev points defined in A_{l_X}. Method (a) needs to know the mathematical expression of the probability density $f_X(x)$, whereas method (b) requires the true probability densities of the Chebyshev points in each A_{l_X}. Moreover, Taylor series expansion cannot ensure that MoP approximations are valid densities, i.e., they are non-negative and integrate

to one, and although Lagrange interpolation can ensure non-negativity it cannot ensure that the resulting MoP integrates to one.

In López-Cruz *et al.*[13], a new proposal for learning MoP approximations of one- and multi-dimensional probability densities from data using B-spline interpolation does not assume any prior knowledge about the true density. It ensures that the resulting MoP approximation is non-negative and integrates to one and provides maximum likelihood estimators of some parameters. Additionally, it ensures that the obtained densities are continuous, which can be advantageous in some scenarios, e.g., for visual analysis or expert validation.

B-splines or basis splines[16] are polynomial curves that form a basis for the space of piecewise polynomial functions[17] over a closed domain $\Omega_X = [\epsilon_X, \xi_X] \subset \mathbb{R}$. Given an increasing knot sequence (or split points) of $L_X + 1$ real numbers $\boldsymbol{\delta}_X = \{a_0, a_1, \ldots, a_{L_X}\}$ in the approximation domain $\Omega_X = [\epsilon_X, \xi_X]$ with $a_{i-1} < a_i$, $\epsilon_X = a_0$ and $\xi_X = a_{L_X}$, one can define $M_X = L_X + r_X - 1$ different B-splines with order r_X spanning the whole domain Ω_X. The j_Xth B-spline $B^{r_X}_{X,j_X}(x), j_X = 1, \ldots, M_X$, is

$$B^{r_X}_{X,j_X}(x) = (a_{j_X} - a_{j_X - r_X}) H(x - a_{j_X - r_X}) \sum_{t=0}^{r_X} \frac{(a_{j_X - r_X + t} - x)^{r_X - 1} H(a_{j_X - r_X + t} - x)}{w'_{j_X - r_X}(a_{j_X - r_X + t})}, \quad x \in \Omega_X, \tag{2}$$

where $w'_{j_X - r_X}(x)$ is the first derivative of $w_{j_X - r_X}(x) = \prod_{u=0}^{r_X}(x - a_{j_X - r_X + u})$ and $H(x)$ is the Heaviside function

$$H(x) = \begin{cases} 1 & x \geq 0, \\ 0 & x < 0. \end{cases}$$

A B-spline $B^{r_X}_{X,j_X}(x)$ can be written as an MoP function with L_X pieces, where each piece $pol_{l_X}(x)$ is defined as the expansion of Equation (2) in the interval $A_{l_X} = [a_{l_X - 1}, a_{l_X}), l_X = 1, \ldots, L_X$. B-splines have a number of interesting properties[18] for approximating probability densities, e.g., $B^{r_X}_{X,j_X}(x)$ is right-side continuous, differentiable, positive in and zero outside $(a_{j_X}, a_{j_X - r_X})$.

Zong[19] proposed using B-spline interpolation to find an approximation of the one-dimensional density $f_X(x)$ as a linear combination of $M_X = L_X + r_X - 1$ B-splines

$$\varphi_X(x; \boldsymbol{\alpha}) = \sum_{j_X=1}^{M_X} \alpha_{j_X} B^{r_X}_{X,j_X}(x), \quad x \in \Omega_X, \tag{3}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{M_X})$ are the mixing coefficients and $B^{r_X}_{X,j_X}(x), j_X = 1, \ldots, M_X$ are B-splines with order r_X (degree $d_X = r_X - 1$) as defined in Equation (2).

Therefore, the MoP defined using B-spline interpolation requires four kinds of parameters: the order (r_X), the number of intervals/pieces (L_X), the knot sequence ($\boldsymbol{\delta}_X$) and the mixing coefficients ($\boldsymbol{\alpha}$). In López-Cruz *et al.*[13] we used uniform B-splines, i.e. equal-width intervals A_{l_X}, to determine the knots in $\boldsymbol{\delta}_X$. r_X and L_X were found by trying different values and selecting

those with the highest BIC score (see Section 3.3). We used the Zong's[19] iterative procedure for computing the maximum likelihood estimators of the mixing coefficients, $\hat{\boldsymbol{\alpha}}$.

Zong and Lam's[20] and Zong's[19] methods for two-dimensional densities were extended in López-Cruz *et al.*[13] to general n-dimensional joint probability density functions. Given a vector of n random variables $\mathbf{X} = (X_1, \ldots, X_n)$, the joint probability density function $f_{\mathbf{X}}(\mathbf{x})$ is approximated with a multidimensional linear combination of B-splines:

$$\varphi_{\mathbf{X}}(\mathbf{x}; \boldsymbol{\alpha}) = \sum_{\substack{j_{X_1}=1,\ldots,M_{X_1} \\ \vdots \\ j_{X_n}=1,\ldots,M_{X_n}}} \alpha_{j_{X_1},\ldots,j_{X_n}} \prod_{i=1}^{n} B^{r_{X_i}}_{X_i,j_{X_i}}(x_i), \quad \mathbf{x} \in \Omega_{\mathbf{X}}, \tag{4}$$

where r_{X_i} is the order of the B-splines for variable X_i, $M_{X_i} = L_{X_i} + r_{X_i} - 1$ is the number of B-splines for variable X_i, L_{X_i} is the number of pieces for variable X_i, and $\alpha_{j_{X_1},\ldots,j_{X_n}}$ is the mixing coefficient for the combination of B-splines given by the indices $j_{X_1}, \ldots, j_{X_n}$.

Thus, the multidimensional MoP requires four kinds of parameters: the number of intervals $(L_{X_1}, \ldots, L_{X_n})$, the order of the polynomials $(r_{X_1}, \ldots, r_{X_n})$, the knot sequence $(\boldsymbol{\delta}_{\mathbf{X}})$ and the mixing coefficients $(\boldsymbol{\alpha})$. In López-Cruz *et al.*[13] we used the multidimensional knots given by the Cartesian product of the knot sequences of each dimension $\boldsymbol{\delta}_{\mathbf{X}} = \boldsymbol{\delta}_{X_1} \times \cdots \times \boldsymbol{\delta}_{X_n}$, where $\boldsymbol{\delta}_{X_i}$ correspond to equal-width intervals as in the one-dimensional case. Similarly, the mixing coefficient vector has one value for each combination of one-dimensional B-splines, i.e., $\boldsymbol{\alpha} = (\alpha_{1,\ldots,1}, \ldots, \alpha_{M_{X_1},\ldots,M_{X_n}})$. The resulting MoP has $L = \prod_{i=1}^{n} L_{X_i}$ pieces, where each piece $pol_{l_{X_1},\ldots,l_{X_n}}(\mathbf{x})$ is defined in an n-dimensional hyperrectangle $A_{l_{X_1},\ldots,l_{X_n}} = \left[a_{l_{X_1}-1}, a_{l_{X_1}}\right] \times \cdots \times \left[a_{l_{X_n}-1}, a_{l_{X_n}}\right]$.

3. Learning Conditional Distributions

Given a sample $\mathcal{D}_{X,\mathbf{Y}} = \{(x_i, \mathbf{y}_i), i = 1, \ldots, N\}$, from the joint density of $(X, \mathbf{Y})$, the aim is to learn an MoP approximation $\varphi_{X|\mathbf{Y}}(x|\mathbf{y})$ of the conditional density $f_{X|\mathbf{Y}}(x|\mathbf{y})$ of $X|\mathbf{Y}$ from $\mathcal{D}_{X,\mathbf{Y}}$. Following the terminology used for BNs, we consider the conditional random variable X as the child variable and the vector of conditioning random variables $\mathbf{Y} = (Y_1, \ldots, Y_n)$ as the parent variables.

3.1. Learning Conditional MoPs Using Sampling

The proposed method is based on first obtaining a sample from the conditional density of $X|\mathbf{Y}$ and then learning a conditional MoP density from the sampled values. Algorithm 1 shows the main steps of the procedure.

Algorithm 1.

Input: A training dataset $\mathcal{D}_{X,\mathbf{Y}} = \{(x_i, \mathbf{y}_i), i = 1, \ldots, N\}$.

Output: $\varphi_{X|\mathbf{Y}}(x|\mathbf{y})$, the MoP approximation of the density of $X|\mathbf{Y}$.

Steps:

1. Learn an MoP $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})$ of the joint density of $(X,\mathbf{Y})$ from the dataset $\mathcal{D}_{X,\mathbf{Y}}$ using the algorithm in López-Cruz *et al.*[13].
2. Marginalize out X from $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})$ to yield an MoP $\varphi_{\mathbf{Y}}(\mathbf{y})$ of the marginal density of the parent variables $\mathbf{Y}$: $\varphi_{\mathbf{Y}}(\mathbf{y}) = \int_{\Omega_X} \varphi_{X,\mathbf{Y}}(x,\mathbf{y})dx$.
3. Use the Metropolis-Hastings algorithm (Algorithm 2) to produce a sample $\mathcal{D}_{X|\mathbf{Y}}$ from a density proportional to the conditional density $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$.
4. Find an unnormalized conditional MoP $\varphi^{(u)}_{X|\mathbf{Y}}(x|\mathbf{y})$ based on $\mathcal{D}_{X|\mathbf{Y}}$ and using the algorithm in López-Cruz *et al.*[13].
5. Partially normalize the conditional MoP $\varphi^{(u)}_{X|\mathbf{Y}}(x|\mathbf{y})$ to make it integrate the Lebesgue measure of the $\mathbf{Y}$ domain (as the true conditional density).

