International Journal of Approximate Reasoning 55 (2014) 989–1010

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Learning mixtures of polynomials of multidimensional probability densities from data using B-spline interpolation

Pedro L. López-Cruz[*], Concha Bielza, Pedro Larrañaga

Computational Intelligence Group, Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo s/n, 28660 Boadilla del Monte, Madrid, Spain

ARTICLE INFO

Article history:
Available online 1 October 2013

Keywords:
Mixtures of polynomials
Interpolation
Density estimation
Bayesian classifiers

ABSTRACT

Non-parametric density estimation is an important technique in probabilistic modeling and reasoning with uncertainty. We present a method for learning mixtures of polynomials (MoPs) approximations of one-dimensional and multidimensional probability densities from data. The method is based on basis spline interpolation, where a density is approximated as a linear combination of basis splines. We compute maximum likelihood estimators of the mixing coefficients of the linear combination. The Bayesian information criterion is used as the score function to select the order of the polynomials and the number of pieces of the MoP. The method is evaluated in two ways. First, we test the approximation fitting. We sample artificial datasets from known one-dimensional and multidimensional densities and learn MoP approximations from the datasets. The quality of the approximations is analyzed according to different criteria, and the new proposal is compared with MoPs learned with Lagrange interpolation and mixtures of truncated basis functions. Second, the proposed method is used as a non-parametric density estimation technique in Bayesian classifiers. Two of the most widely studied Bayesian classifiers, i.e., the naive Bayes and tree-augmented naive Bayes classifiers, are implemented and compared. Results on real datasets show that the non-parametric Bayesian classifiers using MoPs are comparable to the kernel density-based Bayesian classifiers. We provide a free R package implementing the proposed methods.

1. Introduction

In real life problems, continuous data may not fit any standard parametric distribution. Therefore, the assumption of a parametric shape might yield misleading conclusions or results. Non-parametric density estimation is used to avoid the parametric assumptions in probabilistic modeling and reasoning [1,2]. Non-parametric estimation techniques can be classified in four categories [3]: histograms, orthogonal series, kernels and splines. Histograms are based on transforming the continuous data into discrete data. Discretization is one of the most widely used approaches for data transformation, and a large number of discretization techniques have been proposed in the literature [4,5]. However, important information can be lost during the discretization process. A different approach approximates probability densities by an orthogonal series expansion using, e.g., Hermite, Fourier or trigonometric orthonormal systems of functions. Their main drawback is that the resulting estimate is not a proper density (non-negative and integrating to one). Recently, kernel-based density estimation has received a lot of attention because it provides a flexible and powerful tool for non-parametric density estimation. However, kernel density estimation has to save and analyze the complete training dataset to evaluate the density of each data point. Also, bandwidth selection for kernel density estimation can be challenging and a lot of different approaches have

[*] Corresponding author.
E-mail addresses: pedro.lcruz@upm.es (P.L. López-Cruz), mcbielza@fi.upm.es (C. Bielza), pedro.larranaga@fi.upm.es (P. Larrañaga).

0888-613X/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ijar.2013.09.018

been proposed for finding bandwidth values that accurately model the data without overfitting [6,7]. Finally, splines are piecewise polynomial curves frequently used for approximating arbitrary functions.

Here, we focus on a family of related models including mixtures of truncated exponentials (MTEs) [8], mixtures of polynomials (MoPs) [9] and mixtures of truncated basis functions (MoTBFs) [10]. Given a continuous random variable X with a probability density function $f_X(x)$, the goal is to find an approximation $\varphi_X(x)$ of $f_X(x)$ over a closed domain $\Omega_X \subset \mathbb{R}$. MTEs approximate $f_X(x)$ by a linear combination of exponential functions [8]. Additionally, the domain of approximation Ω_X can be divided into subintervals, and $f_X(x)$ is approximated (piecewise) in each subinterval using a linear combination of exponential functions. Similarly, MoPs use polynomial functions instead of exponential functions to approximate $f_X(x)$ in each subinterval of Ω_X [9]. Recently, MoTBFs [10] have been proposed as a framework for representing hybrid Bayesian networks with continuous and discrete variables. MoTBFs approximate $f_X(x)$ as a (piecewise) linear combination of truncated basis functions. MTEs and MoPs are particular scenarios of MoTBFs when using exponential or polynomial functions, respectively, as basis functions. MoTBFs, MTEs and MoPs are closed under multiplication, addition and integration. Therefore, exact probabilistic inference can be performed using the Shenoy–Shafer architecture [11].

Different methods have been proposed for approximating with MTEs. Cobb et al. [12] provided MTE approximations of seven standard parametric probability density functions. Rumí et al. [13] proposed an iterative least squares algorithm for learning MTE approximations of one-dimensional and conditional probability densities from data. This approach used exponential regression and empirical histograms for density estimation. Romero et al. [14] enhanced the algorithm by applying a kernel smoothing of the probability densities obtained with the empirical histograms. Langseth et al. [15] provided a maximum likelihood (ML) estimation approach for MTE approximations. Polynomial approximation and interpolation techniques have been used to obtain MoPs. Shenoy and West [9] found MoP approximations of known parametric densities by computing their Taylor series expansions. Later, Lagrange interpolating polynomials (LIPs) were used to obtain MoPs [16]. Regarding MoTBFs, Langseth et al. [10] proposed a method for finding approximations of one-dimensional and conditional densities by minimizing the Kullback–Leibler divergence from the MoTBF to the true distribution. Recently, the Kullback–Leibler approach was combined with kernel density estimation techniques to approximate MoTBFs from data [17].

In this paper, we present a method for learning MoP approximations of one-dimensional and multidimensional probability densities directly from data. Langseth et al. [18] have shown that only univariate and conditional MoTBFs appear during inference. However, finding MoP approximations of conditional densities is challenging. Polynomials are not closed under division. Therefore, given two multidimensional random variables $\mathbf{X}$ and $\mathbf{Y}$, it is not possible to obtain a MoP approximation of the conditional density of $\mathbf{X}|\mathbf{Y}$ by dividing the MoP approximations of the joint density of $(\mathbf{X}, \mathbf{Y})$ and the marginal density of $\mathbf{Y}$. The standard approach for learning approximations of conditional densities in the MoTBF framework involves discretizing the conditioning variables $\mathbf{Y}$. Then, an approximation of the marginal density of $\mathbf{X}$ is found for each combination of the (discrete) values of the conditioning variables $\mathbf{Y}$ [10,15,19]. Therefore, the correlation between $\mathbf{X}$ and $\mathbf{Y}$ is captured by the discretization procedure, instead of directly in the functional forms of the MoTBFs. However, some information may be lost during this discretization procedure. Here, we focus on finding MoP approximations of one-dimensional and multidimensional probability densities from data instead. Non-parametric density estimation for multidimensional data is an interesting topic, e.g., a number of multidimensional kernel-based approaches have been proposed [20–23]. These MoP approximations of multidimensional densities could be used inside probabilistic models which do not require conditional densities, e.g., compositional models defined in the valuation-based systems framework [24,25]. Additionally, even though explicit MoP approximations of conditional densities cannot be obtained, it is still possible to compute the value of the conditional density of $\mathbf{X}|\mathbf{Y}$ for two given values $\mathbf{x}$ and $\mathbf{y}$. This conditional density value is computed by dividing the evaluation of the MoP of joint density of $(\mathbf{X}, \mathbf{Y})$ for values $(\mathbf{x}, \mathbf{y})$ and the evaluation of the MoP of the marginal density of $\mathbf{Y}$ for value $\mathbf{y}$.

The proposed method is based on a probability density estimation technique [26,27] that uses basis spline (B-spline) interpolation [28]. B-splines have some good properties for probability density estimation. B-splines form a basis in the space of piecewise polynomial functions, and the Stone–Weierstrass theorem proves that any continuous function can be approximated arbitrarily well in a closed domain using a polynomial function. Also, B-splines are continuous, differentiable and non-negative. B-splines have been used to find approximations of one-dimensional and two-dimensional probability densities underlying a dataset [27]. Here, we extend the method to n-dimensional MoP approximations of probability densities. The method ensures that the MoP approximation is a valid probability density function, i.e., it is non-negative and integrates to one. Additionally, the proposed method yields continuous MoP approximations. Previous proposals for learning MoPs assume that the mathematical expression of the generating parametric density is known [9] or apply some interpolation technique using the true densities of a set of points [16]. In real settings, this information might not be available because the data may not fit any standard parametric distribution or the true density at the interpolation points may not be known. On the contrary, the proposed method learns the MoP approximations directly from data without assuming any prior knowledge. Additionally, the proposed method is able to fit both one-dimensional and multidimensional joint probability densities. To the best of our knowledge, multidimensional joint probability density estimation has not been attempted with MoPs, MTEs or MoTBFs before. The proposed methodology comprises three steps: First, we use Zong's B-spline interpolation method [27] to approximate the probability density underlying the data. Second, we develop the approximated B-splines into a MoP function. Third, we use the Bayesian information criterion (BIC) score for model selection. In this way, we can select the order of the polynomials and the number of pieces of the MoP in a principled way, avoiding overfitting and highly complex models. We have implemented a free `R` package including the proposed methods.

Finally, we illustrate the use of the proposed methods as a non-parametric density estimation technique for supervised classification problems. Supervised classification [29] studies the problem of assigning a class label to an object based on a set of features that characterize the object. Bayesian classifiers [30] are a kind of Bayesian networks specially designed to solve supervised classification problems. Bayesian classifiers have traditionally dealt with problems where the predictive features are discrete. For hybrid Bayesian classifiers with both continuous and discrete predictive features, the conditional linear Gaussian (CLG) network model [31] has been studied in a number of works, e.g., see [32] for a recent study. However, the Gaussianity assumption and the structural constraints of CLG-based networks can limit the applicability and the performance of these classifiers in real world problems. In these scenarios, discretization approaches can yield better results than CLG-based Bayesian classifiers [33]. Some theoretical analysis of this phenomenon has been given in [34,35]. Kernel density estimation has been proposed as a non-parametric alternative to avoid discretization and the assumptions of CLG-based Bayesian classifiers [36,37]. Also, Bayesian classifiers using MTEs have been studied in [38,39]. Here, we use the proposed methods for learning one-dimensional and multidimensional MoPs from data as a non-parametric density estimation technique in supervised classification problems for the first time. Two of the best-known Bayesian classifier models, i.e., the naive Bayes (NB) classifier [40] and the tree-augmented naive Bayes (TAN) classifier [30], are analyzed.

The remainder of the paper is organized as follows. Section 2 reviews MoPs and the MoP learning methods reported in the literature. Section 3 details the proposed method for learning MoP approximations of probability densities from data. Section 4 reports the experimental evaluation of the proposed methods for learning MoPs and their use for probability density estimation in Bayesian classifiers. Finally, Section 5 ends with conclusions and future work.

2. Mixtures of polynomials

Let X be a one-dimensional continuous random variable with probability density $f_X(x)$. Shenoy and West [9] 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

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

where $p_{l_X}(x) = b_{0,l_X} + b_{1,l_X}x + b_{2,l_X}x^2 + \cdots + b_{d_X,l_X}x^{d_X}$ 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 with $\Omega_X = \bigcup_{l_X=1}^{L_X} A_{l_X}$, $A_i \cap A_j = \emptyset, i \neq j$.

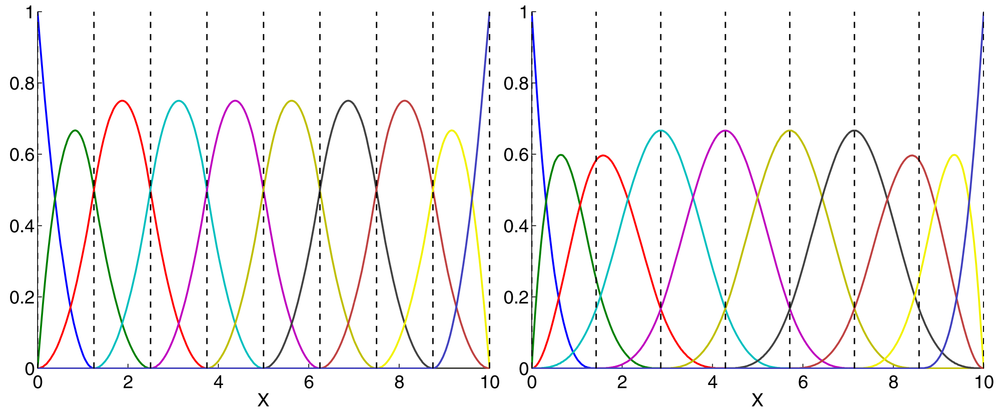
Given a vector of n random variables $\mathbf{X} = (X_1, \ldots, X_n)$ and an approximation domain $\Omega_{\mathbf{X}} = \Omega_{X_1} \times \cdots \times \Omega_{X_n}$, Shenoy and West [9] defined an n-dimensional MoP as the product of one-dimensional MoPs as defined in (1):

$$\varphi_{\mathbf{X}}(\mathbf{x}) = \prod_{i=1}^{n} \varphi_{X_i}(x_i). \tag{2}$$

MoPs are closed under multiplication, integration, differentiation and addition. Therefore, the Shenoy–Shafer algorithm [11] can be used to perform exact inference in the associated hybrid Bayesian network. Shenoy and West [9] found MoP approximations of parametric probability density functions by computing the Taylor series expansion around the middle point of each subinterval A_{l_X} in the MoP. The mathematical expression of the probability density $f_X(x)$ needs to be known for computing the Taylor series expansion. However, real data might not fit any known parametric density, so the Taylor series expansion cannot be used in practice. Also, Taylor series expansion cannot ensure that MoP approximations are valid densities, i.e., they are continuous, non-negative and integrate to one. Later, Shenoy [16] proposed estimating $p_{l_X}(x)$ as the LIP over the Chebyshev points defined in A_{l_X}. However, the true probability densities of the Chebyshev points in each A_{l_X} need to be known or estimated beforehand (e.g., using empirical histograms or kernel density estimation techniques). Lagrange interpolation can ensure non-negativity by increasing the order of the polynomials, and continuity by putting interpolation points at the limits of the intervals. However, it cannot ensure that the resulting MoP integrates to one.

3. Learning MoPs using B-spline interpolation

In this section we detail a method for learning MoP approximations of one-dimensional and multidimensional probability densities from data. The proposal does not assume any prior knowledge about the true density underlying the data, as opposed to previously proposed methods such as the Taylor series or the Lagrange interpolation. Also, it ensures that the resulting MoP approximation is continuous, non-negative and integrates to one. Finally, it provides maximum likelihood estimators of some of the parameters in the approximation. The BIC is used as a score for finding appropriate values for the other parameters in a principled way. Section 3.1 introduces B-spline interpolation. Section 3.2 studies MoP approximations of one-dimensional probability densities. Section 3.3 extends the proposed approach to multidimensional densities. Section 3.4 addresses the model selection problem for finding appropriate values for the order and the number of intervals of a MoP.



