Neural Processing Letters
https://doi.org/10.1007/s11063-023-11196-2

Multivariate Density Estimation with Deep Neural Mixture Models

Edmondo Trentin[1]

Accepted: 13 February 2023

Abstract
Albeit worryingly underrated in the recent literature on machine learning in general (and, on deep learning in particular), multivariate density estimation is a fundamental task in many applications, at least implicitly, and still an open issue. With a few exceptions, deep neural networks (DNNs) have seldom been applied to density estimation, mostly due to the unsupervised nature of the estimation task, and (especially) due to the need for constrained training algorithms that ended up realizing proper probabilistic models that satisfy Kolmogorov's axioms. Moreover, in spite of the well-known improvement in terms of modeling capabilities yielded by mixture models over plain single-density statistical estimators, no proper mixtures of multivariate DNN-based component densities have been investigated so far. The paper fills this gap by extending our previous work on neural mixture densities (NMMs) to multivariate DNN mixtures. A maximum-likelihood (ML) algorithm for estimating Deep NMMs (DNMMs) is handed out, which satisfies numerically a combination of hard and soft constraints aimed at ensuring satisfaction of Kolmogorov's axioms. The class of probability density functions that can be modeled to any degree of precision via DNMMs is formally defined. A procedure for the automatic selection of the DNMM architecture, as well as of the hyperparameters for its ML training algorithm, is presented (exploiting the probabilistic nature of the DNMM). Experimental results on univariate and multivariate data are reported on, corroborating the effectiveness of the approach and its superiority to the most popular statistical estimation techniques.

Keywords Mixture of experts · Density estimation · Mixture density · Constrained deep learning · Unsupervised deep learning · Deep neural networks

1 Introduction

Let us consider an unlabeled training set $\mathcal{T} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of n independent random vectors (i.e., patterns) in a d-dimensional feature space, say $\mathbb{R}^d$. The patterns are assumed to be

✉ Edmondo Trentin
trentin@dii.unisi.it

[1] Dipartimento di Ingegneria dell'Informazione e Sc. Matematiche, Università di Siena, Via Roma n. 56, 53100 Siena, Italy

identically distributed according to an unknown probability density function (pdf) $p(\mathbf{x})$. Density estimation consists in finding a model for $p(\mathbf{x})$ based on $\mathcal{T}$. This requires exploiting the statistical knowledge on $p(\mathbf{x})$ implicitly underlying the very data sample $\mathcal{T}$. To this aim, a suitable algorithm is applied, either rooted in statistics or in machine learning. The algorithm is expected to come up with a model of $p(\mathbf{x})$ that fits the data well, in compliance with some well-defined statistical criterion (the maximum likelihood criterion [8] being possibly the most popular). Section 1.1 pinpoints the major difficulties of applying machine learning to the task. Established approaches based on neural networks are reviewed in Sect. 1.2, while the potential of using mixture density models is highlighted in Sect. 1.3.

The present paper proposes and investigates mixtures of multivariate component densities realized via deep neural networks (DNNs) for density estimation. The model is introduced in Sect. 1.4. It extends to multivariate mixture densities our previous work on univariate pdfs, presented as a workshop communication in Ref. [23]. The reader is referred to that paper for a representative list of up-to-date applications where (implicitly or explicitly) density estimation is fundamental and still an open issue. The overall organization of the paper is overviewed in Sect. 1.5.