First, we find an MoP representation of the joint density $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})$ (step 1) using the B-spline interpolation approach proposed in López-Cruz *et al.*[13] and reviewed in Section 2. Second, we obtain an MoP of the marginal density of the parents $\varphi_{\mathbf{Y}}(\mathbf{y})$ by marginalization (step 2). Next, we use a sampling algorithm to obtain a sample $\mathcal{D}_{X|\mathbf{Y}}$ from a distribution proportional to the conditional density of $X|\mathbf{Y}$ (step 3), where the conditional density values are obtained by evaluating the quotient $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$. More specifically, we have used a standard Metropolis-Hastings sampler for the reported experimental results, as specified in Algorithm 2. Finally, we find an MoP approximation, $\varphi^{(u)}_{X|\mathbf{Y}}(x|\mathbf{y})$, from data set $\mathcal{D}_{X|\mathbf{Y}}$ (step 4). The MoP $\varphi^{(u)}_{X|\mathbf{Y}}(x|\mathbf{y})$ is an approximation of a proper density which is proportional to the conditional density $f_{X|\mathbf{Y}}(x|\mathbf{y})$. To normalize it we know that

$$\int_{\Omega_X\times\Omega_{\mathbf{Y}}} f_{X|\mathbf{Y}}(x|\mathbf{y})\mathrm{d}x\mathrm{d}y = \int_{\Omega_{\mathbf{Y}}} 1\mathrm{d}y = |\Omega_{\mathbf{Y}}|\,.$$

Consequently, to find the partial normalization constant, we can impose the analogous constraint to the approximating MoP. In particular we find K such that

$$\frac{1}{K}\int_{\Omega_X\times\Omega_{\mathbf{Y}}} \varphi^{(u)}_{X|\mathbf{Y}}(x|\mathbf{y})\mathrm{d}x\mathrm{d}y = |\Omega_{\mathbf{Y}}|\,,$$

and set $\varphi_{X|\mathbf{Y}}(x|\mathbf{y}) = \frac{1}{K}\varphi^{(u)}_{X|\mathbf{Y}}(x|\mathbf{y})$ as the approximating MoP of the conditional density $f_{X|\mathbf{Y}}(x|\mathbf{y})$ (step 5).

For the sampling process described in Algorithm 2, we generate uniformly distributed values over $\Omega_{\mathbf{Y}}$ for the parent variables $\mathbf{Y}$, whereas we use a Gaussian distribution $Q(x_{new};x) \equiv \mathcal{N}(x,\sigma)$ as proposal distribution for the child variable; in the experiments we set $\sigma^2 = 0.5$. The Metropolis-Hastings algorithm is a Markov Chain Monte Carlo method; i.e., it is based on building a Markov chain which has as stationary distribution the one we would like to sample from. Consequently we have to wait (termed the *burn in* period) until the Markov chain is close to its stationary distribution before sampling from it. This is the purpose of discarding the first h values. Another

consequence of the Metropolis-Hastings algorithm is the correlation that may be present between near sampled values, which follows from the Markov chain assumption. This is partially avoided by setting a *jumping width*, h'.

Algorithm 2.

Input: $\varphi_{X|\mathbf{Y}}(x|\mathbf{y})$, an approximation to the conditional density of $X|\mathbf{Y}$.

Output: $\mathcal{D}_{X|\mathbf{Y}}$ a sample of a distribution, with density proportional to the conditional density $\varphi_{X|\mathbf{Y}}(x|\mathbf{y})$.

Steps:

1. Initialize $x = x_0$, $\mathbf{y} = \mathbf{y}_0$
2. Generate a candidate (x_{new}, y_{new}) from the product of a proposal distribution for the X_{new} variable, $Q(X_{new}; x)$, and an independent uniform distribution for $\mathbf{y}_{new}$.
3. Calculate the acceptance ratio $t = \varphi_{X|\mathbf{Y}}(x_{new}|\mathbf{y}_{new})/\varphi_{X|\mathbf{Y}}(x|\mathbf{y})$
4. if $t \geq u$, where u is a realization from a uniform distribution in $[0, 1]$, the candidate is accepted and we set $(x, \mathbf{y}) = (x_{new}, \mathbf{y}_{new})$, otherwise the candidate is rejected and the old values $(x, \mathbf{y})$ are kept.
5. Repeat from step 1, discarding the first h values generated and storing the following values, one every h' repetitions.

The proposed method has some interesting properties. The B-spline interpolation algorithm for learning MoPs in López-Cruz *et al.*[13] guarantees that the approximations are continuous, non-negative and integrate to one. Therefore, the conditional MoPs obtained with Algorithm 1 are also continuous and non-negative. Continuity is not required for inference in BNs, but it is usually a desirable property, e.g., for visualization purposes. The algorithm provides maximum likelihood estimators of the mixing coefficients of the linear combination of the B-splines when learning MoPs of the joint density $\varphi_{X\mathbf{Y}}(x, \mathbf{y})$ and the marginal density $\varphi_{\mathbf{Y}}(\mathbf{y})$. Hence the quotient $\varphi_{X,\mathbf{Y}}(x, \mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$ corresponds to a maximum likelihood model of the conditional distribution. It should be noted, though, that this property is not shared by the model learned in step 4, i.e., it is not necessarily a maximum likelihood model. Furthermore, since the partial normalization (step 5) does not ensure that the learned MoP is a proper conditional density, the posterior densities may need to be normalized to integrate to one during inference.

3.2. Learning Conditional MoPs Using Interpolation

The preliminary empirical results provided by Algorithm 1 show that the sampling approach can produce good approximations. However, it is difficult to control or guarantee the quality of the approximation due to the sampling procedure and the partial normalization in the last step.

This shortcoming has motivated an alternative method for learning an MoP approximation of a conditional probability density for $X|\mathbf{Y}$. The main steps of the procedure are summarized in Algorithm 3. First, we find MoP approximations of both the joint density of $(X, \mathbf{Y})$ and the marginal density of $\mathbf{Y}$ following the same procedure as in Algorithm 1 (steps 1 and 2). Next, we build the conditional MoP $\varphi_{X|\mathbf{Y}}(x|\mathbf{y})$ by finding, for each piece $pol_l(x, \mathbf{y})$ defined in the hyperrectangle A_l, a multidimensional interpolation polynomial of the function given by the quotient of the joint and the marginal densities $\varphi_{X,\mathbf{Y}}(x, \mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$.

Algorithm 3.

Input: A training dataset $\mathcal{D}_{X,\mathbf{Y}} = \{(x_i, \mathbf{y}_i), i = 1, \ldots, N\}$.

Output: $\varphi_{X|\mathbf{Y}}(x|\mathbf{y})$, the MoP approximation of the density of $X|\mathbf{Y}$.

Steps:

1. Find an MoP $\varphi_{X,\mathbf{Y}}(x, \mathbf{y})$ of the joint density of the variables X and $\mathbf{Y}$ from the dataset $\mathcal{D}_{X,\mathbf{Y}}$ using the method in López-Cruz *et al.*[13].
2. Marginalize out X from $\varphi_{X,\mathbf{Y}}(x, \mathbf{y})$ to obtain an MoP $\varphi_{\mathbf{Y}}(\mathbf{y})$ of the marginal density of the parent variables $\mathbf{Y}$: $\varphi_{\mathbf{Y}}(\mathbf{y}) = \int_{\Omega_X} \varphi_{X,\mathbf{Y}}(x, \mathbf{y})dx$.
3. For piece $pol_l(x, \mathbf{y})$, defined in A_l, $l = 1, \ldots, L$, in the conditional MoP $\varphi_{X|\mathbf{Y}}(x|\mathbf{y})$:

   Find a multi-dimensional polynomial approximation of the function $g(x, \mathbf{y}) = \varphi_{X,\mathbf{Y}}(x, \mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$ using an interpolation method with polynomial degree equal to the degree of the MoP of the joint density.

We consider two multidimensional interpolation methods to obtain the polynomials of the pieces $pol_l(x, \mathbf{y})$ in step 3 of Algorithm 3:

- The multidimensional Taylor series expansion (TSE) for a point yields a polynomial approximation of any differentiable function g. The quotient of any two functions is differentiable as long as the two functions are also differentiable and the denominator is not zero. In our scenario, polynomials are differentiable functions and, thus, we can compute the TSE of the quotient of two polynomials. Consequently, we can use multidimensional TSEs to find a polynomial approximation of $g(x, \mathbf{y}) = \varphi_{X,\mathbf{Y}}(x, \mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$ for each piece $pol_l(x, \mathbf{y})$. We computed these TSEs of $g(x, \mathbf{y})$ for the midpoint of the hyperrectangle A_l.

- Lagrange interpolation (LI) finds a polynomial approximation of any function g. Before finding the LI polynomial, we need to evaluate function g on a set of interpolation points. In the one-dimensional scenario, Chebyshev points are frequently used as interpolation points[21]. However, multidimensional LI is not a trivial task because it is difficult to find good interpolation points in a multidimensional space. Some researchers have recently addressed the

two-dimensional scenario[21,22]. To find a conditional MoP using LI, we first find and evaluate the conditional density function $g(x, \mathbf{y}) = \varphi_{X,\mathbf{Y}}(x, \mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$ on the set of interpolation points in A_l. Next, we compute the polynomial $pol_l(x, \mathbf{y})$ for the piece as the LI polynomial over the interpolation points defined in A_l. Note that other approaches, e.g., kernel-based conditional estimation methods, can also be used to evaluate the conditional density $g(x, \mathbf{y})$ on the set of interpolation points.

Compared with Algorithm 1, there are some apparent (dis)advantages. First, the conditional MoPs produced by Algorithm 3 are not necessarily continuous. Second, interpolation methods cannot in general ensure non-negativity, although LI can be used to ensure it by increasing the order of the polynomials. On the other hand, the learning method in Algorithm 3 does not need a partial normalization (step 2). Thus, if the polynomial approximations are close to the conditional density $\varphi_{X,\mathbf{Y}}(x, \mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$, then the conditional MoP using these polynomial interpolations is expected to be close to normalized. As a result, we can more directly control the quality of the approximation by varying the degree of the polynomials and the number of hyperrectangles. It should be observed that both Algorithm 1 and 3 output MoPs approximations, but the approximations are built differently and lead to different models. Algorithm 1 uses B-spline interpolation and so the number of parameters in the resulting models is

$$(L_X + r_X - 1) \prod_{i=1}^{n} (L_{Y_i} + r_{Y_i} - 1) .$$

On the other hand, Algorithm 3 builds MoPs that are not necessarily continuous and therefore more general. The number of parameters in the learned models is

$$r_X L_X \prod_{i=1}^{n} L_{Y_i} r_{Y_i} .$$

3.3. Heuristic to Search for a Good MoP Approximation

Steps 1 and 4 in Algorithm 1 and step 1 in Algorithm 3 require finding an MoP approximation starting from a data set $\mathcal{D}_{X,\mathbf{Y}}$. The algorithm proposed in López-Cruz *et al.*[13] provides a way to compute a multi-dimensional MoP approximation given a data set, the orders of the polynomials, and the pieces of the domains of approximation for each dimension. Here we use a penalized likelihood-based search iterating over the algorithm in López-Cruz *et al.*[13] in order to find the best MoP approximation for the data set $\mathcal{D}_{X,\mathbf{Y}}$. The method performs a simple greedy search for the optimal parameters. From now on we will refer to those parameters as follows: r is the order of the polynomials in each dimension, L_X is the number of pieces for variable X, and L_Y is the number of pieces for variable Y (to simplify the presentation we assume a single parent variable). The algorithm starts from the initial point $(r, L_X, L_Y) = (2, 1, 1)$, computes the MoP with these

parameters and compares its BIC score to those of the nearest neighbour solutions: $(r+1, L_X, L_Y)$, (r, L_X+1, L_Y), (r, L_X, L_Y+1) and (r, L_X+1, L_Y+1). The parameters (r, L_X, L_Y) are updated to the best ones and the steps are iterated until no improvement in BIC score is achieved or the parameters (r, L_X, L_Y) reach some user predefined boundaries. Algorithm 4 lists the steps of this heuristic search.