(a) B-splines with $r_X = 3$ (b) B-splines with $r_X = 4$

Fig. 1. Ten uniform B-splines defined in the domain $\Omega_X = [0, 10]$. Each B-spline is shown in a different color. The vertical dashed lines show the knot sequence $\boldsymbol{\delta} = (a_0, \ldots, a_{L_X})$, where $L_X = M_X - r_X + 1$ and $M_X = 10$ B-splines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.1. B-spline interpolation

B-splines or basis splines [28] are polynomial curves that form a basis for the space of piecewise polynomial functions over a closed domain $\Omega_X = [\epsilon_X, \xi_X] \subset \mathbb{R}$ [41]. Therefore, any piecewise polynomial function can be written as a linear combination of B-splines. Zong [27] proposed a method for finding B-spline approximations of one-dimensional and two-dimensional probability density functions from data.

Given a non-decreasing knot sequence 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 written as

$$B^{r_X}_{X,j_X}(x) = (a_{j_X} - a_{j_X - r_X}) H(x - a_{j_X - r_X}) \sum_{t=0}^{r_X} \frac{(a_{j_X - r_X + t} - x)^{r_X - 1} H(a_{j_X - r_X + t} - x)}{w'_{j_X - r_X}(a_{j_X - r_X + t})}, \quad x \in \Omega_X, \tag{3}$$

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

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

A recursive definition of B-splines and an efficient and well conditioned evaluation algorithm are available, e.g., in [42]. B-splines have a number of interesting properties for approximating probability densities, e.g., $B^{r_X}_{X,j_X}(x)$ is right-side continuous, differentiable, positive in and zero outside $(a_{j_X}, a_{j_X - r_X})$ [43]. B-splines form a basis in the space of piecewise polynomials and MoPs are piecewise polynomials. Therefore, every MoP can be written as a linear combination of B-splines. Also, given a continuous function $f_X(x)$ defined in a closed domain $\Omega_X = [\epsilon_X, \xi_X] \subset \mathbb{R}$, the Stone–Weierstrass approximation theorem [44] states that there is a polynomial function $pol_X(x)$ that uniformly converges to $f_X(x)$ with an error less than ζ, i.e., there is a polynomial function $pol_X(x)$ so that $\sup_{x \in \Omega_X} |f_X(x) - pol_X(x)| < \zeta$.

When the points in the knot sequence $\boldsymbol{\delta}_X$ are equally spaced, the B-splines are called uniform. A B-spline $B^{r_X}_{X,j_X}(x)$ can be written as a MoP function (Eq. (1)) with L_X pieces, where each piece $p_{l_X}(x)$ is defined as the expansion of Eq. (3) in the interval $A_{l_X} = [a_{l_X - 1}, a_{l_X})$, $l_X = 1, \ldots, L_X$. Fig. 1 shows ten uniform B-splines defined in $\Omega_X = [0, 10]$ for orders (a) $r_X = 3$ and (b) $r_X = 4$. With the exception of the B-splines in the limits of Ω_X, we find that each B-spline is non-zero in r_X intervals and zero in the rest. Also, for each interval A_{l_X}, we find r_X non-zero B-splines.

3.2. Learning one-dimensional MoPs

Zong [27] proposed using B-spline interpolation to find an approximation of the density $f_X(x)$ as a linear combination of $M_X = L_X + r_X - 1$ B-splines

$$\varphi_X(x; \boldsymbol{\alpha}) = \sum_{j_X = 1}^{M_X} \alpha_{j_X} B^{r_X}_{X,j_X}(x), \quad x \in \Omega_X, \tag{4}$$

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 Eq. (3). MoPs are closed under multiplication and addition. Thus, the linear combination of M_X

B-splines with order r_X (Eq. (4)) yields a MoP function with L_X pieces, where each piece $p_{l_X}(x)$ is a polynomial with order r_X defined in the interval A_{l_X}: $p_{l_X}(x) = \sum_{j_X=1}^{M_X} \alpha_{j_X} B_{X,j_X}^{r_X}(x)$, $\forall x \in A_{l_X} = [a_{l_X-1}, a_{l_X})$.

Therefore, four elements need to be specified to define a MoP using B-spline interpolation: the order (r_X), the number of intervals/pieces (L_X), the knot sequence ($\boldsymbol{\delta}_X$) and the mixing coefficients ($\boldsymbol{\alpha}$). We used uniform B-splines so the intervals A_{l_X} have an equal width of $a_{l_X} - a_{l_X-1} = \frac{\xi_X - \epsilon_X}{L_X}$, and $\boldsymbol{\delta}_X$ is easily found. The values of the order (r_X) and the number of intervals (L_X) of the MoP were found by testing different values and selecting the ones with the highest BIC score (see Section 3.4). Zong [27] derived an iterative procedure for computing the ML estimators of the mixing coefficients, $\hat{\boldsymbol{\alpha}}$, in Eq. (4). To ensure that the resulting linear combination of B-splines is a valid density (non-negative and integrating to one), the optimization procedure introduces two constraints: $\sum_{j_X=1}^{M_X} \alpha_{j_X} c_{j_X} = 1$ and $\alpha_{j_X} \geqslant 0$, $j_X = 1, \ldots, M_X$, where

$$c_{j_X} = \int_{a_{j_X - r_X}}^{a_{j_X}} B_{X,j_X}^{r_X}(x)\, dx = \frac{a_{j_X} - a_{j_X - r_X}}{r_X}.$$

Given a dataset $\mathcal{D}_X = \{x_1, \ldots, x_N\}$ with N observations of variable X, the ML estimators of the mixing coefficients are computed using the formula:

$$\hat{\alpha}_{j_X}^{(q)} = \frac{1}{N c_{j_X}} \sum_{x \in \mathcal{D}} \frac{\hat{\alpha}_{j_X}^{(q-1)} B_{X,j_X}^{r_X}(x)}{\varphi_X\left(x; \hat{\boldsymbol{\alpha}}^{(q-1)}\right)}, \quad j_X = 1, \ldots, M_X, \tag{5}$$

where q is the iteration number in the optimization process. Zong showed that Eq. (5) yields the only maximum of the log-likelihood ℓ of $\mathcal{D}_X$ given the approximation $\varphi_X(x; \boldsymbol{\alpha})$ (Eq. (4)). The initial values $\hat{\alpha}_{j_X}^{(0)}$ are set to $1/\sum_{j_X=1}^{M_X} c_{j_X}$. The relative change in the log-likelihood ℓ of $\mathcal{D}_X$ given $\varphi_X\left(x; \hat{\boldsymbol{\alpha}}^{(q)}\right)$ is used as a stopping criterion, i.e., Eq. (5) iterates until $\left|\frac{\ell^{(q)} - \ell^{(q-1)}}{\ell^{(q)}}\right| < e$, where $\ell^{(q)}$ is the log-likelihood at iteration q. We used $e = 10^{-6}$ in our experiments. The computational complexity of this optimization process is $\mathcal{O}(M_X N q_{\max})$, where $q_{\max}$ is the number of iterations of Eq. (5) performed until the optimization converges. Algorithm 1 summarizes the whole process for obtaining a MoP approximation of a probability density function using a dataset.

Algorithm 1 *(Learning a MoP approximation of a one-dimensional probability density from data).*

- *Inputs*:
  - $\mathcal{D}_X$: *A dataset with N observations* $\mathcal{D}_X = \{x_1, \ldots, x_N\}$
  - L_X: *The number of pieces of the MoP*
  - r_X: *The order of the polynomials*
- *Output*: *An* L_X*-piece* $(r_X - 1)$*-degree MoP approximation* $\varphi_X(x; \hat{\boldsymbol{\alpha}})$ *of the probability density underlying the dataset* $\mathcal{D}_X$
- *Steps*:
  1. *Compute the domain of the approximation* $\Omega_X = [\epsilon_X, \xi_X]$, *where* $\epsilon_X = \min\{x_1, \ldots, x_N\}$ *and* $\xi_X = \max\{x_1, \ldots, x_N\}$.
  2. *Compute the knot sequence* $\boldsymbol{\delta}_X = \{a_0, a_1, \ldots, a_{L_X}\}$ *and define the intervals* $A_{l_X} = [a_{l_X-1}, a_{l_X})$, $l_X = 1, \ldots, L_X$.
  3. *Apply Eq.* (3) *to build the* $M_X = L_X + r_X - 1$ *B-splines* $B_{X,j_X}^{r_X}(x)$.
  4. *Apply Eq.* (5) *iteratively to compute the ML estimators of the mixing coefficients* $\hat{\boldsymbol{\alpha}}$.
  5. *Compute the polynomials* $p_{l_X}(x)$ *from Eq.* (1) *as the linear combination of the B-splines defined for each interval* A_{l_X}, *and build the MoP.*

3.3. Learning multidimensional MoPs

The approach in Section 3.2 can be intuitively extended to learn MoP approximations of multidimensional joint probability densities. Zong and Lam [26] and Zong [27] studied B-spline approximations of two-dimensional joint probability density functions. Here, we extend their work to general n-dimensional joint probability density functions. Given a vector of n random variables $\mathbf{X} = (X_1, \ldots, X_n)$, we can approximate the joint probability density function $f_{\mathbf{X}}(\mathbf{x})$ with a multidimensional linear combination of B-splines:

$$\varphi_{\mathbf{X}}(\mathbf{x}; \boldsymbol{\alpha}) = \sum_{\substack{j_{X_1}=1,\ldots,M_{X_1} \\ \cdots \\ j_{X_n}=1,\ldots,M_{X_n}}} \alpha_{j_{X_1},\ldots,j_{X_n}} \prod_{i=1}^{n} B_{X_i,j_{X_i}}^{r_{X_i}}(x_i), \quad \mathbf{x} \in \Omega_{\mathbf{X}}, \tag{6}$$

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 $j_{X_1}, \ldots, j_{X_n}$. Fig. 2(a) shows an example of two-dimensional B-splines defined as a linear combination of the product of one-dimensional B-splines as in Eq. (6). The corresponding one-dimensional B-splines are shown in Figs. 2(b) and 2(c).

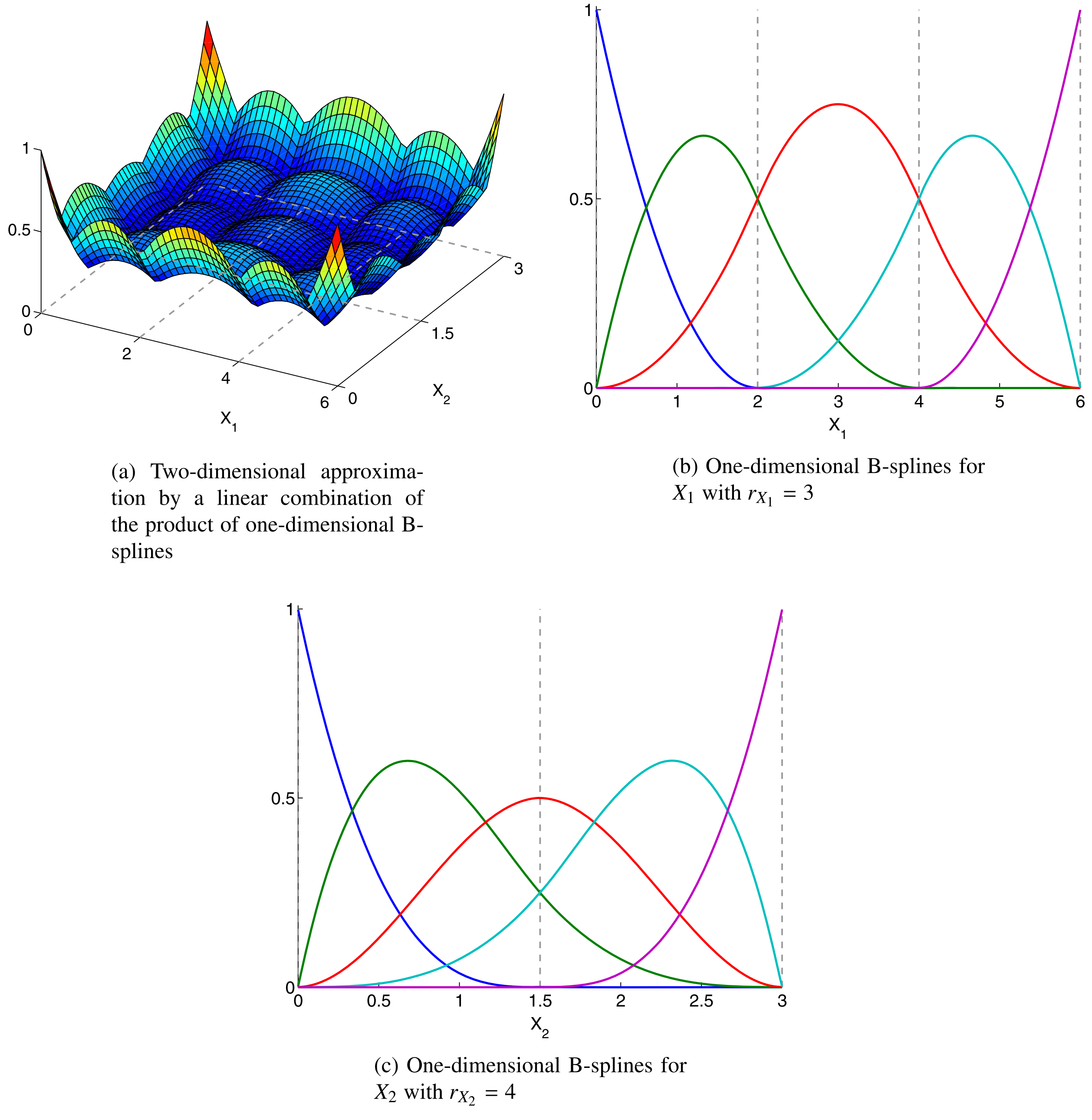


Fig. 2. (a) Two-dimensional B-splines defined as the product of one-dimensional B-splines. The domain of the approximation is $\Omega_{\mathbf{X}} = [0, 6] \times [0, 3]$. Five B-splines are used for each dimension ($M_{X_1} = M_{X_2} = 5$). Different orders are used for dimensions X_1 ($r_{X_1} = 3$) and X_2 ($r_{X_2} = 4$). Therefore, the number of intervals for each dimension is different, i.e., $L_{X_1} = 3$ and $L_{X_2} = 2$. The dashed lines show the two-dimensional subintervals (rectangles A_l) in which the two-dimensional B-splines are defined. (b) One-dimensional B-splines for X_1. (c) One-dimensional B-splines for X_2. Each B-spline in (b) and (c) is shown in a different color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

As in the one-dimensional scenario, each B-spline $B^{r_{X_i}}_{X_i, j_{X_i}}(x_i)$ can be written as a MoP, and each product of one-dimensional B-splines in Eq. (6) yields a multidimensional MoP as defined in Eq. (2). The dimensions in the multidimensional MoP $\varphi_{\mathbf{X}}(\mathbf{x}; \boldsymbol{\alpha})$ in Eq. (6) are related through the mixing coefficients $\alpha_{j_{X_1},\ldots,j_{X_n}}$, one for each combination of B-splines.