1.1 Intrinsic Difficulties of Density Estimation via Machine Learning

As pointed out by Vapnik [29], density estimation is an intrinsically difficult problem, and it is still open nowadays. This latter fact is mostly due to the shortcomings of established statistical approaches, either parametric or non-parametric, [27] and by the technical difficulties that arise from attempting to use artificial neural networks (ANNs) or machine learning for pdf estimation. The main shortcomings of the statistical approaches include: the unrealistic assumption of knowing the form of the underlying pdf, necessary in parametric methods; the memory-based nature of non-parametric approaches, which requires to keep all the training examples in memory (resulting in a dramatic burden in terms of space and time) and entails a lack of the generalization capability expected of a proper learning machine. The reader is referred to [27] for a complete list of the major drawbacks of the statistical techniques. On the other hand, the difficulties of using ANNs for pdf estimation stem from: (1) the unsupervised nature of the learning task, (2) the numerical instability problems entailed by pdfs, whose codomains may span the interval $[0, +\infty)$, and (3) the requirement for the resulting model to respect the axioms of probability (in particular the second axiom, that is $\int_{\mathbb{R}^d} p(\mathbf{x})d\mathbf{x} = 1$). Furthermore, the use of maximum-likelihood (ML) training in ANNs tends to result in the "divergence problem", observed first in the realm of hybrid ANN/hidden Markov models [22]. It consists in the progressive divergence of the value of the ANN connection weights as ML training proceeds, resulting in an unbounded growth of the integral of the pseudo-pdf computed by the ANN. The problem does not affect radial basis functions (RBF) networks whose hidden-to-output weights were constrained to be positive and to sum to one, as in the RBF/echo state machine for sequences proposed in Ref. [28], or in the RBF/graph neural network presented in Ref. [4] for the estimation of generalized random graphs. Unfortunately, the use of RBFs in the latter contexts is justified by its allowing for a proper algorithmic hybridization with models devised specifically for sequence/structure processing, but using RBFs as a stand-alone paradigm for density estimation is of neglectable practical interest, since they end up realizing plain Gaussian mixture models (GMM) estimated via ML.

1.2 Earlier Approaches to ANN-Based Density Estimation

In spite of these difficulties, several approaches to pdf estimation via ANNs are found in the literature [26]. A critical survey of the literature is found in Ref. [27]. Most of these approaches suffer from some limitations. First, a ML technique is presented in Ref. [11] where the "integral equals 1" requirement is satisfied numerically by dividing the output of a multilayer Perceptron (MLP) by the numerical integral of the function the MLP computes. No algorithms for computing the numerical integral over high-dimensional spaces are handed out in Ref. [11]. Nonetheless, this approach is related to the technique presented in this paper, insofar that ML will be exploited herein. Differently from [11], a multi-dimensional ad-hoc numeric integration method will be used in the following, jointly with hard constraints, over a mixture of DNNs.

Instead of estimating the pdf directly, techniques have been proposed that focus on the (theoretically equivalent) estimation of the corresponding cumulative distribution function (cdf) [10, 14, 30]. Albeit formally unsupervised (since they learn from unlabeled training samples), such techniques build on training algorithms that include a step of supervised learning, e.g. the backpropagation algorithm (BP) [17]. Application of the latter within the unsupervised framework relies on generating the target outputs (required in supervised training) via the empirical cdf of the data. After training the MLP model $\phi(\cdot)$ of the cdf, the pdf can be retrieved by taking derivatives of $\phi(\cdot)$. Explicit analytical calculations of such derivatives are presented in Ref. [14]. There are some drawbacks to these approaches, as well (see [27]). In particular, a good approximation of the cdf does not entail a good estimate of its derivative. Negative values of $\frac{\partial \phi(x)}{\partial x}$ may even occur, since a linear combination of logistics is not necessarily monotonically increasing, although monotonicity can be enforced by adding penalty terms to the criterion function to be optimized by the training algorithm [10, 14]. Traditional cdf-based approaches like [30] apply naturally to univariate cases, but do not fit the multivariate scenario. An algorithmic extension to multivariate data was presented in a theoretical manner in Ref. [10], but no empirical evidence of its effectiveness was provided. Recently, [14] reported an in-depth experimental investigation of the cdf-based approach to multivariate density estimation.