Algorithm 4.

Input: A training dataset $\mathcal{D}_{X,Y} = \{(x_i, y_i), i = 1, ..., N\}$.

Output: $\varphi_{X,Y}(x, y)$, the MoP approximation of the density of (X, Y).

Steps:

1. Set $r = 2$ and $L_X = L_Y = 1$.
2. Calculate, using the method in López-Cruz *et al.*[13], MoP approximations given the dataset $\mathcal{D}_{X,Y}$ with the following parameters:
   - (r, L_X, L_Y).
   - $(r+1, L_X, L_Y)$.
   - (r, L_X+1, L_Y).
   - (r, L_X, L_Y+1).
   - (r, L_X+1, L_Y+1).
3. Compute the BIC score of the five MoPs computed in the previous step with the dataset $\mathcal{D}_{X,Y}$.
4. Select the MoP with the highest BIC score and update r, L_X and L_Y to their parameters.
5. Repeat from step 2 until there is no gain in the BIC score or the maximum boundaries for the parameters are reached.

The BIC score[23] is defined as

$$BIC(\varphi_{X,Y}(x, y), \mathcal{D}_{X,Y}) = \ell(\mathcal{D}_{X,Y}|\varphi_{X,Y}(x, y)) - \frac{\dim(\varphi_{X,Y}(x, y)) \log N}{2},$$

where $\ell(\mathcal{D}_{X,Y}|\varphi_{X,Y}(x, y))$ is the log-likelihood of the training dataset $\mathcal{D}_{X,Y}$ given an MoP model $\varphi_{X,Y}(x, y)$, N is the number of observations in the dataset $\mathcal{D}_{X,Y}$ and $\dim(\varphi_{X,Y}(x, y))$ is the number of free parameters in the model encoding the split points and the coefficients in the polynomials.

The previous algorithm could be implemented with uniform knots or using data-dependent knots. In particular it is possible to use empirical quantiles (i.e. an equal-frequency rather than an equal-width approach), calculated over the data set $\mathcal{D}_{X,Y} = \{(x_i, y_i), i = 1, ..., N\}$.

Conceptually, the algorithm can also easily be extended to handle a multidimensional parent set $\mathbf{Y}$, but at the cost of a considerable increase in the computational complexity. Introducing a multidimensional parent set $\mathbf{Y}$ means that at each iteration of Algorithm 4, we have to compute an increasing number of candidate MoPs resulting in a corresponding increase in the computational cost: If at every iteration we select the best parameter set among all possible combinations of parameters, the number of MoP computations increases exponentially with the size of $\mathbf{Y}$. As an alternative, one may attempt to devise heuristic-based search strategies or constrain the parameter combinations. However, even if this approach would turn out successful we still have to face the fact that Algorithm 4 uses the procedure described in López-Cruz *et al.*[13] to compute multidimensional MoPs and this procedure is not immediately scalable. In summary, to ensure scalability a new algorithm for computing multidimensional MoPs might be developed and more efficient search strategies should be deployed.

3.4. Illustrative Examples

We apply the proposed algorithms to three examples, all of them are thought of as graphical models with two variables, a parent Y and a child X. In the first example we consider a joint Gaussian distribution, $(X,Y) \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}\right)$. This two-dimensional density corresponds to a Gaussian Bayesian network, where $Y \sim \mathcal{N}(0,1)$ and $X|Y \sim \mathcal{N}(y,1)$.

In the second example we consider Y distributed as a Gamma distribution with $rate = 10$ and $shape = 10$, and X distributed, conditionally to $Y = y$, as an exponential distribution with $rate = y$.

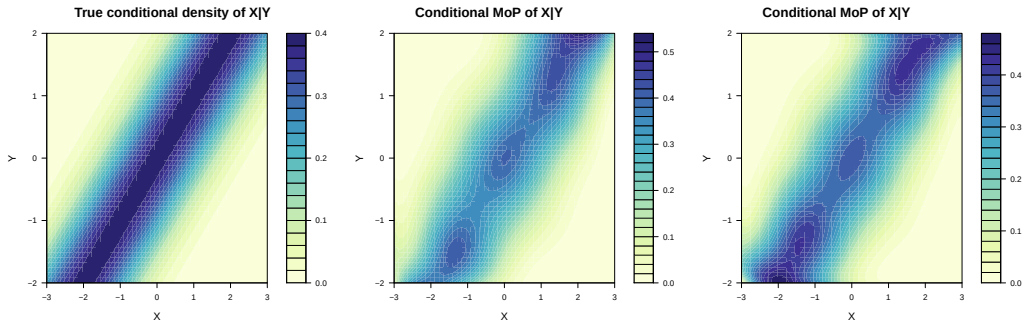
In the third example we model Y as a mixture of two Gaussian distributions, $Y \sim 0.5\mathcal{N}(-3,1) + 0.5\mathcal{N}(3,1)$. The distribution of X, conditioned on $Y = y$ is considered a Gaussian with mean y and unit variance, i.e., $X|Y \sim \mathcal{N}(y,1)$.

For each model we generate sets of ten (X,Y) samples of length equal to $N = 25, 500, 2500, 5000$. For each example we apply the two algorithms (Algorithm 1 and 3) to approximate the conditional density (see Figure 1 for $N = 5000$). In Algorithm 2 we set the parameters h and h' to 1000 and 3 samples, respectively, and in Algorithm 4 we set the boundaries artificially high so that they are not reached.

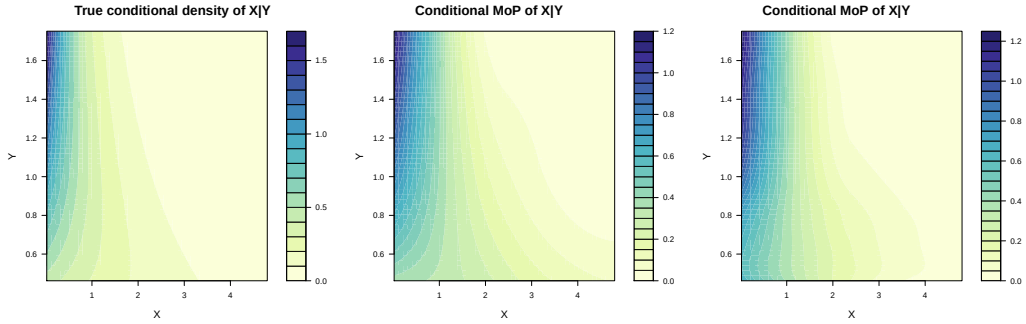
To check the goodness of the learned MoP we evaluate the mean square error (MSE) between the approximated conditional densities $\varphi_{X|Y}(x|y)$ and the true one $f_{X|Y}(x|y)$, for three values of y_0, corresponding to the percentiles 25, 50, 75 of the distribution of Y. The results can be found in Tables 1, 2 and 3. The comparison is done without normalizing the approximated conditional densities $\varphi_{X|Y}(x|y_0)$, hence Kullback-Leibler divergence cannot be used as an evaluation measure.

The results in Tables 1 and 2 show that Algorithms 1 and 3 perform similarly with respect to the Gaussian model, but that Algorithm 3 achieves better results with respect to the Exp-Gamma

(a) $Y \sim \mathcal{N}(0,1)$ and $X|Y \sim \mathcal{N}(y,1)$



(b) $Y \sim \text{Gamma}(rate = 10, shape = 10)$ and $X|Y \sim \text{Exp}(y)$



(c) $Y \sim 0.5\mathcal{N}(-3,1) + 0.5\mathcal{N}(3,1)$ and $X|Y \sim \mathcal{N}(y,1)$

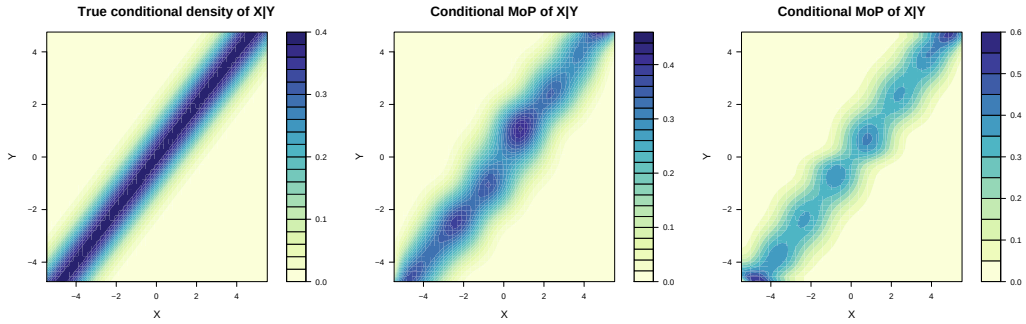


Figure 1: For the three examples (in rows), true conditional density and MoP approximation obtained with Algorithm 1 (second column) and Algorithm 3 with Lagrange Interpolation (third column), case $N = 5000$.

model. The results in Table 3 show that even in the more complex mixture model, the proposed algorithms perform quite well with respect to the mean squared errors. Moreover with respect to the complexity of the learned MoPs, we can see that the algorithms deal with the increasing complexity by learning MoPs with more pieces instead of MoPs with higher orders. The main problem with Algorithm 1 is the partial normalization step and the loose link between the MoP

Table 1: Mean MSE between the MoP approximations and the true conditional densities for ten datasets sampled from the BN, where $Y \sim \mathcal{N}(0,1)$ and $X|Y \sim \mathcal{N}(y,1)$. Mean order r and mean number of pieces in the X and Y domains L_X, L_Y are also reported.