As in the one-dimensional scenario, we have to specify the number of intervals for each dimension ($L_{X_1}, \ldots, L_{X_n}$), the order of the polynomials for each dimension ($r_{X_1}, \ldots, r_{X_n}$), the knot sequence ($\boldsymbol{\delta}_{\mathbf{X}}$) and the mixing coefficients ($\boldsymbol{\alpha}$) in order to completely define a multidimensional MoP. Here, we found the knot sequence as 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 $\boldsymbol{\delta}_{X_i}$ are defined to yield equal-width intervals as in the one-dimensional case (see Section 3.2). 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 $\prod_{i=1}^{n} L_{X_i}$ pieces, where each piece $p_{l_{X_1},\ldots,l_{X_n}}(\mathbf{x})$ is defined in an n-dimensional hyperrectangle $A_{l_{X_1},\ldots,l_{X_n}} = [a_{l_{X_1}-1}, a_{l_{X_1}}] \times \cdots \times [a_{l_{X_n}-1}, a_{l_{X_n}}]$.

Given a dataset with N observations of n-dimensional vectors $\mathcal{D}_{\mathbf{X}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_z = (x_{z,1}, \ldots, x_{z,n})$, $z = 1, \ldots, N$, the ML estimators of the mixing coefficients $\hat{\boldsymbol{\alpha}}$ in Eq. (6) are approached by the iteration formula

$$\hat{\alpha}^{(q)}_{j_{X_1},\ldots,j_{X_n}} = \frac{1}{N c_{j_{X_1},\ldots,j_{X_n}}} \sum_{\mathbf{x}\in\mathcal{D}_{\mathbf{X}}} \frac{\hat{\alpha}^{(q)}_{j_{X_1},\ldots,j_{X_n}} \prod_{i=1}^{n} B^{r_{X_i}}_{X_i,j_{X_i}}(x_i)}{\varphi_{\mathbf{X}}\big(\mathbf{x}\,;\,\hat{\boldsymbol{\alpha}}^{(q-1)}\big)}, \tag{7}$$

subject to the constraints $\sum_{\substack{j_{X_1}=1,\ldots,M_{X_1} \\ \cdots \\ j_{X_n}=1,\ldots,M_{X_n}}} \alpha_{j_{X_1},\ldots,j_{X_n}} c_{j_{X_1},\ldots,j_{X_n}} = 1$ and $\alpha_{j_{X_1},\ldots,j_{X_n}} \geqslant 0$, where $j_{X_i} = 1, \ldots, M_{X_i}$, $i = 1, \ldots, n$, and

$$c_{j_{X_1},\ldots,j_{X_n}} = \prod_{i=1}^{n} \int_{a_{j_{X_i}-r_{X_i}}}^{a_{j_{X_i}}} B^{r_{X_i}}_{X_i,j_{X_i}}(x_i)\,dx_i = \prod_{i=1}^{n} \frac{a_{j_{X_i}} - a_{j_{X_i}-r_{X_i}}}{r_{X_i}}.$$

Algorithm 2 details the steps for learning a MoP approximation of a multidimensional joint density from a dataset of observations.

Algorithm 2 *(Learning a MoP approximation of a multidimensional joint probability density from data).*

- *Inputs*:
  - $\mathcal{D}_{\mathbf{X}}$: *A dataset with* N *observations* $\mathcal{D}_{\mathbf{X}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
  - $L_{X_1}, \ldots, L_{X_n}$: *The number of pieces of the MoP for each dimension*
  - $r_{X_1}, \ldots, r_{X_n}$: *The order of the polynomials for each dimension*
- *Output*: *A multidimensional MoP approximation* $\varphi_{\mathbf{X}}(\mathbf{x}; \hat{\boldsymbol{\alpha}})$ *of the joint probability density underlying the dataset* $\mathcal{D}_{\mathbf{X}}$
- *Steps*:
  1. *Compute the domain of the approximation for each dimension* $\Omega_{X_i} = [\epsilon_{X_i}, \xi_{X_i}]$, $i = 1, \ldots, n$, *where* $\epsilon_{X_i} = \min\{x_{1,i}, \ldots, x_{N,i}\}$ *and* $\xi_{X_i} = \max\{x_{1,i}, \ldots, x_{N,i}\}$.
  2. *Compute the multidimensional domain of the approximation* $\Omega_{\mathbf{X}} = \Omega_{X_1} \times \cdots \times \Omega_{X_n}$.
  3. *Compute the knot sequence for each dimension* $\boldsymbol{\delta}_{X_i} = \{a_{X_i,0}, a_{X_i,1}, \ldots, a_{X_i,L_{X_i}}\}$ *and define the hyperrectangles* $A_{l_{X_1},\ldots,l_{X_n}} = [a_{l_{X_1}-1}, a_{l_{X_1}}] \times \cdots \times [a_{l_{X_n}-1}, a_{l_{X_n}}]$ *for each piece.*
  4. *Apply Eq.* (3) *to build the* $M_{X_i} = L_{X_i} + r_{X_i} - 1$ *B-splines* $B^{r_{X_i}}_{X_i,j_{X_i}}(x_i)$ *for each dimension* $i = 1, \ldots, n$.
  5. *Apply Eq.* (7) *to compute the ML estimators of the mixing coefficients* $\hat{\boldsymbol{\alpha}}$.
  6. *Compute the polynomials* $p_{l_{X_1},\ldots,l_{X_n}}(\mathbf{x})$ *from Eq.* (2) *as the linear combination of the B-splines in Eq.* (6) *and build the MoP.*

3.4. Model selection

The number of pieces L_X and the order of the polynomials r_X have to be specified a priori in Algorithm 1. Similarly, the parameters L_{X_i} and r_{X_i} have to be specified for each variable $X_i, i = 1, \ldots, n$, in Algorithm 2. Since the ML estimators of the mixing coefficients, $\hat{\boldsymbol{\alpha}}$, are computed in Eqs. (5) and (7), we can use a penalized likelihood score to perform model selection in a principled way. Here, we used the BIC [45] to find appropriate values for the order of the polynomials and the number of pieces of the MoP. The BIC score is defined as

$$BIC\big(\varphi_{\mathbf{X}}(\mathbf{x}), \mathcal{D}_{\mathbf{X}}\big) = \ell\big(\mathcal{D}_{\mathbf{X}} \big| \varphi_{\mathbf{X}}(\mathbf{x})\big) - \frac{\dim(\varphi_{\mathbf{X}}(\mathbf{x})) \log N}{2}, \tag{8}$$

where $\ell(\mathcal{D}_{\mathbf{X}}|\varphi_{\mathbf{X}}(\mathbf{x}))$ is the log-likelihood of the training dataset $\mathcal{D}_{\mathbf{X}}$ given a MoP model $\varphi_{\mathbf{X}}(\mathbf{x})$, N is the number of observations in the dataset $\mathcal{D}_{\mathbf{X}}$ and $\dim(\varphi_{\mathbf{X}}(\mathbf{x}))$ is the dimension of the model. Given the knot sequence $\boldsymbol{\delta}_{\mathbf{X}}$, the order $r_{X_1}, \ldots, r_{X_n}$ of the B-splines and the number of pieces $L_{X_1}, \ldots, L_{X_n}$, the only free parameters that need to be estimated are the mixing coefficients $\boldsymbol{\alpha}$ of the linear combination of B-splines. Therefore, we used the number of mixing coefficients as a measure of the dimension of the MoP, i.e., $\dim(\varphi_{\mathbf{X}}(\mathbf{x})) = (\prod_{i=1}^{n} M_{X_i})$. In all our experiments, we selected the MoP approximation with the highest BIC score.

4. Experiments

We report the results of the experiments to evaluate the proposed methods. Section 4.1 includes the experiments related to MoP learning of probability densities from data. Section 4.2 reports the results of the Bayesian classifiers using MoPs as a non-parametric density estimation technique. Section 4.3 shows a comparison of the evaluation time of MoPs versus kernel density estimation.

Table 1
Probability density functions used to sample artificial datasets and for learning mixtures of polynomials.

Name	Dimensions	Distribution	Domain
`Gauss`	1	$\mathcal{N}(0,1)$	$[-3,3]$
`Exp`	1	$\text{Exp}(1)$	$[0,3]$
`Chisq`	1	χ_3^2	$[0,8]$
`MixGauss`	1	$0.5\mathcal{N}(0,1)+0.5\mathcal{N}(4,1)$	$[-3,7]$
`Mix1d`	1	$0.8\chi^2(3)+0.2\mathcal{N}(7,1)$	$[0,10]$
`Gauss2d`	2	$\mathcal{N}\left((0,0),\begin{pmatrix}1&0\\0&1\end{pmatrix}\right)$	$[-3,3]\times[-3,3]$
`Mix2d`	2	$0.7\mathcal{N}\left((5,6),\begin{pmatrix}0.5&0.3\\0.3&0.5\end{pmatrix}\right)+0.3\mathcal{N}\left((7,6.5),\begin{pmatrix}0.4&-0.2\\-0.2&0.4\end{pmatrix}\right)$	$[3,8]\times[4,8]$
`Mix3d`	3	$0.4\mathcal{N}\left((3,5.5,4),\begin{pmatrix}1&0.5&0.3\\0.5&0.5&0\\0.3&0&1\end{pmatrix}\right)+0.6\mathcal{N}\left((4,4,2),\begin{pmatrix}1&0.75&-0.1\\0.75&1.5&-0.2\\-0.1&-0.2&2\end{pmatrix}\right)$	$[2,7]\times[1,7]\times[0,8]$

4.1. Experiments with MoP approximations

In this section we report the experiments on the MoP approximations of one-dimensional and multidimensional probability densities from data.

4.1.1. Artificial datasets

We sampled datasets with different number of observations N from known one-dimensional and multidimensional densities with different shapes. The study included both known parametric probability densities and mixtures of densities. Table 1 shows the name of the datasets, the respective probability distributions and the domains of approximation.

4.1.2. Comparison measures

We used different measures to evaluate the quality of the MoPs learned from the datasets. First, two measures are reported to analyze the goodness of fit of the MoPs to the datasets from which they were learned:

- The log-likelihood (ℓ) of the dataset given the MoP. We used both the training dataset and a test dataset with the same size as the training dataset.
- The BIC score of the dataset given the MoP. We used both the training dataset and a test dataset with the same size as the training dataset.

Additionally, we report three measures for evaluating how close the MoP approximation is to the true density which generated the training datasets (Table 1):

- The Kullback–Leibler (KL) divergence [46] of the MoP $\varphi_{\mathbf{X}}(\mathbf{x})$ from the true density $f_{\mathbf{X}}(\mathbf{x})$:

$$KL\big(f_{\mathbf{X}}(\mathbf{x}),\varphi_{\mathbf{X}}(\mathbf{x})\big)=\int_{\Omega_{\mathbf{X}}} f_{\mathbf{X}}(\mathbf{x})\log\frac{f_{\mathbf{X}}(\mathbf{x})}{\varphi_{\mathbf{X}}(\mathbf{x})}\,d\mathbf{x}.$$

- The mean squared error (MSE) between the MoP $\varphi_{\mathbf{X}}(\mathbf{x})$ and the true density $f_{\mathbf{X}}(\mathbf{x})$:

$$MSE\big(f_{\mathbf{X}}(\mathbf{x}),\varphi_{\mathbf{X}}(\mathbf{x})\big)=\int_{\Omega_{\mathbf{X}}} f_{\mathbf{X}}(\mathbf{x})\big(f_{\mathbf{X}}(\mathbf{x})-\varphi_{\mathbf{X}}(\mathbf{x})\big)^2\,d\mathbf{x}.$$

- The maximum absolute error (MAE) between the MoP $\varphi_{\mathbf{X}}(\mathbf{x})$ and the true density $f_{\mathbf{X}}(\mathbf{x})$:

$$MAE\big(f_{\mathbf{X}}(\mathbf{x}),\varphi_{\mathbf{X}}(\mathbf{x})\big)=\max_{\mathbf{x}\in\Omega_{\mathbf{X}}}\big|f_{\mathbf{X}}(\mathbf{x})-\varphi_{\mathbf{X}}(\mathbf{x})\big|.$$

The MoP $\varphi_{\mathbf{X}}(\mathbf{x})$ needs to be a proper density (non-negative and integrating to one in $\Omega_{\mathbf{X}}$) to compute the log-likelihood and the BIC scores. Similarly, both the MoP $\varphi_{\mathbf{X}}(\mathbf{x})$ and the true density $f_{\mathbf{X}}(\mathbf{x})$ need to be proper densities in the domain $\Omega_{\mathbf{X}}$ to compute the KL divergence. MoPs using B-splines learned with Algorithms 1 and 2 are ensured to be non-negative and integrate to one in $\Omega_{\mathbf{X}}$ (see Sections 3.2 and 3.3). MoTBFs learned using the approach in [17] are valid densities (see Section 4.1.3). Also, non-negative MoPs learned using LIPs were considered and normalized to integrate to one in $\Omega_{\mathbf{X}}$ (see

Section 4.1.3). Finally, the true probability densities $f_{\mathbf{X}}(\mathbf{x})$ also need to be normalized because a (small) part of the density mass lays outside the domain of approximation $\Omega_{\mathbf{X}}$. Thus, a normalization constant $T = \int_{\Omega_{\mathbf{X}}} f_{\mathbf{X}}(\mathbf{x})\,d\mathbf{x}$ was computed and the true density values $f_{\mathbf{X}}(\mathbf{x})$ were normalized by multiplying them by $1/T$ when computing KL, MSE and MAE.

The integrals in the KL divergence and the MSE were computed using an adaptive quadrature integration procedure available in `R` [47,48]. To avoid local maxima, the MAE was computed from the density differences on 1000 equally-spaced points defined in the domain $\Omega_{\mathbf{X}}$. Then, the 50 points yielding the maximum values of $|f_{\mathbf{X}}(\mathbf{x}) - \varphi_{\mathbf{X}}(\mathbf{x})|$ in the equally-spaced grid were used as starting values of a non-linear optimization algorithm bounded to the interval $\Omega_{\mathbf{X}}$ (`nlminb` function in `R`) to find the global maximum.