The idea of exploiting synthetically-generated target outputs was applied to DNN-based pdf estimation in Refs. [27] and [24]. The former relies on a unbiased variant of the Parzen-window (PW) density estimation technique [8] for labeling the training set for a DNN, known as Parzen neural network. Like in the traditional k_n-Nearest Neighbor (k_n-NN) statistical technique [8], the resulting model does not satisfy the axioms of probability. To the contrary, the algorithm for density estimation via feed-forward DNNs presented in Ref. [24] uses a modified loss function, optimized via gradient descent. Such a criterion function combines two terms: the loss between the network output and a synthetically-generated non-parametric estimate of the pdf at hand evaluated over the corresponding input pattern, and a loss between the integral of the function realized by the MLP and the unity value (that is the training "target" for the integral of the DNN). Efficient integration methods based on Markov chain Monte Carlo with importance sampling are used to compute the integral of the DNN and its derivatives w.r.t. the DNN parameters within the gradient-descent scheme used for the optimization. The asymptotic convergence of the resulting technique to the correct solution was formally proven in Ref. [25]. The ideas behind such integration methods are exploited in this paper, as well.

1.3 Mixture Densities

A generalization of plain pdf models stems from the adoption of mixture densities, where the unknown pdf is rather modeled in terms of a combination of any number of component densities [8]. GMMs are the most popular instance of mixture densities [3]. Mixture densities were originally intended as real-life extensions of the single-pdf parametric model (e.g., one Gaussian may not be capable to explain the data distribution but K Gaussian pdfs might as well be). Yet, there is much more than this behind the notion of mixture density. In fact, in mixture densities the different component pdfs are specialized to explain distinct latent phenomena (e.g., stochastic processes) that underlie the overall data generation process, each such phenomenon having different likelihood of occurrence w.r.t. others at diverse regions of the feature space. This suites particularly those situations where the statistical population under analysis is composed of several sub-populations, each having different distribution. Relevant examples are the following:

1. The *statistical study of heterogeneity* in meta-analysis [5], where samples drawn from disjoint populations (e.g., adults and children, male and female subjects, etc.) are collectively collected and have to be analyzed as a whole;
2. The modeling of *unsupervised* or *partially-supervised* [19] data samples in statistical pattern recognition [8], where each sub-population corresponds to a class or category;
3. The *distribution of financial returns* on the stock market depending on latent phenomena such as a political crisis or a war [7];
4. The *assessment of projectile accuracy* in the military science of ballistics when shots at the same target come from multiple locations and/or from different munition types [20].

As it happens, the sub-populations in a mixture are unlikely to be distributed individually according to simple (e.g., Gaussian) pdfs. Consequently, parametric models (like GMMs) are generally not a very good fit. In fact, let $\xi_1, \ldots, \xi_K$ be K disjoint states of nature (the outcomes of a discrete, latent random variable Ξ, each outcome corresponding to a specific sub-population), and let $p(\mathbf{x} \mid \xi_i)$ be the pdf that explains the distribution of the random observations $\mathbf{x}$ given the i-th state of the latent variable, for $i = 1, \ldots, K$. At the whole population level the data will then be distributed according to the mixture $p(\mathbf{x}) = \sum_{i=1}^{K} P(\xi_i)\, p(\mathbf{x} \mid \xi_i)$. Attempts to apply a GMM to model $p(\mathbf{x})$ will not necessarily result in a one-to-one relationship between the Gaussian components in the GMM and the state-specific generative models $p(\mathbf{x} \mid \xi_i)$. In general, at the very least, more than one Gaussian component will be needed to model $p(\mathbf{x} \mid \xi_i)$. Although mixtures of mixture models offer increased modeling capabilities over plain mixture models to this end, they turned out to be unpopular due to the difficulties of estimation of their parameters [2] and their excessive sensitivity to local maxima of their criterion function (generally the likelihood of their parameters given the data).

1.4 The Proposed DNN-Based Model

Aiming at overcoming the aforementioned difficulties encountered with the established approaches, and due to the unexploited potential of using mixture models, the paper proposes a plausible solution in the form of a mixture model built on deep neural networks. The model, called deep neural mixture model (DNMM), relies on a convex combination of component pdfs estimated by component-specific DNNs. The DNMM belongs to the broad family of non-parametric pdf estimation techniques, insofar that (differently from, say, GMMs) no assumptions on the form of the individual component pdfs are made [8]. Due to the learning

and generalization capabilities of DNNs, the DNMM can actually learn a general form for the mixture at hand, overcoming the drawbacks of the traditional non-parametric techniques, as well. A ML training algorithm is devised, satisfying (at least numerically) a combination of hard and soft constraints required in order to guarantee a proper probabilistic interpretation of the estimated model. The resulting machine can also be seen as a novel, special case of mixture of experts [31] having a specific task, a ML-based unsupervised training algorithm, and a particular probabilistic strategy for assigning credit to its individual experts.