		Alg 1				Alg 3 LI				Alg 3 TSE			
N	$f_{X\|Y}(x\|y)$	MSE	r	L_X	L_Y	MSE	r	L_X	L_Y	MSE	r	L_X	L_Y
	$y = -0.6748$	0.0103				0.0113				0.0114			
25	$y = 0.00$	0.0089	3.1	2	1.5	0.0108	2	1.7	1.3	0.0108	2	1.7	1.3
	$y = 0.6748$	0.0105				0.0123				0.0122			
	$y = -0.6748$	0.0025				0.0031				0.0033			
500	$y = 0.00$	0.0009	4	4	2.6	0.0008	3	3	3	0.0008	3	3	3
	$y = 0.6748$	0.0020				0.0032				0.0031			
	$y = -0.6748$	0.0006				0.0006				0.0006			
2500	$y = 0.00$	0.0002	4	4	4	0.0001	4	4	4	0.0001	4	4	4
	$y = 0.6748$	0.0006				0.0006				0.0006			
	$y = -0.6748$	0.0006				0.0005				0.0005			
5000	$y = 0.00$	0.0002	4	4	4	0.0001	4	4	4	0.0001	4	4	4
	$y = 0.6748$	0.0006				0.0005				0.0005			

Table 2: Mean MSE between the MoP approximations and the true conditional densities for ten datasets sampled from the BN, where $Y \sim \text{Gamma}(rate = 10, shape = 10)$ and $X|Y \sim \text{Exp}(y)$. Mean order r and mean number of pieces in the X and Y domains L_X, L_Y are also reported.

		Alg 1				Alg 3 LI				Alg 3 TSE			
N	$f_{X\|Y}(x\|y)$	MSE	r	L_X	L_Y	MSE	r	L_X	L_Y	MSE	r	L_X	L_Y
	$y = 0.7706$	0.0131				0.0060				0.0059			
25	$y = 0.9684$	0.0225	3.5	2.8	1	0.0117	2	1.5	1.2	0.0121	2	1.5	1.2
	$y = 1.1916$	0.0374				0.0225				0.0226			
	$y = 0.7706$	0.0012				0.0008				0.0009			
500	$y = 0.9684$	0.0022	3	2.4	2.2	0.0005	3	2	1.9	0.0004	3	2	1.9
	$y = 1.1916$	0.0057				0.0016				0.0016			
	$y = 0.7706$	0.0025				0.0004				0.0005			
2500	$y = 0.9684$	0.0043	3.1	3.1	1.8	0.0003	3	2.2	2.5	0.0003	3	2.2	2.5
	$y = 1.1916$	0.0074				0.0009				0.0009			
	$y = 0.7706$	0.0015				0.0003				0.0004			
5000	$y = 0.9684$	0.0022	3	2.2	2	0.0003	3.1	2.3	2.7	0.0003	3.1	2.3	2.7
	$y = 1.1916$	0.0032				0.0006				0.0006			

approximation for the joint density (step 1) and the MoP approximation of the conditional density (steps 4 and 5).

Next, we perform inference based on the MoP learned with the algorithms. We compute the posterior density of $Y|X$ and compare it with the true one (Figure 2, 3 and 4). The comparison is done based on the MSE and the Kullback-Leibler divergence (KL). The posterior density is calculated conditional on nine different values for the child variable, corresponding to the percentiles

Table 3: Mean MSE between the MoP approximations and the true conditional densities for ten datasets sampled from the BN, where $Y \sim 0.5\mathcal{N}(-3,1) + 0.5\mathcal{N}(3,1)$ and $X|Y \sim \mathcal{N}(y,1)$. Mean order r and mean number of pieces in the X and Y domains L_X, L_Y are also reported.

		Alg 1				Alg 3 LI				Alg 3 TSE			
N	$f_{X\|Y}(x\|y)$	MSE	r	L_X	L_Y	MSE	r	L_X	L_Y	MSE	r	L_X	L_Y
	$y=-3$	0.0099				0.0111				0.0111			
25	$y=0$	0.0204	3.9	2.9	3.2	0.0115	2	1.9	2	0.0256	2	1.9	2
	$y=3$	0.0090				0.0109				0.0116			
	$y=-3$	0.0024				0.0031				0.0022			
500	$y=0$	0.0158	4.4	4.5	5.4	0.0157	4	4.1	3.7	0.0156	4	4.1	3.7
	$y=3$	0.0024				0.0025				0.0024			
	$y=-3$	0.0014				0.0007				0.0007			
2500	$y=0$	0.0078	4	6.2	6.2	0.0049	4	6.2	7.3	0.0048	4	6.2	7.3
	$y=3$	0.0015				0.0009				0.0009			
	$y=-3$	0.0012				0.0007				0.0007			
5000	$y=0$	0.0014	4	7.1	7.4	0.0019	4	7	9	0.0019	4	7	9
	$y=3$	0.0011				0.0005				0.0005			

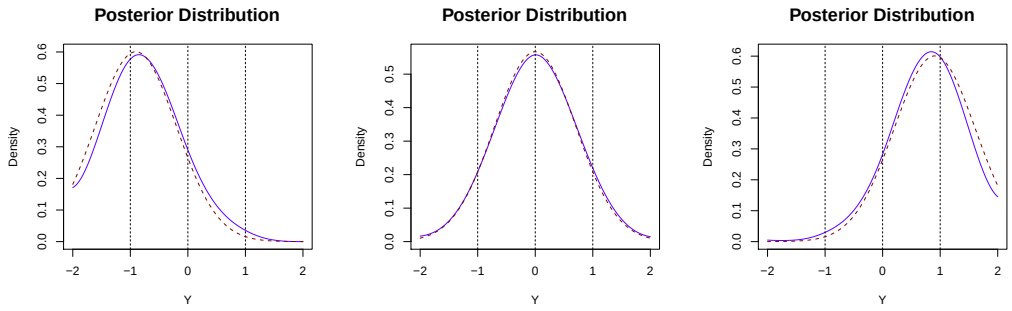
10, 20, 30, 40, 50, 60, 70, 80, 90. The results of the comparison are shown in Tables 4, 5, 6, 7, 8 and 9.

For both algorithms, we cannot ensure that the approximated conditional densities, $\varphi_{X|Y}(x|y_0)$, integrate to one for every y_0. This is not necessarily a problem when doing inference, though, as one may perform an additional normalization step in order to obtain proper densities.

To compare the algorithms we apply a paired Wilcoxon signed-rank test. For every pair of algorithms, for every N and for every fixed value x_{obs} of the conditioning variable we run a Wilcoxon signed-rank over the results of the comparison between the approximated posterior density and the true posterior density. The results are reported in Table 10. We list the number of cases in which the algorithm on the left significantly outperforms (significance level $\alpha = 0.05$) the algorithm on the top. Recall that the total number of cases is 36, for each of the datasets (4 values for N and 9 quantiles corresponding to the x_{obs} values).

With respect to the posterior density approximation, the results of the Wilcoxon signed-rank test based on KL divergence indicate that Algorithm 1 outperforms Algorithm 3 using Lagrange interpolation in the Exp-Gamma model (Table 10). This is visually appreciated in Figure 3. However, for the Gaussian model, Algorithm 3 achieves statistically significant better results with respect to KL (Table 10) in some cases. The mixture model is the one that shows the greatest difference between the two algorithms (Table 10 and Figure 4). From the results of the Wilcoxon signed-rank test with respect to the mixture model we see that Algorithm 3 outperforms Algorithm 1 in almost one-third of the cases (Table 10). When looking closer at the results (Tables 8 and 9) we observe that Algorithm 3 achieves better results for the largest sample cases

(a) Algorithm 1



(b) Algorithm 3

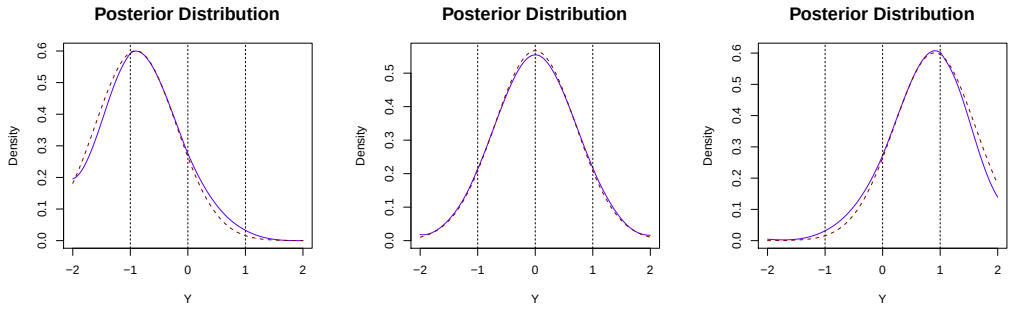


Figure 2: True posterior densities (red dashed) and approximated one (solid blue) for $Y \sim \mathcal{N}(0,1)$ and $X|Y \sim \mathcal{N}(y,1)$, case $N = 5000$.

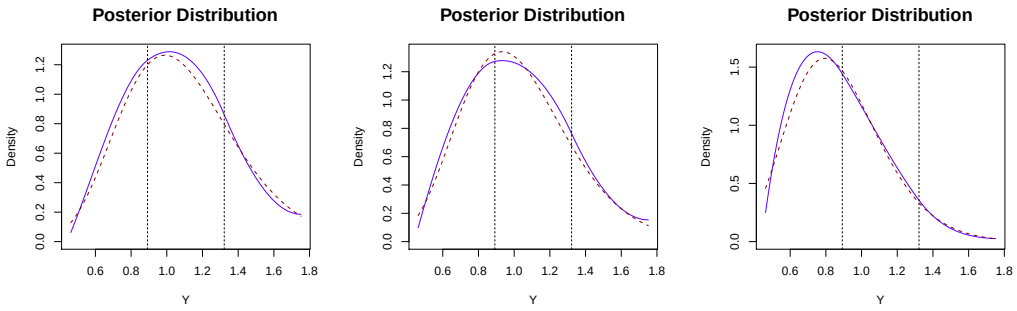
($N = 5000$), where, according to the Wilcoxon signed-rank tests, Algorithm 3 outperforms Algorithm 1 for every value of the child variable. In comparison, the cases for which Algorithm 1 outperforms Algorithm 3 are mainly found when dealing with smaller data sets ($N = 25, 500$).

Note that Algorithm 1 is computationally more costly than Algorithm 3 due to the use of Algorithm 4 in two steps. From Figure 2 and 3 we also see that Algorithm 3 provides posterior densities that are almost continuous in the two first simpler models. In the mixture model, however, the TSE variant of Algorithm 3 outputs MoP approximations of conditional densities which show strong discontinuities in the form of high peaks. Those errors are due to approximation faults in the computations of the ratio between the joint and the marginal distributions in step 3 of Algorithm 3. These errors are not observed using interpolation over Padua points.