4.1.3. Related work

We compared the proposed method for learning MoP approximations of one-dimensional densities from data with two proposals from the literature:

- **MoP approximation using Lagrange interpolating polynomials**: The results were compared with the MoPs obtained by computing the LIP over the Chebyshev points defined in each interval independently [16]. The density at the Chebyshev points was estimated using kernel density estimation. Gaussian kernels with the normal scale bandwidth were computed using the `ks` package [49]. We considered different values for the order r_X and the number of pieces L_X of the MoP. Equal-width intervals A_{l_X} were assumed for each piece $l_X = 1, \ldots, L_X$. The BIC score (see Section 3.4) was used to select appropriate values for the order r_X and the number of pieces L_X of the MoPs. Here, the number of polynomial coefficients different from zero was used as a measure of the dimension of the model $\dim(\varphi_X(x))$ in Eq. (8).
  We checked whether or not the MoP approximations learned with LIPs yielded negative values. To check this situation, the values of $\varphi_X(x)$ were computed at 1000 equally-spaced points in Ω_X. Then, the points corresponding to local minima values of $\varphi_X(x)$ were found. These points were used as the starting points for a non-linear optimization procedure (`nlminb` function in `R`) to find the global minimum $\upsilon = \min_{x \in \Omega_X} \varphi_X(x)$. If the MoP yielded negative values, the minimum value of the function was added to the MoP: $\varphi_X(x) \leftarrow \varphi_X(x) - \upsilon$. Then, the normalization constant was computed as $T = \int_{\Omega_X} \varphi_X(x)\,dx$ and the MoP was normalized by computing $\varphi_X(x) \leftarrow \frac{1}{T}\varphi_X(x)$. These two steps (finding the minimum value and normalizing) were repeated until the final MoP $\varphi_X(x)$ was non-negative. Thus, we ensured that $\varphi_X(x)$ learned with LIPs was non-negative and integrated to one in Ω_X. Other options could be considered to ensure that the MoP approximations with LIPs are proper densities, e.g., see [50].
  As opposed to the one-dimensional scenario, multidimensional Lagrange interpolation is more challenging [51], although some approaches have been proposed for two-dimensional scenarios, e.g., see [52–54]. Additionally, estimating the densities at the interpolation nodes is more difficult in multidimensional domains than in the one-dimensional scenario.
- **Mixtures of Truncated Basis Functions**: We compared the proposed approach with the method proposed in [17] for learning MoTBF approximations of one-dimensional densities from data. MoPs are a particular scenario of MoTBFs when using polynomials as basis functions. Here, we used Legendre polynomials as basis functions. The MoTBF approximations obtained in [17] are valid densities, i.e., non-negative and integrating to one. Additionally, the approach does not split the domain of approximation Ω_X, i.e., the resulting MoTBF has only one piece ($L_X = 1$) and it is therefore continuous. We considered Legendre polynomials up to order $r_X = 7$. The BIC score was used to select the order r_X of the final MoTBF approximation.

4.1.4. Results for one-dimensional datasets

We analyzed the behavior of Algorithm 1 used to build MoP approximations of probability densities from data using artificial examples. For each one of the datasets in Table 1, we sampled ten training sets and ten test sets with $N = 50, 100, 500, 1000$ observations each. From each training dataset, we found a MoP approximation of the probability density underlying the data using Algorithm 1. We considered different values for the order of the polynomials $r_X = 2, \ldots, 5$ and the number of intervals/pieces $L_X = 1, \ldots, 10$.

Fig. 3 shows the MoP approximations for each of the one-dimensional datasets in Table 1. For the first repetition, the MoPs with the highest BIC score in the training dataset with $N = 1000$ are shown. The MoPs learned with the proposed Algorithm 1 (dashed lines) are continuous, non-negative and integrate to one. Eq. (9) is an example of the MoP with the highest BIC score learned with Algorithm 1 for the `Mix1d` dataset ($L_X = 5$ and $r_X = 3$) in Fig. 3(d):

$$\varphi_X(x) = \begin{cases} 0.1006 + 0.1266x - 0.0451x^2, & 0 \leqslant x \leqslant 2, \\ 0.3038 - 0.0766x + 0.0057x^2, & 2 \leqslant x \leqslant 4, \\ 0.4843 - 0.1668x + 0.0169x^2, & 4 \leqslant x \leqslant 6, \\ -1.0028 + 0.3289x - 0.0244x^2, & 6 \leqslant x \leqslant 8, \\ 1.6187 - 0.3265x + 0.0166x^2, & 8 \leqslant x \leqslant 10. \end{cases} \tag{9}$$

It is easy to check that the MoP in Eq. (9) is continuous for $x = 2, 4, 6, 8$ and that the integral of $\varphi_X(x)$ in $\Omega_X = [0, 10]$ is 1.

Table 2 shows the numerical comparison of the MoPs learned using B-splines, LIPs and the MoTBF approach for the one-dimensional datasets in Table 1. For each of the ten experiments, we selected the MoP with the highest BIC score (8) in training. Table 2 reports the mean of the ten values for the number of pieces (L_X), order of the polynomials (r_X), number

Table 2
Comparison of MoPs learned from data using B-splines (Algorithm 1) with LIPs and MoTBFs (Section 4.1.3). For each of the ten repetitions, the MoP with the highest BIC score in training was selected, and the mean values of the performance measures were reported: order of the polynomials (r_X), number of pieces (L_X), number of free parameters to estimate (#par), Kullback–Leibler divergence (KL), mean squared error (MSE) and maximum absolute error (MAE). The best (lowest) value for each dataset, training size (N) and performance measure is shown in bold. Statistically significant results of MoPs using B-splines with respect to LIPs and MoTBFs are marked with $*$ and $\dagger$, respectively. A Wilcoxon signed-rank tests was used with $\alpha = 0.05$.

	B-splines					LIP					MoTBF				
	`Gauss`	`Exp`	`Chisq`	`MixGauss`	`Mix1d`	`Gauss`	`Exp`	`Chisq`	`MixGauss`	`Mix1d`	`Gauss`	`Exp`	`Chisq`	`MixGauss`	`Mix1d`
								$N = 50$							
r_X	**2.1***†	**2.3***†	**2.0**	2.6	2.0	3.8	3.5	2.3	3.0	2.0	4.4	3.2	2.5	**2.1**	**1.9**
L_X	2.7	1.1	**1.0**	2.9	1.2	1.4*	1.2	**1.0**	1.3	**1.0**	**1.0**†	**1.0**	**1.0**	**1.0**	**1.0**
#par	**3.8**	**2.4***†	**2.0**	4.5	2.2	4.6	3.9	2.3	3.8	2.0	4.4	3.2	2.5	**2.1**†	**1.9**
KL	0.2303	0.0342	**0.0259***	0.1606	0.0434	**0.0728**	0.0555	0.0384	**0.1205**	**0.0417**	0.1796	**0.0341**	0.0482	0.2168	0.0522
MSE	**0.0045**	0.0199	**0.0010**	**0.0024**†	0.0008	0.0045	0.0148	0.0013	0.0028	**0.0008**	0.0121	**0.0085**†	0.0016	0.0040	0.0010
MAE	0.1064	0.3018	0.2461	0.0922	0.1889	**0.1049**	0.2564	**0.2330**	**0.0880**	0.1715*	0.1619	**0.1706**†	0.2332	0.1015	**0.1682**†
								$N = 100$							
r_X	**2.3***†	**2.1***†	**2.0**†	**2.4**	**2.0**	3.9	3.4	2.1	3.0	**2.0**	4.0	3.5	3.3	3.1	2.4
L_X	3.6	1.5	1.4	4.1	1.2	1.5*	1.1	1.1	2.1*	**1.0**	**1.0**†	**1.0**†	**1.0**	**1.0**†	**1.0**
#par	4.9	**2.6***†	2.4	5.5	2.2	5.3	3.6	**2.3**	6.0	**2.0**	**4.0**	3.5	3.3	**3.1**†	2.4
KL	0.0359†	0.0265*	0.0311	**0.0623**†	0.0410	**0.0356**	0.0551	**0.0301**	0.0715	0.0416	0.1858	**0.0242**	0.0513	0.1432	**0.0389**
MSE	0.0019†	0.0159	0.0013	**0.0015**†	0.0008	**0.0019**	0.0186	**0.0012**	0.0018	0.0008	0.0131	**0.0077**	0.0015	0.0033	**0.0007**
MAE	**0.0880**†	0.2670	0.2368	**0.0707**†	0.1909	0.0948	0.3041	0.2397	0.0821	0.1825	0.1638	**0.1704**	**0.2086**	0.0941	**0.1661**†
								$N = 500$							
r_X	**2.8**†	**2.7**	**2.6**†	**2.8***	2.9	3.5	3.4	3.1	3.8	**2.4**	4.6	3.4	4.5	3.7	4.6
L_X	3.2	1.3	1.4	5.6	2.9	1.8*	1.9	**1.0**	2.7*	1.4*	**1.0**†	**1.0**	**1.0**	**1.0**†	**1.0**†
#par	5.0*	**3.0***	**3.0**†	7.4*	4.8	6.0	5.3	3.1	9.8	**3.6**	**4.6**	3.4	4.5	**3.7**†	4.6
KL	**0.0069***†	**0.0041***	0.0190	**0.0078***†	**0.0205**	0.0148	0.0131	0.0215	0.0201	0.0305	0.1120	0.0092	**0.0158**	0.1327	0.0213
MSE	**0.0003***†	0.0030	0.0009	**0.0002**†	0.0004	0.0008	0.0039	0.0009	0.0005	0.0006	0.0075	**0.0022**	**0.0004**†	0.0030	**0.0004**
MAE	**0.0336***†	0.1434	0.2079	**0.0277**†	0.1389	0.0552	0.1486	0.1926	0.0401	0.1657	0.1136	**0.1014**†	**0.1687**	0.0924	**0.1206**
								$N = 1000$							
r_X	**2.9**†	**2.8***	**3.6**†	**2.9***	**3.3**†	3.3	4.2	4.0	3.8	**3.3**	5.1	3.8	6.0	3.7	6.6
L_X	3.2	1.6	2.7	5.2	4.2	2.2*	1.5	**1.0***	3.1*	2.5*	**1.0**†	**1.0**†	**1.0**†	**1.0**†	**1.0**†
#par	**5.1***	**3.4***	5.3	7.1*	**6.5***	7.2	5.6	**4.0**	11.1	7.7	**5.1**	3.8	6.0	**3.7**†	6.6
KL	**0.0043***†	**0.0031***	**0.0071***	**0.0047***†	**0.0068***	0.0087	0.0074	0.0151	0.0115	0.0120	0.1056	0.0082	0.0109	0.1199	0.0118
MSE	**0.0003***†	0.0022	**0.0003***	**0.0001***†	**0.0002***	0.0006	0.0018	0.0006	0.0002	0.0004	0.0067	**0.0017**	0.0003	0.0029	0.0002
MAE	**0.0311***†	0.1199	0.1551	**0.0188***†	**0.0982**	0.0511	**0.0927**	0.1724	0.0288	0.0995	0.1054	0.1363	**0.1298**	0.0930	0.1002

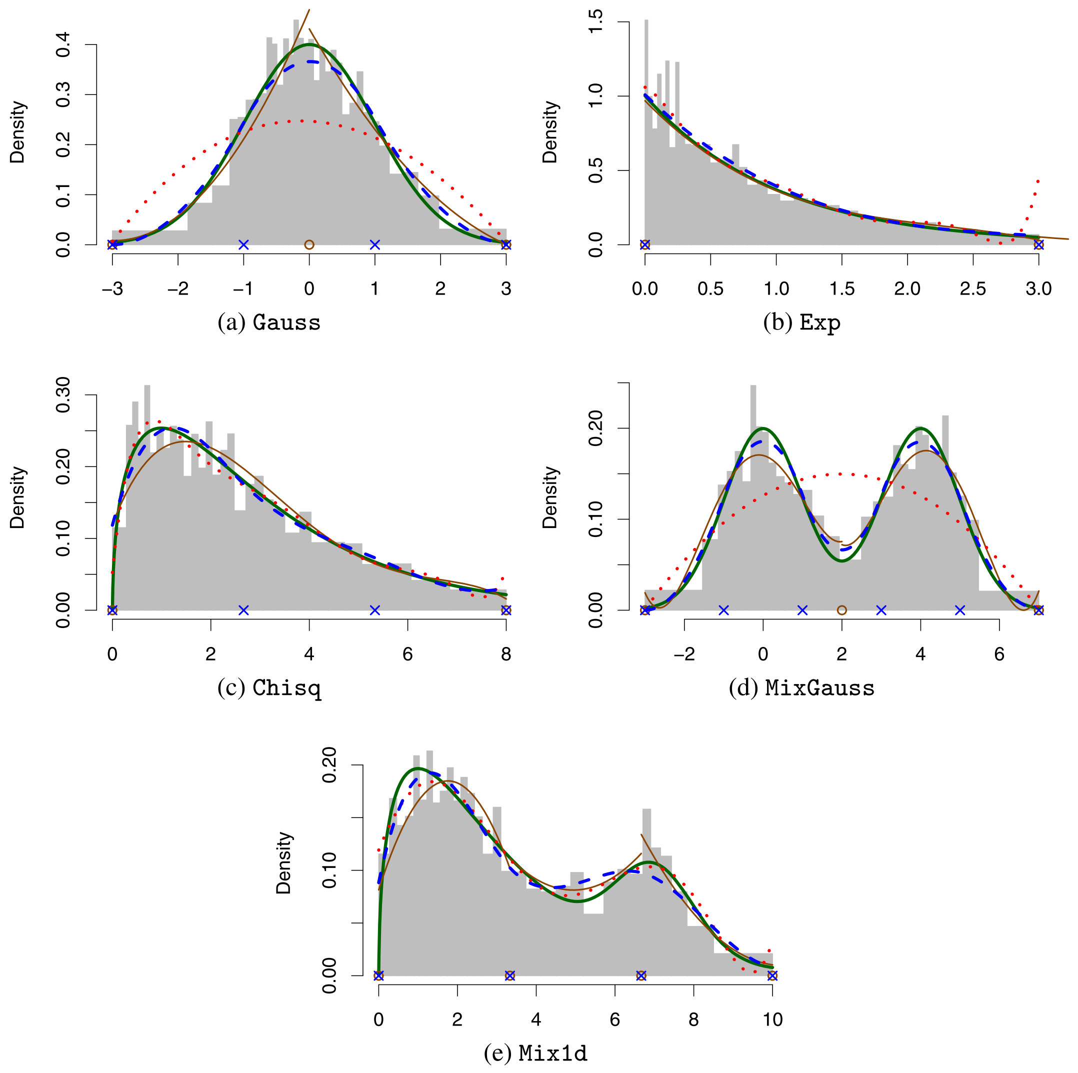


Fig. 3. MoP approximations learned from a training dataset of $N = 1000$ observations. The MoPs with the highest BIC score for the first training dataset learned with Algorithm 1 (dashed lines), using Lagrange interpolating polynomials (thin solid lines) and using MoTBFs (dotted lines) are shown. A thick solid line represents the true density used to generate the training datasets. An equal-frequency histogram of the training dataset is displayed in gray. Crosses along the horizontal axis mark the limits of the intervals for the MoPs learned with Algorithm 1, whereas circles mark the intervals for the MoPs learned with Lagrange interpolating polynomials. MoTBFs do not split the approximation domain.

of free parameters to be estimated from data (#par), KL divergence, MSE and MAE. The best result out of the three methods (B-splines, LIP and MoTBF) is highlighted in bold for each dataset, sample size (N) and performance measure. We performed a non-parametric Wilcoxon paired signed-rank test to check whether or not the differences between the methods (B-splines vs. LIPs and MoTBFs) in the ten repetitions were significant. Statistically significant differences between B-splines and LIPs at a significance level $\alpha = 0.05$ are marked with an asterisk ($*$). Similarly, statistically significant differences between B-splines and MoTBFs are shown with † symbols. In general, we can see that the three methods performed similarly in datasets with fewer observations ($N = 50, 100$) according to KL, MSE and MAE. However, for larger sample sizes ($N = 1000$), MoPs learned with the proposed approach yielded the best result in all but three scenarios (`MSE` and `MAE` for `Exp` dataset, and MAE for `Chisq` dataset).

According to the KL divergence and MSE, MoPs learned with LIPs did not significantly outperform MoPs learned with B-splines in any experiment. On the other hand, MoPs learned with B-splines frequently outperformed MoPs learned with LIPs, specially in datasets with higher sample sizes ($N = 500, 1000$). Regarding the MAE values, MoPs learned with LIPs significantly outperformed MoPs learned with B-splines in only one scenario (`Mix1d` dataset with $N = 50$). On the other hand, MoPs learned with B-splines significantly outperformed MoPs learned with LIPs according to MAE in three scenarios (`Gauss` dataset with $N = 500, 1000$ and `MixGauss` dataset with $N = 1000$). In general, MoPs learned with B-splines had a lower order r_X than MoPs learned with LIPs, whereas MoPs learned with LIPs had fewer pieces L_X than MoPs learned with B-splines. MoPs learned with B-splines frequently outperformed MoPs learned with LIPs regarding the number of free

Table 3
Goodness of fit to training data of MoPs learned using B-splines, LIPs or MoTBFs. For each of the ten repetitions, the MoP with the highest BIC score in training was selected. The mean of the BIC and the log-likelihood values for the training datasets were reported. The best (highest) values for each dataset, training size (N) and performance measure are shown in bold. Statistically significant results of MoPs using B-splines with respect to LIPs and MoTBFs are marked with $*$ and $\dagger$, respectively. A Wilcoxon signed-rank tests was used with $\alpha = 0.05$.

	`Gauss`	`Exp`	`Chisq`	`MixGauss`	`Mix1d`
			B-spline		
$N = 50$					
ℓ	**−74.2434**	−92.9197	−129.8114	**−105.4568***	−121.8635*
BIC	**−81.6763***	−97.6141	−133.7234	**−114.2589***	−126.1668*
$N = 100$					
ℓ	**−140.3938***†	−202.9901	−298.2955*	**−204.5909***†	−238.3183*
BIC	**−151.6765***†	−208.9768	−303.8217*	**−217.2551***†	−243.3840*
$N = 500$					
ℓ	**−742.0852***	−936.4823	−1430.2364*	**−1034.8860***†	−1251.5952*
BIC	**−757.6217***	−945.8042	−1439.5583*	**−1057.8801***†	−1266.5103*
$N = 1000$					
ℓ	**−1465.4902**	−1894.6422	−2820.4319*	**−2086.0252***†	−2432.0485
BIC	**−1483.1049***	−1906.3854	−2838.7374*	**−2110.5478***†	−2454.4987*
			LIP		
$N = 50$					
ℓ	−75.2285	−56.9162*	−129.5575	−108.8679	−122.4329
BIC	−84.2261	−64.5446*	−134.0564	−116.3008	−126.3449
$N = 100$					
ℓ	−141.4615	−107.7676*	−298.8976	−207.8215	−238.9595
BIC	−153.6652	−116.0569*	−304.1935	−221.6370	−243.5647
$N = 500$					
ℓ	−744.4494	−505.7897*	−1431.0348	−1039.7299	−1259.1202
BIC	−763.0932	−522.2584*	−1440.6675	−1070.1815	−1270.3065
$N = 1000$					
ℓ	−1467.9710	−1026.9327*	−2828.6025	−2089.9849	−2435.6393
BIC	−1492.8390	−1046.2744*	−2842.4180	−2128.3230	−2462.2342
			MoTBF		
$N = 50$					
ℓ	−78.7599	**−44.8323**†	**−101.4539**†	−112.4362	**−110.8565**
BIC	−87.3664	**−51.0915**†	**−106.3439**†	−116.5438	**−114.5729**
$N = 100$					
ℓ	−158.4464	**−99.4518**†	**−200.9559**†	−218.4757	**−219.0745**
BIC	−167.6568	**−107.5109**†	**−208.5544**†	−225.6137	**−224.6007**
$N = 500$					
ℓ	−772.2923	**−426.3810**†	**−981.8123**†	−1093.0535	**−1097.6671**†
BIC	−786.5859	**−436.9458**†	**−995.7952**†	−1104.5505	**−1111.9607**†
$N = 1000$					
ℓ	−1519.3761	**−864.4513**†	**−1922.0468**†	−2172.7794	**−2175.2008**†
BIC	−1536.9909	**−877.5760**†	**−1942.7701**†	−2185.5588	**−2197.9964**†

parameters that need to be estimated from data (#par). MoPs learned with LIPs did not significantly outperform MoPs learned with B-splines in any scenario according to #par.

When we compared MoPs learned with B-splines and the MoTBF approach, we found that MoPs learned using B-splines significantly outperformed MoTBFs according to the KL, MSE and MAE values for the `Gauss` and `MixGauss` datasets. On the other hand, MoTBFs significantly outperformed MoPs learned with B-splines for MSE and MAE in some experiments with datasets `Exp` (MSE and MAE for $N = 50$, MAE for $N = 500$), `Chisq` (MSE for $N = 500$) and `Mix1d` (MAE for $N = 50, 100$). MoPs learned using B-splines frequently had significantly lower orders (r_X) than MoTBFs. On the other hand, MoTBFs frequently had significantly fewer intervals (L_X) than MoPs with B-splines, because MoTBFs do not split the approximation domain. Summarizing, the proposed B-spline approach yields MoPs with several pieces of lower order polynomials, whereas MoTBFs yield only one high-order polynomial.

Additionally, Tables 3 and 4 show the mean log-likelihood and mean BIC values in the training and the test datasets, respectively, for the three methods (B-splines, LIPs and MoTBFs). The best results out of the three methods (B-splines, LIPs and MoTBFs) are highlighted in bold. Statistically significant results according to a Wilcoxon signed-rank test are marked: $*$ for the comparison of B-splines vs. LIPs, and $\dagger$ for the comparison of B-splines vs. MoTBFs. MoPs learned using B-spline interpolation yielded the best results for the `Gauss` and `MixGauss` datasets, significantly outperforming both LIPs and MoTBFs. Additionally, B-splines significantly outperformed LIPs in `Chisq` and `Mix1d` datasets. On the other hand, MoPs learned with LIPs yielded significantly better log-likelihood and BIC scores in training and test datasets than MoPs learned with B-splines for the `Exp` dataset. Also, MoTBFs significantly outperformed MoPs learned with the proposed approach for the `Exp`, `Chisq` and `Mix1d` datasets.

Table 4
Goodness of fit to test data of MoPs learned using B-splines, LIPs or MoTBFs. For each of the ten repetitions, the MoP with the highest BIC score in training was selected. The mean of the BIC and the log-likelihood values for the test datasets were reported. The best (highest) values for each dataset, training size (N) and performance measure are shown in bold. Statistically significant results of MoPs using B-splines with respect to LIPs and MoTBFs are marked with $*$ and †, respectively. A Wilcoxon signed-rank tests was used with $\alpha = 0.05$.

	`Gauss`	`Exp`	`Chisq`	`MixGauss`	`Mix1d`
			B-spline		
$N = 50$					
ℓ	−79.5767	−93.2633	−129.8750	−114.7723	−123.3364
BIC	−87.0095†	−97.9577	−133.7870	−123.5744	−127.6396
$N = 100$					
ℓ	**−146.3397**†	−204.1776	−301.2035	**−209.8748***†	−239.9437
BIC	**−157.6224***†	−210.1643	−306.7297	**−222.5390***†	−245.0094
$N = 500$					
ℓ	**−746.2855***	−936.5723	−1435.5890*	**−1039.9328***†	−1259.3034*
BIC	**−761.8220***	−945.8942	−1444.9109	**−1062.9269***†	−1274.2185*
$N = 1000$					
ℓ	**−1469.7016***	−1896.4301	−2824.9728*	**−2091.7876***†	−2435.8958*
BIC	**−1487.3164***	−1908.1733	−2843.2783*	**−2116.3102***†	−2458.3460*
			LIP		
$N = 50$					
ℓ	**−77.6997**	−58.9507*	−129.9454	**−110.7329***	−122.6929
BIC	**−86.6974***	−66.5792*	−134.4443	−118.1658*	−126.6049
$N = 100$					
ℓ	−146.6636	−111.9894*	−300.9243	−211.3858	−239.5149
BIC	−158.8673	−120.2787**	−306.2203	−225.2013	−244.1201
$N = 500$					
ℓ	−748.0553	−511.8273*	−1436.5960	−1049.0035	−1265.4826
BIC	−766.6991	−528.2960*	−1446.2287	−1079.4551	−1276.6688
$N = 1000$					
ℓ	−1473.6051	−1029.4941*	−2832.5492	−2099.5512	−2440.9640
BIC	−1498.4730	−1048.8358*	−2846.3647	−2137.8892	−2467.5588
			MoTBF		
$N = 50$					
ℓ	−79.1049	**−44.8089**†	**−101.5481**†	−113.7129	**−111.6256**†
BIC	−87.7113	**−51.0682**†	**−106.4381**†	**−117.8206**	**−115.3420**†
$N = 100$					
ℓ	−158.3109	**−95.4190**†	**−203.3731**†	−218.8418	**−219.7180**†
BIC	−167.5212	**−103.4780**†	**−210.9716**†	−225.9798	**−225.2442**†
$N = 500$					
ℓ	−766.1069	**−426.1658**†	**−989.0500**†	−1092.1853	**−1103.0910**†
BIC	−780.4004	**−436.7306**†	**−1003.0329**†	−1103.6824	**−1117.3846**†
$N = 1000$					
ℓ	−1528.5510	**−862.8373**†	**−1926.7964**†	−2171.7743	**−2178.6629**†
BIC	−1546.1658	**−875.9620**†	**−1947.5197**†	−2184.5536	**−2201.4585**†

Finally, we assessed the accuracy of the coefficients of the polynomial functions in the MoP approximations learned with Algorithm 1. We considered the following one-dimensional polynomial functions:

$$g_X(x) = 2.25 - 7.5x + 7.5x^2, \quad 0 \leqslant x \leqslant 1,$$

and

$$h_X(x) = 1.0746 + 0.8955x - 3.5821x^2 + 2.6866x^3, \quad 0 \leqslant x \leqslant 1.$$

These two polynomial functions are non-negative and integrate to one in the domain $\Omega_X = [0, 1]$. Therefore, they are valid densities in Ω_X. We used an acceptance–rejection sampling algorithm to generate datasets with different sample sizes ($N = 100, 1000, 10\,000, 100\,000, 1\,000\,000$) from $g_X(x)$ and $h_X(x)$. We learned MoP approximations from the datasets by applying Algorithm 1. We considered different values for the order $r_X = 2, 3, 4, 5$ and the number of intervals $L_X = 1, \ldots, 10$ and selected the MoP with the highest BIC score. Ten independent repetitions were performed for each polynomial function ($g_X(x)$ and $h_X(x)$) and each sample size N. Table 5 shows the mean and the standard deviation of the absolute value of the difference between the coefficients b_i of the true polynomials ($g_X(x)$ and $h_X(x)$) and the coefficients $\hat{b}_i$, $i = 0, \ldots, r_X - 1$, of the polynomial functions in the MoP approximations. We observe that both the means and the standard deviations decrease as we consider higher sample sizes N. This reduction is clearer for the coefficients of higher order monomials b_i, $i > 1$. These coefficients are the most important because they control the shape of the polynomial functions. For instance, the MoP approximations computed for the first repetition with $N = 1\,000\,000$ are

Table 5
Accuracy of the estimates of the polynomial coefficients of the MoPs learned with Algorithm 1. The table shows the mean and the standard deviation of the absolute value of the difference between the true coefficients b_i of the polynomials and the coefficients $\hat{b}_i$ of the MoP approximations learned from data. Each coefficient b_i corresponds to the monomial term x^{i-1}, $i = 1, \dots, r_X$, in the polynomial.

	$N = 100$	$N = 1000$	$N = 10\,000$	$N = 100\,000$	$N = 1\,000\,000$
			$g_X(x) = 2.25 - 7.5x + 7.5x^2, 0 \leqslant x \leqslant 1$		
$\lvert b_0 - \hat{b}_0\rvert$	1.8962 ± 1.7411	0.5547 ± 1.1966	0.0636 ± 0.0618	0.0181 ± 0.0105	0.011 ± 0.0059
$\lvert b_1 - \hat{b}_1\rvert$	6.7028 ± 3.8739	1.8805 ± 3.0029	0.2435 ± 0.1909	0.0767 ± 0.0526	0.0586 ± 0.0238
$\lvert b_2 - \hat{b}_2\rvert$	6.4774 ± 2.3707	1.8563 ± 2.4548	0.2321 ± 0.1444	0.0714 ± 0.0547	0.0552 ± 0.0242
			$h_X(x) = 1.0746 + 0.8955x - 3.5821x^2 + 2.6866x^3, 0 \leqslant x \leqslant 1$		
$\lvert b_0 - \hat{b}_0\rvert$	0.0653 ± 0.0482	0.3546 ± 0.7428	0.3612 ± 0.4977	0.2747 ± 0.4189	0.0141 ± 0.0039
$\lvert b_1 - \hat{b}_1\rvert$	1.1466 ± 0.1293	1.3046 ± 0.6223	1.6654 ± 2.0572	1.3534 ± 1.7715	0.0815 ± 0.0263
$\lvert b_2 - \hat{b}_2\rvert$	3.5821 ± 0.0000	3.5821 ± 0.0000	2.9182 ± 2.4737	2.3931 ± 2.3841	0.0689 ± 0.0544
$\lvert b_3 - \hat{b}_3\rvert$	2.6866 ± 0.0000	2.6866 ± 0.0000	2.007 ± 1.0685	1.5805 ± 1.3263	0.0353 ± 0.0143

$$\varphi_{g,X}(x) = \begin{cases} 2.2402 - 7.4484x + 7.4547x^2, & 0 \leqslant x \leqslant 0.5, \\ 2.2344 - 7.4251x + 7.4315x^2, & 0.5 \leqslant x \leqslant 1, \end{cases}$$

for the $g_X(x)$ polynomial density, and

$$\varphi_{h,X}(x) = 1.0687 + 0.9352x - 3.5967x^2 + 2.6505x^3, \quad 0 \leqslant x \leqslant 1,$$

for the $h_X(x)$ polynomial density. We observe that the selected MoP approximation $\varphi_{g,X}(x)$ for the polynomial density $g_X(x)$ has two pieces ($L_X = 2$) instead of only one as it would be expected. However, both pieces have very similar coefficients to the true polynomial $g_X(x)$. We could use this kind of comparisons to simplify the MoP approximations obtained with Algorithm 1. On the other hand, the MoP approximation $\varphi_{h,X}(x)$ has only one piece, and its polynomial coefficients are very close to the coefficients of the true polynomial $h_X(x)$. We empirically conclude that the proposed Algorithm 1 asymptotically converges to the true probability density functions for these two polynomial densities, $g_X(x)$ and $h_X(x)$. Therefore, Algorithm 1 is able to retrieve these true polynomial models from the data.