Based on the previous observations over the artificial examples as well as the theoretical properties of the algorithms proposed we suggest that:

- When dealing with small datasets and when requiring continuous densities the use of Algorithm 1 provides better results.
- In case of large datasets, Algorithm 3 using interpolation over Padua point is to be preferred; it outputs almost continuous MoPs and is generally faster than Algorithm 1.

(a) Algorithm 1



(b) Algorithm 3

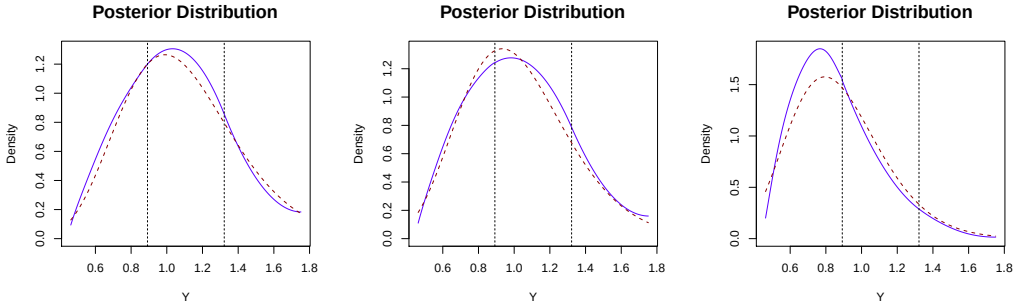


Figure 3: True posterior densities (red dashed) and approximated one (solid blue) for $Y \sim$ Gamma($rate = 10, shape = 10$) and $X|Y \sim \text{Exp}(y)$, case $N = 5000$.

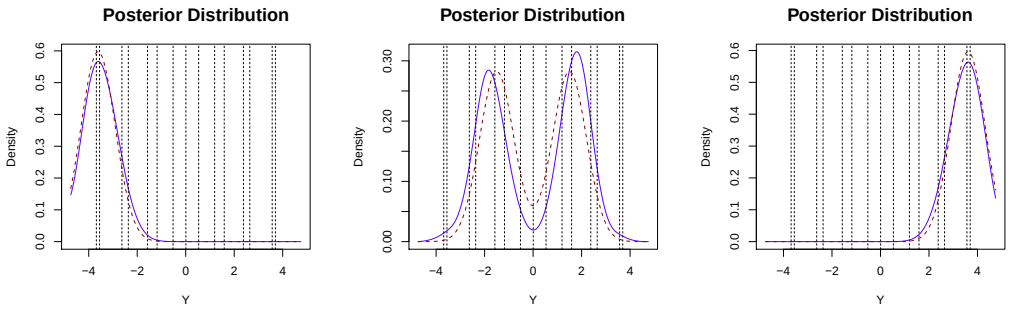
Table 4: Comparison between the true posterior density and the one learned with the MoP approximation obtained using Algorithm 1. Mean KL and MSE for ten datasets sampled from the BN, where $Y \sim \mathcal{N}(0, 1)$ and $X|Y \sim \mathcal{N}(y, 1)$.

N	x_{obs}	-1.8103	-1.1867	-0.7377	-0.3554	0.0000	0.3554	0.7377	1.1867	1.8103
25	KL	0.3312	0.2924	0.2696	0.2578	0.2550	0.2592	0.2700	0.2863	0.3038
	MSE	0.0152	0.0153	0.0146	0.0138	0.0130	0.0124	0.0118	0.0110	0.0092
500	KL	0.0612	0.0516	0.0329	0.0167	0.0099	0.0154	0.0307	0.0497	0.0634
	MSE	0.0013	0.0020	0.0016	0.0008	0.0004	0.0007	0.0015	0.0020	0.0014
2500	KL	0.0115	0.0102	0.0071	0.0031	0.0017	0.0043	0.0092	0.0120	0.0110
	MSE	0.0004	0.0003	0.0003	0.0001	0.0001	0.0002	0.0005	0.0004	0.0002
5000	KL	0.0093	0.0089	0.0063	0.0026	0.0011	0.0031	0.0071	0.0096	0.0102
	MSE	0.0002	0.0003	0.0004	0.0001	0.0001	0.0002	0.0004	0.0003	0.0003

4. A Comparison with MoTBFs

In this section, we compare the two proposed learning methods with the method described in Langseth *et al.*[11] for learning conditional MoTBFs from data. The MoTBF-based learning method

(a) Algorithm 1



(b) Algorithm 3

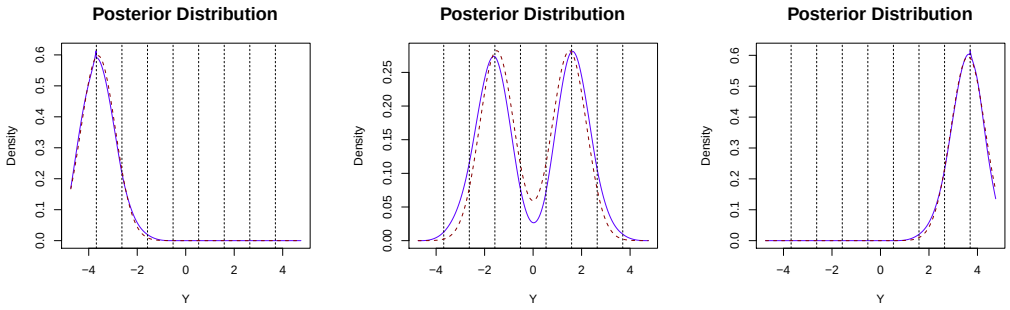


Figure 4: True posterior densities (red dashed) and approximated one (solid blue) for $Y \sim 0.5\mathcal{N}(-3,1)+0.5\mathcal{N}(3,1)$ and $X|Y \sim \mathcal{N}(y,1)$, case $N = 5000$.

Table 5: Comparison between the true posterior density and the one learned with the MoP approximation obtained using Algorithm 3 and Lagrange interpolation. Mean KL and MSE for ten datasets sampled from the BN, where $Y \sim \mathcal{N}(0,1)$ and $X|Y \sim \mathcal{N}(y,1)$.

N	x_{obs}	-1.8103	-1.1867	-0.7377	-0.3554	0.0000	0.3554	0.7377	1.1867	1.8103
25	KL	0.3199	0.2873	0.2752	0.2666	0.2584	0.2631	0.2662	0.2737	0.2937
	MSE	0.0147	0.0151	0.0154	0.0148	0.0131	0.0125	0.0117	0.0108	0.0090
500	KL	0.0586	0.0587	0.0379	0.0163	0.0084	0.0175	0.0395	0.0580	0.0596
	MSE	0.0013	0.0023	0.0019	0.0008	0.0003	0.0008	0.0019	0.0025	0.0012
2500	KL	0.0098	0.0112	0.0081	0.0034	0.0012	0.0031	0.0075	0.0102	0.0087
	MSE	0.0003	0.0003	0.0004	0.0002	0.0001	0.0002	0.0004	0.0003	0.0002
5000	KL	0.0072	0.0080	0.0062	0.0027	0.0010	0.0026	0.0060	0.0081	0.0076
	MSE	0.0002	0.0003	0.0004	0.0002	0.0001	0.0001	0.0003	0.0002	0.0002

relies on a kernel density estimate representation of the data, which is subsequently translated into an MoTBF-representation. In the limit it can be shown that the learned/translated MoTBF parameters converge to the maximum likelihood parameters.

Figure 5 shows the MoTBFs of the conditional (a) and the posterior (c,d,e) densities approx-

Table 6: Comparison between the true posterior density and the one learned with the MoP approximation obtained using Algorithm 1. Mean KL and MSE for ten datasets sampled from the BN, where $Y \sim \text{Gamma}(rate = 10, shape = 10)$ and $X|Y \sim \text{Exp}(y)$.

N	x_{obs}	0.1063	0.2261	0.3638	0.5247	0.7187	0.9599	1.2817	1.7495	2.5946
25	KL	0.1275	0.1215	0.1157	0.1099	0.1041	0.0988	0.0946	0.0935	0.1123
	MSE	0.1149	0.1135	0.1123	0.1112	0.1100	0.1089	0.1080	0.1083	0.1243
500	KL	0.0125	0.0108	0.0098	0.0096	0.0102	0.0117	0.0139	0.0155	0.0240
	MSE	0.0102	0.0088	0.0078	0.0072	0.0072	0.0081	0.0100	0.0134	0.0243
2500	KL	0.0075	0.0060	0.0047	0.0039	0.0038	0.0046	0.0060	0.0048	0.0081
	MSE	0.0067	0.0054	0.0044	0.0038	0.0037	0.0044	0.0054	0.0048	0.0115
5000	KL	0.0038	0.0031	0.0026	0.0024	0.0024	0.0025	0.0026	0.0031	0.0083
	MSE	0.0044	0.0037	0.0032	0.0029	0.0027	0.0027	0.0028	0.0045	0.0111

Table 7: Comparison between the true posterior density and the one learned with the MoP approximation obtained using Algorithm 3 with Lagrange interpolation. Mean KL and MSE for ten datasets sampled from the BN, where $Y \sim \text{Gamma}(rate = 10, shape = 10)$ and $X|Y \sim \text{Exp}(y)$.

N	x_{obs}	0.1063	0.2261	0.3638	0.5247	0.7187	0.9599	1.2817	1.7495	2.5946
25	KL	0.1368	0.1307	0.1239	0.1164	0.1086	0.1013	0.0962	0.0976	0.1124
	MSE	0.1226	0.1207	0.1185	0.1160	0.1132	0.1106	0.1094	0.1132	0.1215
500	KL	0.0135	0.0119	0.0108	0.0104	0.0111	0.0139	0.0177	0.0172	0.0256
	MSE	0.0078	0.0075	0.0071	0.0068	0.0071	0.0093	0.0133	0.0143	0.0261
2500	KL	0.0079	0.0067	0.0057	0.0049	0.0047	0.0056	0.0073	0.0060	0.0115
	MSE	0.0056	0.0051	0.0047	0.0042	0.0042	0.0050	0.0065	0.0047	0.0143
5000	KL	0.0054	0.0047	0.0038	0.0033	0.0032	0.0036	0.0045	0.0039	0.0103
	MSE	0.0042	0.0039	0.0034	0.0030	0.0029	0.0034	0.0044	0.0048	0.0150

imated using the first data described in Section 3.4. The conditional MoTBF has 6 pieces and each piece defines an MoP with at most six parameters; polynomial basis functions are used in all the experiments. MoTBF approximations of conditional densities are obtained by discretizing the parent variables and fitting a one-dimensional MoTBF for each hyperrectangle defined by the split-points of the parents. Compared with the two learning methods proposed in Algorithms 1 and 3, the method in Langseth *et al.*[11] therefore captures the correlation between the parent variables and the child variable through the hyperrectangles instead of directly in the functional polynomial expressions. The selection of split-points and number of basis functions is guided by a greedy search strategy that optimizes the BIC score of the model by iteratively evaluating the BIC-gain of bisecting an existing candidate hyperrectangle and relearning the number of basis functions.