4.1.5. Results for multidimensional datasets

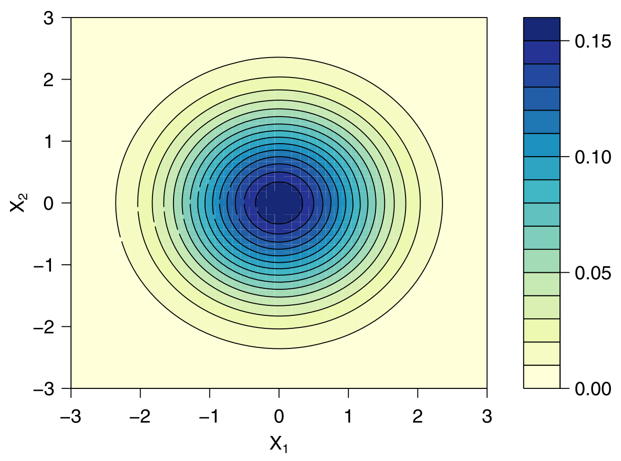
We analyzed the behavior of Algorithm 2 for learning MoP approximations of two-dimensional densities from data (see Table 1). For each probability density, we sampled ten datasets for each sample size $N = 50, 100, 500, 1000$. To reduce the number of combinations of parameters to try, we only considered MoPs that had the same order and number of intervals for each dimension, i.e., $r_{X_1} = r_{X_2} = r_X$ and $L_{X_1} = L_{X_2} = L_X$. The values considered for these parameters were $r_X = 2, \dots, 5$ and $L_X = 1, \dots, 10$. For each combination of values of these parameters, we applied Algorithm 2 to learn a MoP approximation of the two-dimensional density from each of the ten training sets. As in the one-dimensional scenario, the multidimensional Taylor series expansion cannot be used unless the mathematical expression of the true multidimensional joint density is known, so it is not applicable to learning from data. On the other hand, Lagrange interpolation and MoTBF estimation are not straightforward tasks in multidimensional spaces (see Section 4.1.3).

Fig. 4 shows contour plots of the MoP approximations with the highest BIC score of one of the training sets and the respective true densities. Table 6 reports the values of the performance measures for the two-dimensional MoP approximations learned using B-splines. For the `Gauss2d` dataset, the proposed method yielded good results even with very small sample sizes ($N = 50, 100$), as the low KL divergence, MSE and MAE values show. On the other hand, the `Mix2d` dataset is more complex, and the algorithm needed more samples ($N = 500, 1000$) to yield good approximations to the true probability density. As expected, better approximations are obtained as we increase the training sample size. Regarding the complexity of the approximations, we observe that the MoPs frequently have low orders (r_X), whereas the number of parameters (#par) and the number of intervals (L_X) in each dimension increases with the sample size (N). We also observe that more parameters (#par) and intervals (L_X) are necessary to find MoPs approximations for the `Mix2d` dataset than for the `Gauss2d` dataset because the former is more complex than the latter.

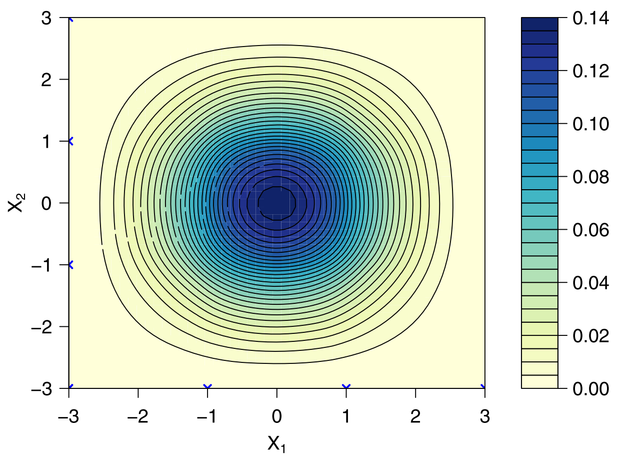
We also show an example of a three-dimensional MoP learned with Algorithm 2. We used a dataset with $N = 1000$ observations sampled from the `Mix3d` density in Table 1. Fig. 5 shows the contour plots of the true density and the MoP approximation for the domain $\Omega_{X_1} \times \Omega_{X_2}$ and three different values of X_3. We can see that the MoP approximation clearly replicates the two modes and is similar to the true multidimensional density.

4.1.6. Rmop: An R package for multidimensional mixture of polynomials learning from data

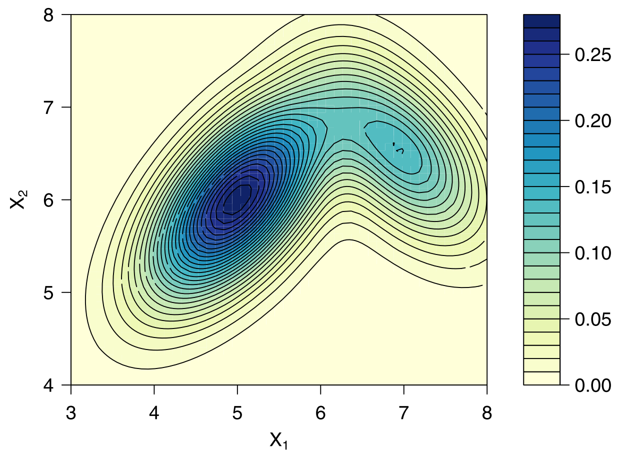
The proposed Algorithms 1 and 2 have been implemented in a freely available `R` package called `Rmop`. The package offers methods for learning multidimensional MoPs from data using B-spline interpolation. Also, it includes methods for managing MoPs, e.g., operations (sum, product, integration, marginalization, etc.), computation of statistics (mean, variance and covariance), comparison (KL divergence, MSE and MAE), plotting, etc. The package is available at http://cig.fi.upm.es/index.php/members/151-rmop.



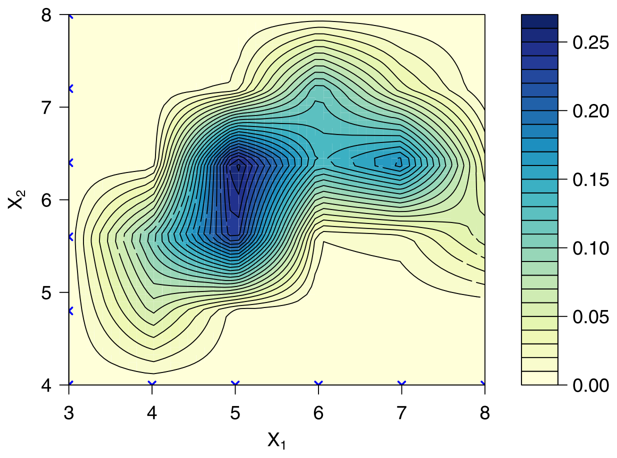
(a) Gauss2d. True density



(b) Gauss2d. MoP approximation



(c) Mix2d. True density



(d) Mix2d. MoP approximation

Fig. 4. Contour plots of the two-dimensional MoP approximations learned from a training dataset of $N = 1000$ observations. The true densities used to generate the training datasets are shown in (a) and (c). The MoPs with the highest BIC score learned with Algorithm 2 are shown in (b) and (d).

Table 6
Evaluation of the two-dimensional MoPs learned from data using B-splines (Algorithm 2). The MoP with the highest BIC score was selected for each of the ten repetitions, and the mean values of the performance measures were reported: order of the polynomials (r_X), number of pieces (L_X), number of free parameters (#par), Kullback–Leibler divergence (KL), mean squared error (MSE) and maximum absolute error (MAE).

	$N = 50$		$N = 100$		$N = 500$		$N = 1000$	
	Gauss2d	Mix2d	Gauss2d	Mix2d	Gauss2d	Mix2d	Gauss2d	Mix2d
r_X	2	2	2	2	2.6	2	2.9	2
L_X	2	2	2	2.3	2.8	4.4	3.1	5.1
#par	9	9	9	11.1	20.2	29.8	25	37.3
KL	0.1182	0.3226	0.1127	0.2782	0.0433	0.1103	0.0116	0.0517
MSE	0.0010	0.0049	0.0010	0.0050	0.0003	0.0020	0.0001	0.0007
MAE	0.0583	0.1531	0.0574	0.1625	0.0364	0.1079	0.0276	0.0673

4.2. Non-parametric Bayesian classifiers using MoP density estimation

In this section, we illustrate how to use the proposed methods for learning one-dimensional and multidimensional MoPs from data as a non-parametric density estimation technique in Bayesian classifiers. We retrieved 14 datasets from the UCI [55] and KEEL [56] repositories. We deleted the first variable in the ion dataset because it was discrete, i.e., it only took values 0 or 1. The values of the predictive variables were scaled to the domain $\Omega_{X_i} = [0, 1], i = 1, \ldots, n$. Table 7 shows the main features of the final datasets used in the experimentation.

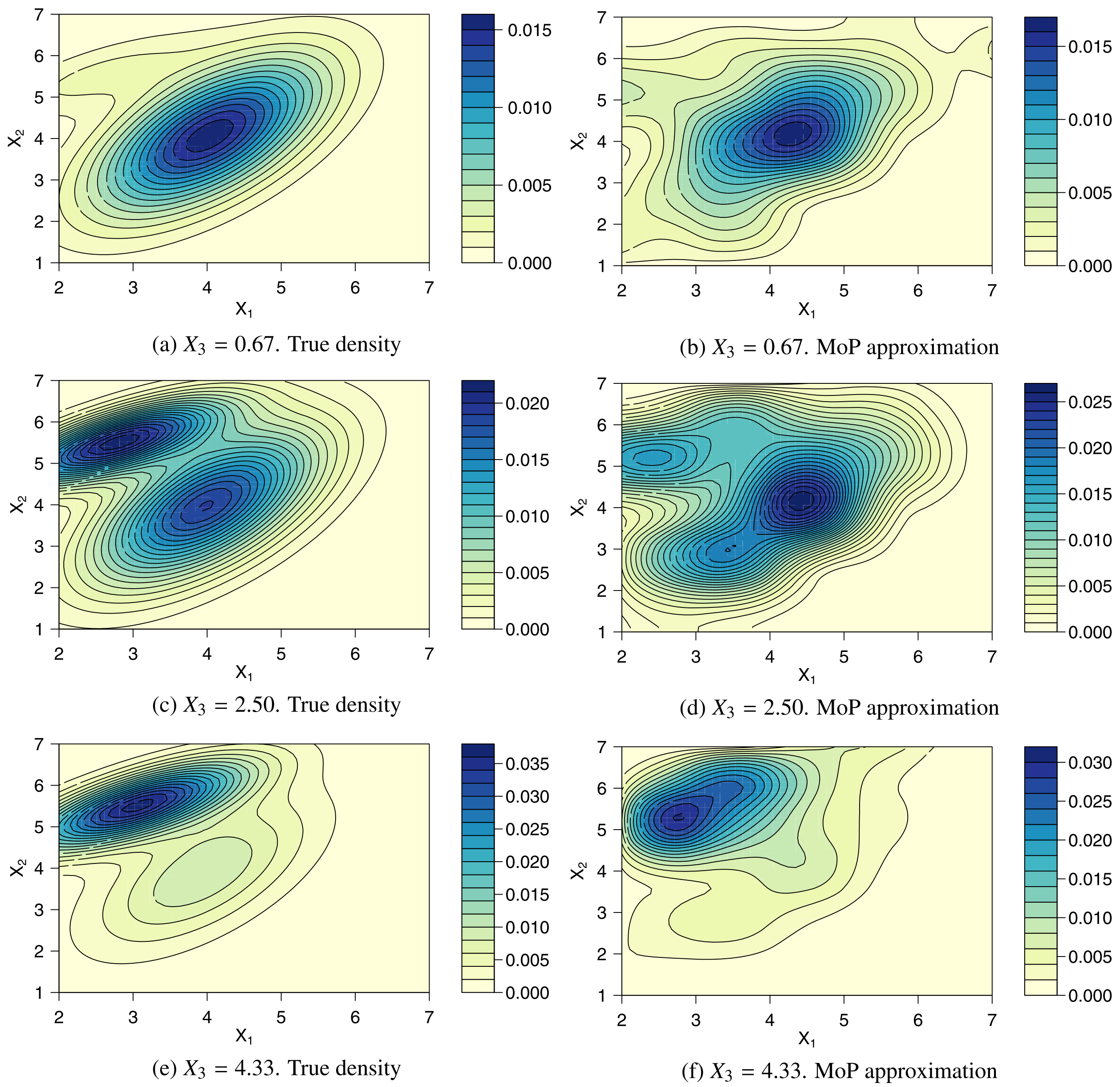


Fig. 5. Contour plots of the three-dimensional MoP approximation learned from a training dataset of $N = 1000$ observations sampled from the `Mix3d` dataset. The true densities are shown in (a), (c) and (e). The respective densities of the MoP approximation with $r_{X_1} = r_{X_2} = r_{X_3} = 3$ and $L_{X_1} = L_{X_2} = L_{X_3} = 5$ are shown in (b), (d) and (f) for different values of X_3.

NB [40] and TAN [30] classifiers were induced for each dataset. The NB classifier is one of the most popular methods for supervised classification. It assumes the predictive variables to be conditionally independent given the class. Although the conditional independence assumption in NB classifiers seems very restrictive, it has shown competitive results in a number of real-world problems [57,30]. Several extensions have been proposed in the literature that relax the conditional independence assumption in NB classifiers. The TAN classifier allows relationships between pairs of predictive variables in the network. At the same time, it controls the complexity of the learning algorithm by imposing a tree structure over the subgraph structure of the predictive variables **X**.

In NB and TAN classifiers, the probability of the class labels was modeled as a categorical distribution $p_C(c)$, $c \in \Omega_C$. On the other hand, the conditional density of the predictive variables **X** given the class label $C = c$ was modeled using six different density estimation techniques, yielding twelve Bayesian classifiers for comparison:

- `NBMoP` and `TANMoP` use MoPs for non-parametric density estimation. The MoPs $\varphi_{X_i|C}(x_i|c)$, $i = 1, \ldots, n$, were approximated using Algorithm 1 for NB classifiers. We considered different number of pieces $L_X = 1, \ldots, 10$ and or-

Table 7
Datasets used in the Bayesian classifier experiments.

Name	Num. instances N	Num. pred. variables n	Num. class labels K
appendicitis	106	7	2
fourclass	862	2	2
glass	214	9	2
haberman	306	3	2
ion	351	32	2
iris	150	4	3
liver	341	6	2
newthyroid	215	5	3
phoneme	5404	5	2
svmguide1	7089	4	2
vehicle	846	18	4
waveform	5000	21	3
wdbc	569	30	2
wine	178	13	3

ders $r_X = 2, \dots, 5$, and we chose the MoP approximation with the highest BIC score (Eq. (8)). Similarly, the MoPs $\varphi_{X_i X_j|C}(x_i, x_j|c)$, $i \neq j$, were approximated using Algorithm 2 for TAN classifiers. In TAN classifiers, we computed $\varphi_{X_i|C}(x_i|c)$ by marginalizing out X_j from $\varphi_{X_i X_j|C}(x_i, x_j|c)$. Note that we did not find explicit MoP approximations of the conditional densities $\varphi_{X_i|X_j C}(x_i|x_j, c)$ for TAN classifiers. Instead, we computed the values of the conditional densities by dividing the evaluation of the MoP $\varphi_{X_i X_j|C}(x_i, x_j|c)$ by the evaluation of the MoP $\varphi_{X_j|C}(x_j|c)$. We only considered two-dimensional MoPs $\varphi_{X_i|X_j C}(x_i|x_j, c)$ that had the same order and number of intervals for each dimension, i.e., $r_{X_i} = r_{X_j} = r_X$ and $L_{X_i} = L_{X_j} = L_X$. The values considered for these parameters were $r_X = 2, \dots, 5$ and $L_X = 1, \dots, 10$. The MoP approximation with highest BIC score was selected (see Section 3.4).