Table 8: Comparison between the true posterior density and the one learned with the MoP approximation obtained using Algorithm 1 . Mean KL and MSE for ten datasets sampled from the BN, where $Y \sim 0.5\mathcal{N}(-3,1)+0.5\mathcal{N}(3,1)$ and $X|Y \sim \mathcal{N}(y,1)$.

N	x_{obs}	-4.2244	-3.3719	-2.6362	-1.7662	0.0000	1.7662	2.6362	3.3719	4.2244
25	KL	0.3947	0.4395	0.6016	0.8949	1.1297	0.9143	0.6482	0.4991	0.4242
	MSE	0.0088	0.0120	0.0173	0.0239	0.0152	0.0233	0.0173	0.0129	0.0091
500	KL	0.2875	0.2118	0.2311	0.3779	0.6586	0.3962	0.2374	0.2187	0.2882
	MSE	0.0061	0.0048	0.0062	0.0105	0.0092	0.0110	0.0065	0.0053	0.0061
2500	KL	0.0773	0.0638	0.0780	0.0687	0.2389	0.0604	0.0802	0.0735	0.0810
	MSE	0.0015	0.0013	0.0022	0.0018	0.0038	0.0015	0.0022	0.0016	0.0016
5000	KL	0.0185	0.0221	0.0212	0.0693	0.1337	0.0649	0.0245	0.0215	0.0237
	MSE	0.0003	0.0007	0.0007	0.0023	0.0020	0.0021	0.0008	0.0007	0.0004

Table 9: Comparison between the true posterior density and the one learned with the MoP approximation obtained using Algorithm 3 with Lagrange interpolation. Mean KL and MSE for ten datasets sampled from the BN, where $Y \sim 0.5\mathcal{N}(-3,1) + 0.5\mathcal{N}(3,1)$ and $X|Y \sim \mathcal{N}(y,1)$.

N	x_{obs}	-4.2244	-3.3719	-2.6362	-1.7662	0.0000	1.7662	2.6362	3.3719	4.2244
25	KL	0.4322	0.4919	0.6442	0.8876	6.7282	0.9882	0.6942	0.5356	0.4336
	MSE	0.0097	0.0138	0.0205	0.0259	0.2251	0.0245	0.0183	0.0146	0.0100
500	KL	0.2552	0.2185	0.2612	0.3952	0.6440	0.4053	0.2484	0.2187	0.2925
	MSE	0.0049	0.0052	0.0073	0.0109	0.0091	0.0116	0.0072	0.0052	0.0059
2500	KL	0.0712	0.0526	0.0739	0.0895	0.1881	0.0875	0.0802	0.0590	0.0773
	MSE	0.0016	0.0008	0.0019	0.0026	0.0029	0.0025	0.0020	0.0010	0.0017
5000	KL	0.0063	0.0138	0.0110	0.0463	0.0663	0.0408	0.0106	0.0128	0.0080
	MSE	0.0001	0.0004	0.0004	0.0016	0.0009	0.0014	0.0004	0.0005	0.0001

If there is a weak correlation between the child and parent variables, then the conditional MoTBF approach is expected to yield approximations with few pieces. On the other hand, as the variables become more strongly correlated, additional subintervals will be introduced by the learning algorithm. The MoTBF learning algorithm does not rely on a discretization of the child variable, but it rather approximates the density using a higher-order polynomial/exponential function. In contrast, Algorithms 1 and 3 yield conditional MoPs with more pieces because the domain of approximation $\Omega_{X,\mathbf{Y}}$ is split into hyperrectangles in all the dimensions. However, with the finer-grained division of the domain into hyperrectangles, the polynomial functions of the conditional MoPs will usually also have a lower order.

We empirically compared Algorithm 1 and Algorithm 3 (using both TSE and LI) to the method proposed in Langseth *et al.*[11] by employing the greedy search strategy in Section 3.3 and using

Table 10: Results of the Wilcoxon signed-rank test: (black) results for KL, (red) results for MSE.

(a) $Y \sim \mathcal{N}(0,1)$ and $X|Y \sim \mathcal{N}(y,1)$

↓ outperforms →	Alg 1	Alg 3 TSE	Alg 3 LI
Alg 1		2 3	2 2
Alg 3 TSE	5 1		0 0
Alg 3 LI	4 1	0 0	

(b) $Y \sim \text{Gamma}(rate = 10, shape = 10)$ and $X|Y \sim \text{Exp}(y)$

↓ outperforms →	Alg 1	Alg 3 TSE	Alg 3 LI
Alg 1		1 2	6 0
Alg 3 TSE	1 1		1 0
Alg 3 LI	0 0	0 0	

(c) $Y \sim 0.5\mathcal{N}(-3,1) + 0.5\mathcal{N}(3,1)$ and $X|Y \sim \mathcal{N}(y,1)$

↓ outperforms →	Alg 1	Alg 3 TSE	Alg 3 LI
Alg 1		5 4	2 6
Alg 3 TSE	10 10		3 2
Alg 3 LI	11 10	3 3	

the three data sets described in Section 3.4.

Tables 11, 12, and 13 show the mean Kullback-Leibler divergences and MSEs between the MoPs and the true posterior densities $Y|X$ for three values of X in the ten repetitions. We applied a paired Wilcoxon signed-rank test and report statistically significant differences at a significance level $\alpha = 0.05$. The null hypothesis is that the two methods perform similarly. The alternative hypothesis is that the algorithm in the column outperforms the algorithm shown with a symbol: $*$ for Algorithm 1, † for Algorithm 3 with TSE, ‡ for Algorithm 3 with LI, and $\star$ for conditional MoTBFs. For instance, a $\star$ in the column corresponding to Algorithm 1 in Table 11 shows that Algorithm 1 significantly outperforms MoTBFs for the corresponding values for N and X. From the Gamma-Exponential distribution (Table 12) we see that the models produced by Algorithms 1 and 3 are generally comparable to or slightly worse than those learned using the MoTBF-based method. However, when considering Table 11 we see that Algorithm 3 significantly outperforms the MoTBF-based method, especially for the larger data sets. When further analyzing the models learned for the data sets with 5000 observations, we find that the learned MoTBF models contain at most six pieces each holding an MoP with at most six parameters (hence a total of 36 parameters, not counting the parameters defining the pieces). In comparison, Algorithm 3 produce models with 256 parameters(16 pieces each holding a polynomial of degree 3 in each variable) and Algorithm 1 outputs models with 49 parameters($7 = 4 + 4 - 1$ parameters for each dimension). Thus, for these

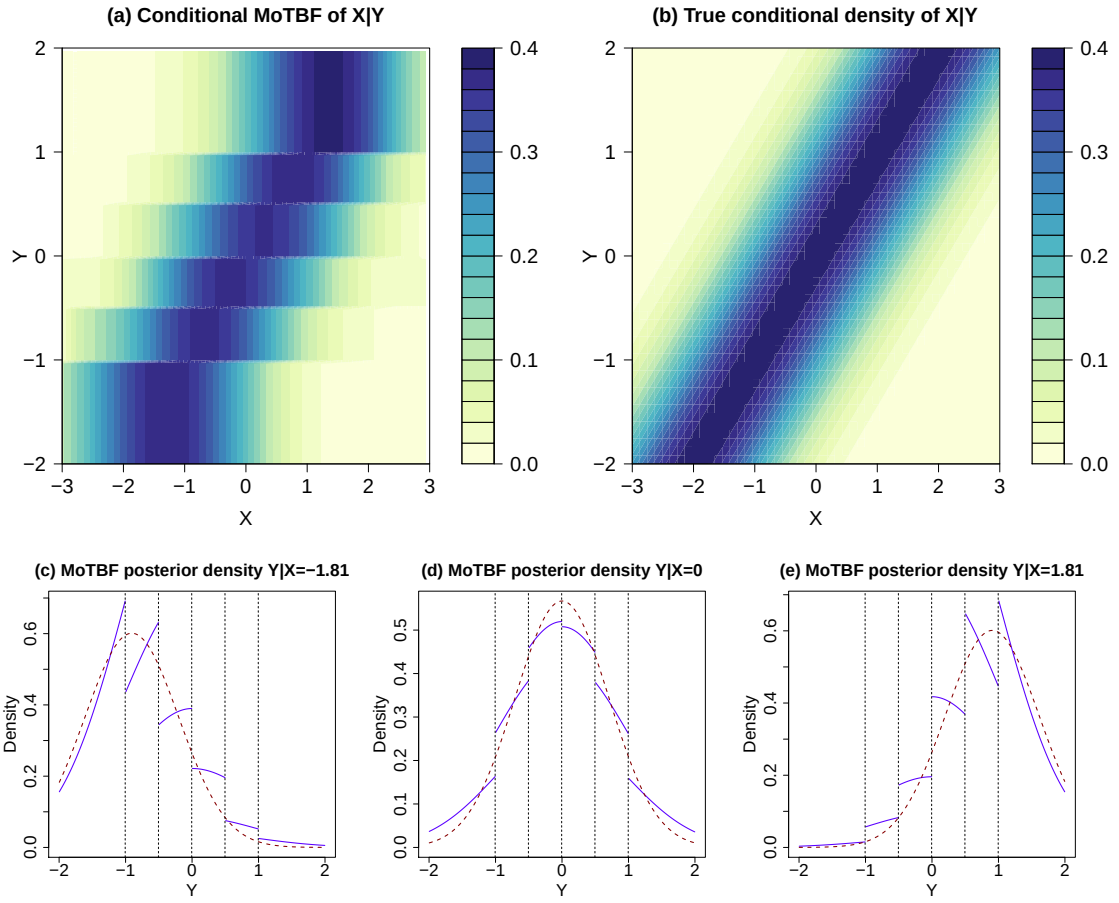


Figure 5: Example of $Y \sim \mathcal{N}(0,1)$ and $X|Y \sim \mathcal{N}(y,1)$, case $N = 5000$. (a) Conditional MoTBF of $X|Y$ learned with the approach in[11]. (b) True conditional density of $X|Y \sim \mathcal{N}(y,1)$. (c,d,e) MoTBF approximations (solid) and true posterior densities (dashed) of $Y|X$ for three values of X.

data sets the proposed learning algorithms seem to allow more complex models to be learned than when using the MoTBF approach. With respect to the mixture model (Table 13) we observe that the proposed algorithms outperform the MoTBF-based method both in the small data sets (Algorithm 1), both in the larger data sets (Algorithm 3).