- NBKernel and TANKernel use a Gaussian kernel density estimation technique as proposed by Pérez et al. [37]. The parameters and the densities of the one-dimensional and the multidimensional Gaussian kernels were computed using the ks [49] and KernSmooth [58] R packages. A normal scale bandwidth was used for computing the kernel density estimates.
- NBGauss and TANGauss use Gaussian distributions to model the conditional densities as in a conditional linear Gaussian network [32].
- NBFI and TANFI use Fayyad and Irani's supervised discretization method [59] to discretize the continuous variables. ML estimators with Laplace correction were computed to fill in the (conditional) probability tables in the NB and TAN classifiers.
- NBEF5 and TANEF5 use an equal-frequency unsupervised discretization technique. Each variable is discretized into five bins. The parameters of the probability distributions were estimated using ML with Laplace correction.
- NBEF10 and TANEF10 discretize each variable into ten equal-frequency bins, and the (conditional) probability tables are filled in using the ML estimators with Laplace correction.

Once the probability distributions have been estimated, a new instance $\mathbf{x}$ is classified by applying the maximum a posteriori rule: $c^* = \arg\max_{c \in \Omega_C} p_C(c) f_{\mathbf{X}|C}(\mathbf{x}|c)$, where $f_{\mathbf{X}|C}(\mathbf{x}|c)$ depends on the probability distribution used for modeling the predictive variables $\mathbf{X}$ and factorizes according to the graphical structure of the classifier (either NB or TAN).

Table 8 shows the mean accuracy achieved by each classifier in each dataset estimated using a stratified 10-fold cross-validation. The results show a clear example of the "no free lunch" theorem [60], as there is no algorithm that significantly outperforms the others in all the datasets. Kernel density estimation (NBKernel and TANKernel) achieves the best results in five out of fourteen datasets, whereas the parametric Gaussian Bayesian classifiers (NBGauss and TANGauss) and the proposed classifiers using MoPs (NBMoP and TANMoP) obtain the best result in three datasets. The null hypothesis of equal performance of all algorithms could not be rejected at a significance level $\alpha = 0.05$ using Friedman's test (p-value $= 0.0524$) [61]. On the other hand, Iman and Davenport's test [62] rejected the null hypothesis of equal performance of all algorithms (p-value $= 0.0456$). However, we found no significant differences when we studied all pairwise comparisons between algorithms using Nemenyi, Holm and Shaffer post-hoc tests [63]. Similarly, we found no significant differences between the algorithms when we analyzed NB classifiers and TAN classifiers separately. Most of the datasets included in the study had few instances for the different class values. This makes it difficult for the proposed methods to obtain MoPs that can model complex probability distributions, as we saw when fitting MoPs to samples from the Mix2d dataset (Section 4.1.5). Additionally, for small samples, the BIC score tends to select simple MoP models with fewer intervals.

NBMoP's accuracy was higher than NBKernel's accuracy in four datasets, whereas NBKernel outperformed NBMoP in nine datasets. However, we found no significant differences between the two methods when we considered all the datasets using Friedman's test and Iman and Davenport's test. Additionally, we compared MoPs with kernel density estimation for each dataset independently. We applied a non-parametric paired Wilcoxon signed-rank test using the accuracy of the classifiers in the 10 folds of the cross-validation procedure. NBMoP significantly outperformed NBKernel in two datasets (vehicle and waveform). On the other hand, TANMoP significantly outperformed TANKernel in only one

Table 8
Mean accuracy of the classifiers estimated using a stratified 10-fold cross-validation. The best result for each dataset is highlighted in boldface.

	NBMoP	NBKernel	NBGauss	NBFI	NBEF5	NBEF10
appendicitis	84.82	84.82	84.82	83.91	82.00	85.00
fourclass	85.49	83.87	75.41	78.18	76.91	83.41
glass	92.51	91.10	90.61	91.10	90.63	92.51
haberman	72.91	73.55	74.85	72.24	73.88	**75.85**
ion	86.06	90.87	81.17	89.16	88.59	88.87
iris	94.67	96.00	95.33	93.33	93.33	94.00
liver	62.76	67.45	57.18	57.75	63.94	61.29
newthyroid	94.87	**96.77**	**96.77**	94.89	95.41	95.87
phoneme	77.90	78.11	76.02	77.20	77.09	77.07
svmguide1	95.67	95.82	93.13	96.42	96.01	96.25
vehicle	64.06	60.51	46.21	61.22	58.63	63.23
waveform	81.08	80.70	80.90	80.78	80.82	80.64
wdbc	94.38	94.55	93.49	94.55	93.50	94.73
wine	96.60	98.33	98.33	98.33	98.33	96.67
	TANMoP	TANKernel	TANGauss	TANFI	TANEF5	TANEF10
appendicitis	82.91	81.91	79.09	84.91	**86.64**	80.91
fourclass	**98.84**	96.64	79.70	86.76	90.83	96.87
glass	90.17	92.47	92.49	92.03	91.13	**93.94**
haberman	**75.85**	74.18	74.84	72.24	69.97	73.52
ion	**92.29**	91.45	90.87	91.72	92.29	91.72
iris	88.00	**97.33**	**97.33**	93.33	94.00	91.33
liver	60.14	**70.69**	59.22	57.75	64.82	59.84
newthyroid	91.67	95.84	95.84	93.51	93.53	92.62
phoneme	80.53	80.31	77.87	80.46	80.64	**83.22**
svmguide1	95.70	95.97	93.75	**96.67**	96.23	96.39
vehicle	61.46	**77.90**	75.89	72.45	71.88	72.11
waveform	80.96	80.74	**82.28**	81.60	81.22	80.96
wdbc	94.02	95.25	95.43	94.37	94.91	**95.44**
wine	93.89	**100.00**	99.44	96.11	97.22	93.30

dataset (fourclass), whereas TANKernel significantly outperformed TANMoP in four datasets (iris, liver, vehicle and wine). We could not find significant differences between MoP-based and kernel-based classifiers in most of the datasets. Therefore, we can conclude that NB and TAN classifiers using MoPs perform competitively against kernel-based NB and TAN classifiers.

NB and TAN classifiers with Gaussian densities and the three discretization algorithms (FI, EF5 and EF10) yielded good performances. Discretization can help to reduce the noise in a dataset, specially when few training instances are available. We could not find significant differences between the Bayesian classifiers using the three discretization algorithms. This conforms to the results reported in [64], showing that there is no discretization method that yields better Bayesian classifiers in terms of accuracy for an extensive set of problems.

4.3. Comparison of evaluation times of MoPs and kernel density estimation

We studied the evaluation time of the non-parametric density estimation techniques, i.e., MoPs and kernel-based density estimation. Other density estimation techniques such as discretization or assuming Gaussian densities were considered parametric alternatives and are expected to yield shorter evaluation times. Fig. 6 shows the evaluation time of MoPs and kernel density estimation for two artificial datasets (see Table 1): the one-dimensional Gauss dataset and the two-dimensional Gauss2d dataset. Different training and test sizes were considered, and the evaluation times were averaged over 10 repetitions. Algorithms 1 and 2 were applied to learn the MoPs from the data and the BIC score was used to find appropriate values for the order and the number of intervals of the MoPs. We can see that the evaluation times for MoPs (solid lines) were almost constant, independently of the sizes of the training and test sets. On the contrary, the evaluation time for the kernel density estimation technique (dashed lines) increased with both the training and test sizes. We can also see that the evaluation time for kernels increased from the one-dimensional (Fig. 6(a)) to the two-dimensional scenario (Fig. 6(b)). Additionally, MoPs are more efficient than kernels regarding storage. A MoP provides an explicit model of a probability density. Therefore, it only needs to store the coefficients of the polynomials and the limits of the intervals/hyperrectangles for each piece. On the other hand, kernels do not provide an explicit model, so they need to save and analyze the complete training dataset to estimate the density of a new observation.

5. Conclusion

We have presented a method for learning MoP approximations of the probability density underlying a dataset using B-spline interpolation. Both one-dimensional and multidimensional MoP approximations of probability densities were learned

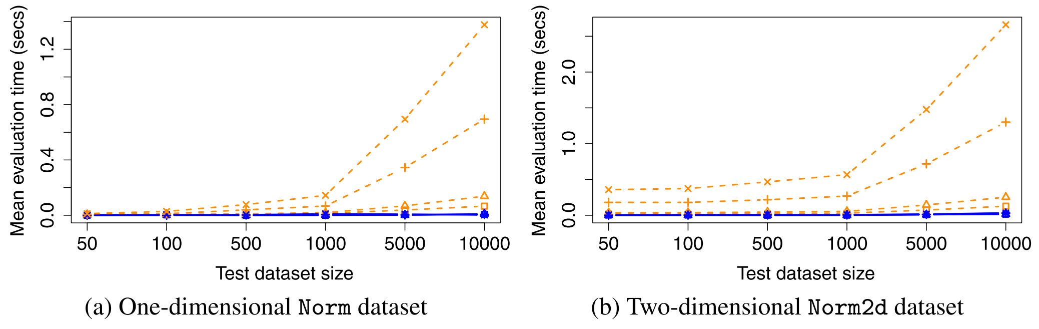


Fig. 6. Comparison of evaluation time of MoPs (solid lines) with kernel density estimation (dashed lines) for: (a) the one-dimensional `Norm` dataset and (b) the two-dimensional `Norm2d` dataset (see Table 1). One line is shown for each training dataset size N: 50 (□), 100 (△), 500 (+) and 1000 (×). The mean evaluation time (in seconds) over ten repetitions is shown. For each repetition, the MoP with the highest BIC score learned with Algorithms 1 and 2 was considered.

from data. A MoP was approximated as a linear combination of B-splines, since B-splines can be written as MoPs, and MoPs are closed under multiplication and addition. The mixing coefficients of the linear combination were found using a maximum likelihood approach. Thus we were able to perform model selection in a principled way by using a penalized likelihood criterion like the BIC score. Appropriate values for the order of the polynomials and the number of pieces of the MoPs were selected based on the BIC score of the MoPs. MoPs learned with B-splines have a number of advantages over other methods proposed in the literature, i.e., they are continuous, non-negative and integrate to one. These properties are important in some settings, e.g., for model interpretation or for performing inference tasks. MoPs learned with B-spline interpolation were compared with MoPs using LIPs as proposed in [16] and with the MoTBF learning approach in [17]. Artificial datasets were used to evaluate the proposed method for learning MoP approximations. MoPs learned using B-spline interpolation outperformed MoPs learned with LIPs according to the Kullback–Leibler divergence, the mean squared error and the maximum absolute error between the MoPs and the true generating distributions. Also, the proposed method yielded MoPs with better generalization behavior than LIPs according to the log-likelihood and BIC values computed in the test datasets. MoPs learned with B-splines yielded competitive results against MoTBFs. The proposed approach yielded MoPs with several pieces and low-order polynomials, whereas the MoTBF approximations had a single high-order polynomial. In general, we would like to obtain low values for both the number of pieces and the order of the MoPs. Therefore, the two proposals represent alternative approaches to the problem of handling the model complexity.

We also studied the use of MoPs as a non-parametric density estimation technique for Bayesian classifiers for the first time. In particular, we studied and implemented two well-known Bayesian classifiers: the naive Bayes classifier and the tree-augmented naive Bayes classifier. NB and TAN classifiers using MoPs yielded competitive results against other state-of-the-art Bayesian classifier approaches. MoPs offer some advantages over kernels as non-parametric density estimators. First, MoPs provide an explicit model of the generating probability density. Second, the evaluation time for MoPs is shorter than for kernel density estimation. Therefore, Bayesian classifiers using MoPs have faster classification times than those using kernel density estimation. Additionally, MoPs only need to store the model parameters (coefficients of the polynomials and limits of the intervals/hyperrectangles), whereas kernel density estimation has to save the complete training dataset. On the other hand, training time is longer for MoPs because the approach includes parameter estimation and model selection, although Eq. (5) converges in few iterations [27]. Gaussian and discretization-based classifiers also yielded competitive results compared with the other methods. The parametric assumption of Gaussian densities may not hold in some settings. On the other hand, MoPs are a more flexible approach since they provide a non-parametric density estimation technique that can model any probability density without the need for discretization.

Future work will study the problem of finding the limits of the intervals/hyperrectangles A_{l_X} when they do not have the same width (non-uniform B-splines). Finding the best knot sequence given a dataset is expected to reduce the number of pieces necessary to find accurate MoPs. Some heuristics can be used to find a set of candidate points defining the limits of the intervals A_{l_X}, e.g., considering the local maxima, the local minima and the inflection points of the probability density function yielded good results for approximating with MTEs [13]. Then, a greedy search procedure could be used to split or merge adjacent intervals defined at these candidate points. Also, more complex methods can be found in the literature, for instance, knot density estimation [65], bootstrapping techniques [66], statistical testing [67], regularization [68], Bayesian estimation [69,70], etc.

In this paper, we evaluated conditional probability density functions by dividing the values of the MoP approximations of the joint probability density of (X_1, X_2) and the marginal probability density of the parent variable X_2: $\varphi_{X_1X_2}(x_1, x_2)/\varphi_{X_2}(x_2)$. However, we do not obtain explicit MoP approximations of conditional probability densities $\varphi_{X_1|X_2}(x_1|x_2)$. Existing methods for approximating conditional densities with MTEs or MoTBFs rely on the discretization of the conditioning variables and the estimation of a collection of marginal densities, one for each combination of the (discrete) values of the conditioning variables. This approach could be easily applied for approximating conditional densities with MoPs, where the marginal densities of the conditional variables could be found with Algorithms 1 and 2. In the future, we would like to investigate methods for learning MoP approximations of conditional densities without the need of discretization.

Regarding the use of MoPs as a non-parametric density estimation technique in supervised classification problems, extensions to more complex Bayesian classifiers will be considered, e.g., semi-naive Bayes classifiers, *k*-dependence Bayesian classifiers, etc. Here, we followed a generative approach and computed ML estimates of the mixing coefficients of the linear combination of B-splines to build the MoPs that were used in the Bayesian classifiers. However, the main goal in supervised learning problems is obtaining models that correctly classify the instances. Discriminative approaches to parameter fitting in Bayesian classifiers look for parameters which maximize the classification accuracy or, alternatively, the conditional log-likelihood of the class variable given the predictive variables, e.g., see [71,72]. We intend to investigate this discriminative approach for fitting Bayesian classifiers with MoPs in the future.

Finally, the performance of the proposed method will be compared with related approximation methods such as MoTBFs or MTEs. These comparisons can be performed at different levels: quality of the approximations, modeling power, efficiency and computational complexity of the learning algorithms, discriminative power when used inside Bayesian classifiers, etc. Also, probabilistic graphical models using MTEs have been applied in clustering [73,74] and regression [75–77] problems. We would like to study and compare the use of MoPs for solving these different machine learning problems.

Acknowledgements

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. The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Centro de Supercomputación y Visualización de Madrid (CeSViMa). P.L.L.C. is supported by a Fellowship (FPU AP2009-1772) from the Spanish Ministry of Education, Culture and Sport. A preliminary version of this paper was presented at the Sixth European Workshop on Probabilistic Graphical Models [78]. This extended version includes methods for learning multidimensional probability densities from data (Section 3.3). All the experimental results in Section 4 have been updated. It also adds a comparison between the proposed algorithms and LIPs (Section 4.1) and the study of MoPs as a density estimation technique in tree-augmented naive Bayes classifiers (Section 4.2).

References

[1] S. Siegel, N.J. Castellan, Nonparametric Statistics for the Behavioral Sciences, 2nd edition, McGraw-Hill, 1988.
[2] J.D. Gibbons, S. Chakraborti, Nonparametric Statistical Inference, 5th edition, Chapman & Hall, 2010.
[3] M.J. Fryer, A review of some non-parametric methods of density estimation, IMA J. Appl. Math. 20 (1977) 335–354.
[4] Y. Yang, G.I. Webb, X. Wu, Discretization methods, in: O. Maimon, L. Rokach (Eds.), Data Mining and Knowledge Discovery Handbook, Springer, 2010, pp. 101–116.
[5] S. García, J. Luengo, J.A. Sáez, V. López, F. Herrera, A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning, IEEE Trans. Knowl. Data Eng. 25 (2013) 734–750.
[6] S.-T. Chiu, A comparative review of bandwidth selection for kernel density estimation, Stat. Sin. 6 (1996) 129–146.
[7] M.C. Jones, J.S. Marron, S.J. Sheather, A brief survey of bandwidth selection for density estimation, J. Am. Stat. Assoc. 91 (1996) 401–407.
[8] S. Moral, R. Rumí, A. Salmerón, Mixtures of truncated exponentials in hybrid Bayesian networks, in: S. Benferhat, P. Besnard (Eds.), Proceedings of the 6th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU2001), in: Lecture Notes in Computer Science, vol. 2143, Springer, 2001, pp. 145–167.
[9] P.P. Shenoy, J.C. West, Inference in hybrid Bayesian networks using mixtures of polynomials, Int. J. Approx. Reason. 52 (2011) 641–657.
[10] H. Langseth, T.D. Nielsen, R. Rumí, A. Salmerón, Mixtures of truncated basis functions, Int. J. Approx. Reason. 53 (2012) 212–227.
[11] P.P. Shenoy, G. Shafer, Axioms for probability and belief functions propagation, in: R.D. Shachter, T.S. Levitt, L.N. Kanal, J.F. Lemmer (Eds.), Proceedings of the 4th Annual Conference on Uncertainty in Artificial Intelligence (UAI1988), North-Holland, 1990, pp. 169–198.
[12] B. Cobb, P.P. Shenoy, R. Rumí, Approximating probability density functions with mixtures of truncated exponentials, Stat. Comput. 16 (2006) 193–308.
[13] R. Rumí, A. Salmerón, S. Moral, Estimating mixtures of truncated exponentials in hybrid Bayesian networks, Test 15 (2006) 397–421.
[14] V. Romero, R. Rumí, A. Salmerón, Learning hybrid Bayesian networks using mixtures of truncated exponentials, Int. J. Approx. Reason. 42 (2006) 54–68.
[15] H. Langseth, T.D. Nielsen, R. Rumí, A. Salmerón, Parameter estimation and model selection for mixtures of truncated exponentials, Int. J. Approx. Reason. 51 (2010) 485–498.
[16] P.P. Shenoy, Two issues in using mixtures of polynomials for inference in hybrid Bayesian networks, Int. J. Approx. Reason. 53 (2012) 847–866.
[17] H. Langseth, T.D. Nielsen, R. Rumí, A. Salmerón, Learning mixtures of truncated basis functions from data, in: A. Cano, M. Gómez-Olmedo, T.D. Nielsen (Eds.), Proceedings of the 6th European Workshop on Probabilistic Graphical Models (PGM2012), 2012, pp. 163–170.
[18] H. Langseth, T.D. Nielsen, R. Rumí, A. Salmerón, Inference in hybrid Bayesian networks with mixtures of truncated basis functions, in: A. Cano, M. Gómez-Olmedo, T.D. Nielsen (Eds.), Proceedings of the 6th European Workshop on Probabilistic Graphical Models (PGM2012), 2012, pp. 171–178.
[19] H. Langseth, T.D. Nielsen, R. Rumí, A. Salmerón, Maximum likelihood learning of conditional MTE distributions, in: C. Sossai, G. Chemello (Eds.), Proceedings of the 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU2009), in: Lecture Notes in Computer Science, vol. 5590, Springer, 2009, pp. 240–251.
[20] V.A. Epanechnikov, Non-parametric estimation of a multivariate probability density, Theory Probab. Appl. 14 (1969) 153–158.
[21] J.-N. Hwang, S.-R. Lay, A. Lippman, Nonparametric multivariate density estimation: A comparative study, IEEE Trans. Signal Process. 42 (1994) 2795–2810.
[22] J.E. Chacón, T. Duong, M. Wand, Asymptotics for general multivariate kernel density derivative estimators, Stat. Sin. 21 (2011) 807–840.
[23] V.M. Panaretos, K. Konis, Nonparametric construction of multivariate kernels, J. Am. Stat. Assoc. 107 (2012) 1085–1095.
[24] R. Jiroušek, P.P. Shenoy, Compositional models in valuation-based systems, Int. J. Approx. Reason. (2013), in press.
[25] P.P. Shenoy, A valuation-based language for expert systems, Int. J. Approx. Reason. 3 (1989) 383–411.
[26] Z. Zong, K. Lam, Estimation of complicated distributions using B-spline functions, Struct. Saf. 20 (1998) 341–355.
[27] Z. Zong, Information-Theoretic Methods for Estimating Complicated Probability Distributions, Elsevier, 2006.
[28] 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, Q. Appl. Math. 4 (1946) 45–99.

[29] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd edition, John Wiley & Sons, 2001.
[30] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Mach. Learn. 29 (1997) 131–163.
[31] S.L. Lauritzen, N. Wermuth, Graphical models for associations between variables, some of which are qualitative and some quantitative, Ann. Stat. 17 (1989) 31–57.
[32] A. Pérez, P. Larrañaga, I. Inza, Supervised classification with conditional Gaussian networks: Increasing the structure complexity from naive Bayes, Int. J. Approx. Reason. 43 (2006) 1–25.
[33] J. Dougherty, R. Kohavi, M. Sahami, Supervised and unsupervised discretization of continuous features, in: A. Prieditis, S.J. Russell (Eds.), Proceedings of the 12th International Conference on Machine Learning (ICML1995), Morgan Kaufmann, 1995, pp. 194–202.
[34] C.-N. Hsu, H.-J. Huang, T.-T. Wong, Implications of the Dirichlet assumption for discretization of continuous variables in naive Bayesian classifiers, Mach. Learn. 53 (2003) 235–263.
[35] Y. Yang, G.I. Webb, On why discretization works for naive-Bayes classifiers, in: T.D. Gedeon, L.C.C. Fung (Eds.), Proceedings of the 16th Australian Conference on Artificial Intelligence, in: Lecture Notes in Computer Science, vol. 2903, Springer, 2003, pp. 440–452.
[36] G.H. John, P. Langley, Estimating continuous distributions in Bayesian classifiers, in: P. Besnard, S. Hanks (Eds.), Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence (UAI1995), Morgan Kaufmann, 1995, pp. 338–345.
[37] A. Pérez, P. Larrañaga, I. Inza, Bayesian classifiers based on kernel density estimation: Flexible classifiers, Int. J. Approx. Reason. 50 (2009) 341–362.
[38] M.J. Flores, J.A. Gámez, A.M. Martínez, A. Salmerón, Mixture of truncated exponentials in supervised classification: Case study for the naive Bayes and averaged one-dependence estimators classifiers, in: S. Ventura, A. Abraham, K.J. Cios, C. Romero, F. Marcelloni, J.M. Benítez, E.L.G. Galindo (Eds.), Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA2011), IEEE, 2011, pp. 593–598.
[39] I. Flesch, A. Fernández, A. Salmerón, Incremental supervised classification for the MTE distribution: A preliminary study, in: I. Rojas, H. Pomares (Eds.), Proceedings of the 2nd Simposio de Inteligencia Computacional (SICO2007), Thomson, 2007, pp. 217–224.
[40] M. Minsky, Steps toward artificial intelligence, Proc. Inst. Radio Eng. 49 (1961) 8–30.
[41] I. Faux, M. Pratt, Computational Geometry for Design and Manufacture, Wiley, 1979.
[42] C. de Boor, A Practical Guide to Splines, Springer-Verlag, 1978.
[43] H. Prautzsch, W. Boehm, M. Paluszny, Bézier and B-Spline Techniques, Springer-Verlag, 2002.
[44] M.H. Stone, The generalized Weierstrass approximation theorem, Math. Mag. 21 (1948) 237–254.
[45] G. Schwarz, Estimating the dimension of a model, Ann. Stat. 6 (1978) 461–464.
[46] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1951) 79–86.
[47] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0, 2012.
[48] S.G. Johnson, B. Narasimhan, Cubature: Adaptive multivariate integration over hypercubes, R package version 1.1-1, 2011.
[49] T. Duong, ks: Kernel smoothing, R package version 1.8.11, 2012.
[50] I.K. Glad, N.L. Hjort, N.G. Ushakov, Correction of density estimators that are not densities, Scand. J. Stat. 30 (2003) 415–427.
[51] M. Gasca, T. Sauer, Polynomial interpolation in several variables, Adv. Comput. Math. 12 (2000) 377–410.
[52] L. Bos, S. De Marchi, M. Vianello, Y. Xu, Bivariate Lagrange interpolation at the Padua points: The ideal theory approach, Numer. Math. 108 (2007) 47–57.
[53] L.A. Harris, Bivariate Lagrange interpolation at the Chebyshev nodes, Proc. Am. Math. Soc. 138 (2010) 4447–4453.
[54] M. Caliari, S. De Marchi, M. Vianello, Hyperinterpolation in the cube, Comput. Math. Appl. 55 (2008) 2490–2497.
[55] K. Bache, M. Lichman, UCI machine learning repository, http://archive.ics.uci.edu/ml, 2013.
[56] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, J. Mult.-Valued Log. Soft Comput. 17 (2011) 255–287.
[57] P. Domingos, M. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss, Mach. Learn. 29 (1997) 103–130.
[58] M. Wand, B. Ripley, KernSmooth: Functions for kernel smoothing for Wand & Jones, "Kernel smoothing", 2012, R package version 2.23-8, 1995.
[59] U.M. Fayyad, K.B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: R. Bajcsy (Ed.), Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI1993), Morgan Kaufmann, 1993, pp. 1022–1027.
[60] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82.
[61] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32 (1937) 675–701.
[62] R.L. Iman, J.M. Davenport, Approximations of the critical region of the Friedman statistic, Commun. Stat., Theory Methods 9 (1980) 571–595.
[63] S. García, F. Herrera, An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, J. Mach. Learn. Res. 9 (2008) 2677–2694.
[64] M.J. Flores, J.A. Gámez, A.M. Martínez, J.M. Puerta, Handling numeric attributes when comparing Bayesian network classifiers: Does the discretization method matter?, Appl. Intell. 34 (2011) 372–385.
[65] A. Crampton, A.B. Forbes, Spline approximation using knot density functions, in: A. Iske, J. Levesley (Eds.), Algorithms for Approximation, Springer, 2007, pp. 249–258.
[66] F. Leitenstorfer, G. Tutz, Knot selection by boosting techniques, Comput. Stat. Data Anal. 51 (2007) 4605–4621.
[67] C.J. Stone, M.H. Hansen, C. Kooperberg, Y.K. Truong, Polynomial splines and their tensor products in extended linear modeling, Ann. Stat. 25 (1997) 1371–1470.
[68] M.R. Osborne, B. Presnell, B.A. Turlach, Knot selection for regression splines via the LASSO, in: S. Weisberg (Ed.), Dimension Reduction, Computational Complexity, and Information, in: Computing Science and Statistics, vol. 30, Interface Foundation of North America, 1998, pp. 44–49.
[69] D.G.T. Denison, B.K. Mallick, A.F.M. Smith, Automatic Bayesian curve fitting, J. R. Stat. Soc., Ser. B, Stat. Methodol. 60 (1998) 333–350.
[70] I. DiMatteo, C.R. Genovese, R.E. Kass, Bayesian curve-fitting with free-knot splines, Biometrika 88 (2001) 1055–1071.
[71] R. Greiner, X. Su, B. Shen, W. Zhou, Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers, Mach. Learn. 59 (2005) 297–322.
[72] A.M. Carvalho, T. Roos, A.L. Oliveira, P. Myllymäki, Discriminative learning of Bayesian networks via factorized conditional log-likelihood, J. Mach. Learn. Res. 12 (2011) 2181–2210.
[73] J.A. Gámez, R. Rumí, A. Salmerón, Unsupervised naive Bayes for data clustering with mixtures of truncated exponentials, in: M. Studený, J. Vomlel (Eds.), Proceedings of the 3rd European Workshop on Probabilistic Graphical Models (PGM2006), 2006, pp. 123–130.
[74] A. Fernández, J.A. Gámez, R. Rumí, A. Salmerón, Data clustering using hidden variables in hybrid Bayesian networks, in: Proceedings of the 4th International Conference of the European Research Consortium for Informatics and Mathematics, Working Group on Computing & Statistics (ERCIM2011), 2011.
[75] A. Fernández, J.D. Nielsen, A. Salmerón, Learning Bayesian networks for regression from incomplete databases, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 18 (2010) 69–86.

[76] A. Fernández, M. Morales, A. Salmerón, Tree augmented naive Bayes for regression using mixtures of truncated exponentials: Application to higher education management, in: M.R. Berthold, J. Shawe-Taylor, N. Lavrac (Eds.), Advances in Intelligent Data Analysis VII, Proceedings of the 7th International Symposium on Intelligent Data Analysis (IDA2007), in: Lecture Notes in Computer Science, vol. 4723, Springer, 2007, pp. 59–69.
[77] M. Morales, C. Rodríguez, A. Salmerón, Selective naive Bayes for regression using mixtures of truncated exponentials, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 15 (2007) 697–716.
[78] P.L. López-Cruz, C. Bielza, P. Larrañaga, Learning mixtures of polynomials from data using B-spline interpolation, in: A. Cano, M. Gómez-Olmedo, T.D. Nielsen (Eds.), Proceedings of the 6th European Workshop on Probabilistic Graphical Models (PGM2012), 2012, pp. 211–218.