5. A Real-World Example in Neuroanatomy

As a real-world example we build a MoP model over some variables describing neurons by their morphological features. We use the database studied in Guerra *et al.*[24], which addresses the problem of classifying a neuron based on its morphological features. The database is made up of 327 observations concerning 52 variables describing morphological and spatial neuron characteristics. We select the variable *relative distance to pia* as the parent variable Y, and the variable *area of the dendrite's convex hull* as the child X. The relative distance to pia is the ratio of the straight-line

Table 11: Mean Kullback-Leibler divergences and MSE between the approximations and the true posterior densities for ten datasets sampled from the BN, where $Y \sim \mathcal{N}(0,1)$ and $X|Y \sim \mathcal{N}(y,1)$. The best results for each sample size are highlighted in bold. Statistically significant differences at $\alpha = 0.05$ are shown with symbols $*, \dagger, \ddagger, \star$.

		KL				MSE			
N	$Y\|X=x$	Alg. 1 ($*$)	Alg. 3 TSE ($\dagger$)	Alg. 3 LI ($\ddagger$)	MoTBF ($\star$)	Alg. 1 ($*$)	Alg. 3 TSE ($\dagger$)	Alg. 3 LI ($\ddagger$)	MoTBF ($\star$)
	X =-1.81	0.3312 $\star$	0.3275 $\star$	**0.3199** $\star$	0.6139	0.0152 $\star$	0.0155 $\star$	**0.0147** $\star$	0.0598
25	X =0.00	0.2550	0.2592	0.2584	**0.0553** $* \ddagger$	0.0130	0.0131	0.0131	**0.0048** $* \dagger \ddagger$
	X =1.81	0.3038 $\star$	**0.2895** $\star$	0.2937 $\star$	0.6349	0.0092 $\star$	**0.0090** $\star$	**0.0090** $\star$	0.0608
	X =-1.81	0.0612 $\star$	**0.0569** $\star$	0.0586 $\star$	0.1588	0.0013 $\star$	**0.0012** $\star$	0.0013 $\star$	0.0174
500	X =0.00	0.0099 $\star$	0.0086 $\star$	**0.0084** $\star$	0.0666	0.0004 $\star$	**0.0003** $\star$	**0.0003** $\star$	0.0047
	X =1.81	0.0634 $\star$	**0.0567** $\star$	0.0596 $\star$	0.1540	0.0014 $\star$	**0.0012** $\star$	**0.0012** $\star$	0.0161
	X =-1.81	0.0115 $\star$	0.0099 $\star$	**0.0098** $\star$	0.0731	0.0004 $\star$	**0.0003** $\star$	**0.0003** $\star$	0.0078
2500	X =0.00	0.0017 $\star$	0.0015 $\star$	**0.0012** $\star$	0.0273	0.0001 $\star$	**0.0001** $\star$	**0.0001** $\star$	0.0016
	X =1.81	0.0110 $\star$	**0.0085** $* \star$	0.0087 $\star$	0.0596	**0.0002** $\star$	**0.0002** $\star$	**0.0002** $\star$	0.0058
	X =-1.81	0.0093 $\star$	0.0075 $\star$	**0.0072** $\star$	0.1110	**0.0002** $\star$	**0.0002** $\star$	**0.0002** $\star$	0.0098
5000	X =0.00	0.0011 $\star$	**0.0010** $\star$	**0.0010** $\star$	0.0301	**0.0001** $\star$	**0.0001** $\star$	**0.0001** $\star$	0.0017
	X =1.81	0.0102 $\star$	0.0080 $* \star$	**0.0076** $* \star$	0.1055	0.0003 $\star$	0.0003 $\star$	**0.0002** $* \star$	0.0086

Table 12: Mean Kullback-Leibler divergences and MSE between the approximations and the true posterior densities for ten datasets sampled from the BN, where $Y \sim \text{Gamma}(rate = 10, shape = 10)$ and $X|Y \sim \text{Exp}(y)$. The best results for each sample size are highlighted in bold. Statistically significant differences at $\alpha = 0.05$ are shown with symbols $*, \dagger, \ddagger, \star$.

		KL				MSE			
N	$Y\|X=x$	Alg. 1 ($*$)	Alg. 3 TSE ($\dagger$)	Alg. 3 LI ($\ddagger$)	MoTBF ($\star$)	Alg. 1 ($*$)	Alg. 3 TSE ($\dagger$)	Alg. 3 LI ($\ddagger$)	MoTBF ($\star$)
	X =0.1063	0.1275	0.1370	0.1368	0.0078 $* \dagger \ddagger$	0.1149	0.1225	0.1226	**0.0155** $* \dagger \ddagger$
25	X =0.7187	**0.1041**	0.1083	0.1086	0.1048	**0.1100**	0.1142	0.1132	0.3302
	X =2.5946	0.1123	0.1097	0.1124	**0.0866** $* \dagger \ddagger$	0.1243	**0.1188**	0.1215	0.1746 $*$
	X =0.1063	0.0125	0.0121	0.0135	**0.0048** $* \dagger \ddagger$	0.0102	**0.0078**	**0.0078**	0.0080
500	X =0.7187	0.0102	0.0099	0.0111	**0.0001** $* \dagger \ddagger$	0.0072	0.0068	0.0071	**0.0010** $* \dagger \ddagger$
	X =2.5946	0.0240 $\star$	**0.0193** $* \star$	0.0256 $\ddagger \star$	0.0706	0.0243 $\star$	**0.0187** $* \ddagger \star$	0.0261 $\star$	0.1144
	X =0.1063	0.0075	0.0071	0.0079	**0.0039** $* \dagger \ddagger$	0.0067	**0.0054**	0.0056	0.0074
2500	X =0.7187	0.0038 $\ddagger$	0.0041	0.0047	**0.0001** $* \dagger \ddagger$	0.0037	0.0040	0.0042	**0.0002** $* \dagger \ddagger$
	X =2.5946	**0.0081** $\star$	0.0092 $\star$	0.0115 $\star$	0.0602	**0.0115** $\star$	0.0128 $\star$	0.0143 $\star$	0.1077
	X =0.1063	**0.0038**	0.0047	0.0054	**0.0038**	0.0044 $\star$	**0.0041** $\star$	0.0042 $\star$	0.0073
5000	X =0.7187	0.0024 $\ddagger$	0.0029	0.0032	**0.0001** $* \dagger \ddagger$	0.0027	0.0032	0.0029	**0.0002** $* \dagger \ddagger$
	X =2.5946	**0.0083** $\star$	0.0084 $\star$	0.0103 $\star$	0.0585	**0.0111** $\ddagger \star$	0.0134 $\star$	0.0150 $\star$	0.1078

distance from soma to pia and the straight-line distance from white matter to pia. Thus, a value close to 0 (resp. 1) corresponds to a soma in a superficial (resp. deep) layer. Convex hull analysis draws a two-dimensional convex shape around the dendrites. The area (μm^2) of this shape is then calculated. Before applying our MoP approximations to $X|Y$, the data are divided by their sample standard deviation. Also, only 96% of the central values of the transformed data have been maintained; the remaining values have been discarded.

Since the dataset considered is quite *small* and continuous densities are desirable for this particular domain, we apply Algorithm 1 for learning the MoP representations, cf. the discussion in subsection 3.4. The results are shown in Figure 6.

The conditional MoP of $X|Y$ on the top figure shows that for small values of the distance to pia the dendrite areas are mostly concentrated around small values, whereas for larger distances the areas spread over more values, i.e., dendrite areas present a higher dispersion when the neurons

Table 13: Mean Kullback-Leibler divergences and MSE between the approximations and the true posterior densities for ten datasets sampled from the BN, where $Y \sim 0.5\mathcal{N}(-3,1) + 0.5\mathcal{N}(3,1)$ and $X|Y \sim \mathcal{N}(y,1)$. The best results for each sample size are highlighted in bold. Statistically significant differences at $\alpha = 0.05$ are shown with symbols $*, \dagger, \ddagger, \star$.

		KL				MSE			
N	$Y\|X=x$	Alg. 1 ($*$)	Alg. 3 TSE ($\dagger$)	Alg. 3 LI ($\ddagger$)	MoTBF ($\star$)	Alg. 1 ($*$)	Alg. 3 TSE ($\dagger$)	Alg. 3 LI ($\ddagger$)	MoTBF ($\star$)
25	X =-4.22	**0.3947** $\star$ $\ddagger$	0.4163 $\star$ $\ddagger$	0.4322 $\star$	0.6859	**0.0088** $\star$	0.0096 $\star$ $\ddagger$	0.0097 $\star$	0.0163
	X =0.00	**1.1297** $\star$	3.5644	6.7282	2.1171	**0.0152** $\star$	0.0191	0.2251	0.0258
	X =4.22	**0.4242** $\star$	0.4541 $\star$	0.4336 $\star$	0.5720	**0.0091** $\star$	0.0103 $\star$	0.0100 $\star$	0.0142
500	X =-4.22	0.2875	0.2384	0.2552	**0.2212**	0.0061	0.0056 $\star$	**0.0049** $\star$	0.0079
	X =0.00	0.6586	1.1011	0.6440	0.6904	0.0092 $\star$	0.0127	**0.0091** $\star$	0.0120
	X =4.22	0.2882	0.2423	0.2925	**0.2134**	0.0061	**0.0056** $\star$	0.0059	0.0079
2500	X =-4.22	0.0773	**0.0692**$\star$	0.0712 $\star$	0.0843	**0.0015** $\star$	**0.0015** $\star$	0.0016 $\star$	0.0029
	X =0.00	0.2389 $\star$	0.3264	**0.1881** $\star$ $*$	0.4593	0.0038 $\star$	0.0036 $\star$	**0.0029** $\star$ $*$ $\dagger$	0.0102
	X =4.22	0.0810	**0.0763**	0.0773	0.0961	**0.0016** $\star$	0.0017 $\star$	0.0017 $\star$	0.0035
5000	X =-4.22	0.0185 $\star$	0.0066 $\star$ $*$	**0.0063** $\star$ $*$	0.0618	0.0003 $\star$	**0.0001** $\star$ $*$	**0.0001** $\star$ $*$ $\dagger$	0.0021
	X =0.00	0.1337 $\star$	0.0668 $\star$ $*$	**0.0663** $\star$ $*$	0.3331	0.0020 $\star$	**0.0009** $\star$ $*$	**0.0009** $\star$ $*$	0.0081
	X =4.22	0.0237 $\star$	0.0090 $\star$ $*$	**0.0080** $\star$ $*$	0.0489	0.0004 $\star$	0.0002 $\star$ $*$	**0.0001** $\star$ $*$	0.0012

are further away from the pia. This MoP has $L_X = 4$ and $L_Y = 2$ pieces for X and Y, respectively, each one with order 3. For the posterior distributions $Y|X$ in the bottom figures, for area $x = 0.38$ (left) the distance to pia is asymmetrically distributed with a mode close to 1, whereas for $x = 1.50$ (right) the density is rather symmetric with a mode close to 2.

6. Conclusion

In this paper, we have considered two methods for learning MoP approximations of conditional densities $X|\mathbf{Y}$ from data. The initializing step in both methods involves estimating the joint density $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})$ and the marginal density of the parents $\varphi_{\mathbf{Y}}(\mathbf{y})$. In the first method, we use the two learned densities to obtain a sample from the quotient $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$ based on which an unnormalized conditional MoP is learned. Proper normalization of the learned MoP is not feasible since the resulting potential would be outside the MoP model class, hence we instead resort to a partial normalization. Although the models obtained from the partial normalization can provide good accuracy results, it is difficult to control the quality of the approximation. This shortcoming has motivated the second learning algorithm, where a conditional MoP is obtained using multidimensional interpolation based on the quotient $\varphi_{X,\mathbf{Y}}(x,\mathbf{y})/\varphi_{\mathbf{Y}}(\mathbf{y})$ obtained from the initial step of the algorithm; for the actual estimation we have considered multidimensional Taylor series expansion and Lagrange interpolation.

The proposed methods have been empirically analyzed and evaluated using data sampled from three different statistical models, one corresponding to a two-dimensional Gaussian distribution an other involving an exponentially distributed variable with a rate parameter following a Gamma distribution, and lastly a model with a Gaussian distribution with mean parameter following a mixture of two Gaussian distributions. From the experimental results we have observed that

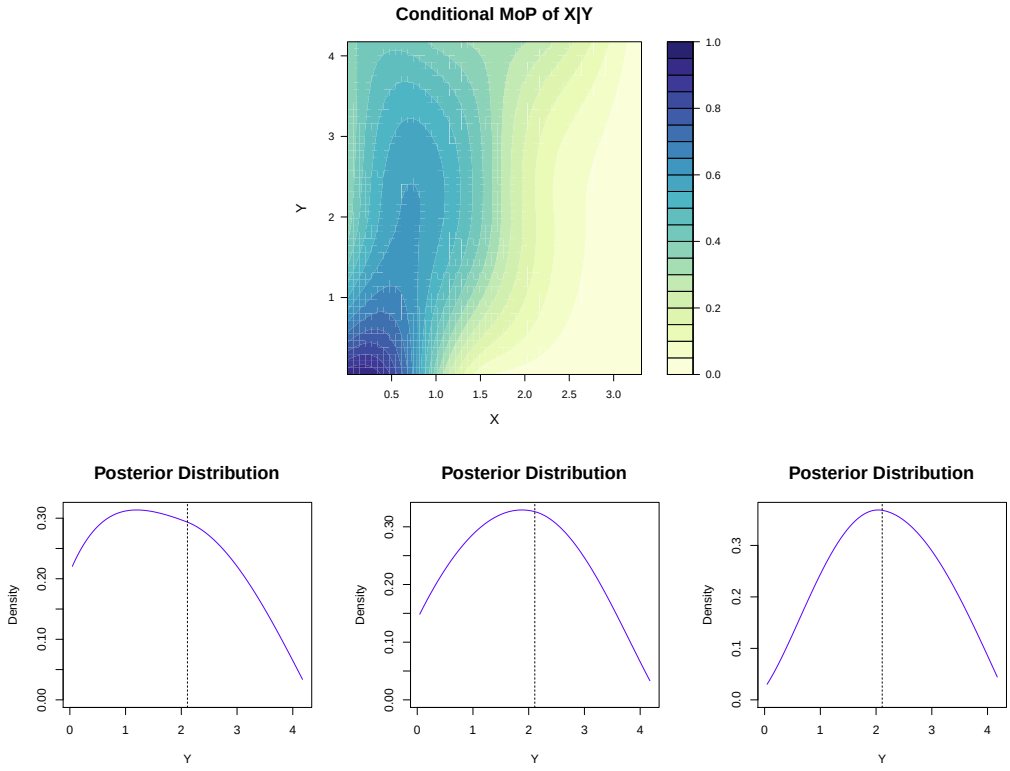


Figure 6: Conditional density (top) and posterior densities ($x = 0.38$ left; $x = 0.79$ middle; $x = 1.50$ right) learned with Algorithm 1 in neuronal morphological data.

both methods yield good approximations (low KL and MSE values) of the true conditional densities. The observations from these studies were supplemented with an analysis of a real-world neuroanatomy dataset. For comparison, we have analyzed the proposed methods relative to the MoTBF learning method described by Langseth *et al.*[11,12] using the previously generated artificial datasets. From the results we observed that although the three methods yield comparable results for the Gamma-Exponential distributed data, we also found that the proposed algorithms significantly outperformed the MoTBF-based algorithm on the Gaussian data sets and on the mixture-model datasets.

In this paper, equal-width intervals $[\epsilon_i, \xi_i]$ are assumed in each dimension, and the hyperrectangles A_l have the same size. In the future, we would like to further study how to automatically find appropriate values for the limits $[\epsilon_i, \xi_i]$. For a given configuration of the model parameters, the computational complexity is dominated by the algorithm for learning the joint and the marginal densities. We would like to investigate methods for improving the computational complexity of this particular step of the algorithm as well as methods for improving the overall runtime of the algorithm. Finally, we intend to use these approaches to learn more complex BNs, which also involves adapting the learned potentials to support efficient inference and considering BN structure

learning.

Acknowledgments. This work has been partially supported by the Spanish Ministry of Economy and Competitiveness through Cajal Blue Brain (C080020-09) and TIN2010-20900-C04-04 projects.

References

References

[1] P. P. Shenoy and J. C. West. Inference in hybrid Bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning*, 52(5):641–657, 2011.

[2] P. P. Shenoy. Two issues in using mixtures of polynomials for inference in hybrid Bayesian networks. *International Journal of Approximate Reasoning*, 53(5):847–866, 2012.

[3] H. Langseth, T. D. Nielsen, R. Rumí, and A. Salmerón. Mixtures of truncated basis functions. *International Journal of Approximate Reasoning*, 53:212–227, 2012.

[4] S. Moral, R. Rumí, and A. Salmerón. Mixtures of truncated exponentials in hybrid Bayesian networks. Lecture Notes in Computer Science 2143, pages 145–167. Springer, 2001.

[5] G.R. Shafer and P.P. Shenoy. Probability Propagation. *Annals of Mathematics and Artificial Intelligence*, 2:327–352, 1990.

[6] Barry R. Cobb and Prakash P. Shenoy. Inference in hybrid Bayesian networks with mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 41(3):257–286, April 2006.

[7] B. Cobb, P.P. Shenoy, and R. Rumí. Approximating probability density functions with mixtures of truncated exponentials. *Statistics and Computing*, 16:293–308, 2006.

[8] S. Moral, R. Rumí, and A. Salmerón. Estimating mixtures of truncated exponentials from data. In *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM'02)*, pages 135–143, 2002.

[9] V. Romero, R. Rumí, and A. Salmerón. Learning hybrid Bayesian networks using mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 42:54–68, 2006.

[10] H. Langseth, T.D. Nielsen, R. Rumí, and A. Salmerón. Parameter estimation and model selection for mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, 51:485–498, 2010.

[11] H. Langseth, T. D. Nielsen, R. Rumí, and A. Salmerón. Learning mixtures of truncated basis functions from data. In *Proceedings of the 6th European Workshop on Probabilistic Graphical Models*, pages 163–170, 2012.

[12] H. Langseth, T.D. Nielsen, I. Pérez-Bernabé, and A. Salmerón. Learning mixtures of truncated basis functions from data. *International Journal of Approximate Reasoning*, 2013.

[13] P.L. López-Cruz, C. Bielza, and P. Larrañaga. Learning mixtures of polynomials of multidimensional probability densities from data using B-spline interpolation. *International Journal of Approximate Reasoning*, 55(4):989–1010, 2014.

[14] H. Langseth, T.D. Nielsen, R. Rumí, and A. Salmerón. Maximum likelihood learning of conditional MTE distributions. Lecture Notes in Computer Science 5590, pages 240–251. Springer, 2009.

[15] P.L. López-Cruz, T.D. Nielsen, C. Bielza, and P. Larrañaga. Learning mixtures of polynomials of conditional densities from data. In *Lecture Notes in Artificial Intelligence 8109*, pages 363–372, 2013.

[16] I. J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. Part A: On the problem of smoothing of graduation. A first class of analytic approximation formulae. *Quarterly of Applied Mathematics*, 4:45–99, 1946.

[17] I.D. Faux and M.J. Pratt. *Computational Geometry for Design and Manufacture*. Wiley, 1979.

[18] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-Spline Techniques*. Springer, 2002.

[19] Z. Zong. *Information-Theoretic Methods for Estimating Complicated Probability Distributions*. Elsevier, 2006.

[20] Z. Zong and K.Y. Lam. Estimation of complicated distributions using B-spline functions. *Structural Safety*, 20(4):341–355, 1998.

[21] L.A. Harris. Bivariate Lagrange interpolation at the Chebyshev nodes. *Proceedings of the American Mathematical Society*, 138(12):4447–4453, 2010.

[22] M. Caliari, S. De Marchi, A. Sommariva, and M. Vianello. Padua2DM: Fast interpolation and cubature at the Padua points in Matlab/Octave. *Numerical Algorithms*, 56(1):45–60, 2011.

[23] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[24] L. Guerra, L.M. McGarry, V. Robles, C. Bielza, P. Larrañaga, and R. Yuste. Comparison between supervised and unsupervised classification of neuronal cell types: A case study. *Developmental Neurobiology*, 71(1):71–82, 2011.