This paper is the extended, journal version of a previous workshop communication that we presented in Ref. [23]. Over that communication, the present article extends the experimental investigation to DNNs (although the algorithm in Ref. [23] was suitable to DNNs, no empirical evidence was provided therein), defines formally the modeling capabilities of DNMM, presents the cross-validated likelihood algorithm for DNMM model selection, and reports on the results of experiments conducted over complex, multi-dimensional pdfs (in Ref. [23] only experiments on univariate setups are reported and analyzed).

1.5 Overview of the Paper

The paper is organized as follows. Section 2 is devoted to the ML-based learning algorithm for DNMMs. The family of pdfs that can actually be estimated to any degree of precision via DNMM is defined formally in Sect. 3. An automatic model selection algorithm that exploits the probabilistic nature of the DNMM along with the cross-validated likelihood criterion is presented in Sect. 4. The experimental evaluation is reported in Sect. 5, where the DNMM compares favorably with respect to established statistical techniques (parametric as well as non-parametric) in the task of estimating non-trivial mixtures densities having different number of components and diverse dimensionality of their definition domain. Finally, Sect. 6 draws the conclusions and outlines the current research directions.

2 DNMM: Formal Definition and Training Algorithm

Hereafter we rely on the notation introduced at the beginning of Sect. 1, such that our goal is to estimate $p(\mathbf{x})$ from the information encapsulated within the training sample $\mathcal{T}$. To this aim, we introduce a deep neural mixture model $\tilde{p}(\mathbf{x} \mid W)$ having the following form:

$$\tilde{p}(\mathbf{x} \mid W) = \sum_{i=1}^{K} c_i \, \tilde{p}_i(\mathbf{x} \mid W_i) \tag{1}$$

where W represents the whole set of parameters in the DNMM (that is $c_1, \ldots, c_K, W_1, \ldots, W_K$). The mixing parameters c_i shall satisfy $c_i \in [0, 1]$ for $i = 1, \ldots, K$ and $\sum_{i=1}^{K} c_i = 1$. The i-th component pdf $\tilde{p}_i(\mathbf{x} \mid W_i)$ is defined, for $i = 1, \ldots, K$, as

$$\tilde{p}_i(\mathbf{x} \mid W_i) = \frac{\varphi_i(\mathbf{x}, W_i)}{\int \varphi_i(\mathbf{x}, W_i) d\mathbf{x}} \tag{2}$$

where $\varphi_i(\mathbf{x}, W_i)$ is the function computed by a component-specific DNN whose set of learnable parameters is W_i. We say that this DNN realizes the i-th deep neural component of the

DNMM. Henceforth, plain feed-forward DNNs are assumed (with arbitrary activation functions), but the following calculations can be adapted to a variety of alternate families of DNNs in a (mostly) straightforward manner. The calculations and the resulting algorithm hold good for any number of layers in the DNNs. In order for the DNMM to satisfy Kolmogorov's axioms of probability, a constraint on $\int \varphi_i(\mathbf{x}, W_i)d\mathbf{x}$ shall be imposed shortly. It goes without saying that each DNN in the DNMM has d input neurons (matching the dimensionality of the feature space) and a single output neuron (yielding the value of the estimated pdf), and it is expected to have as many hidden layers as needed (the automatic model selection procedure presented in Sect. 4 is suitable for architecture selection, as well). Without loss of generality for all the present intents and purposes, we assume[1] that the random vectors of interest lie within a compact $S \subset \mathbb{R}^d$, such that S can be regarded as the definition domain of $\varphi_i(\mathbf{x}, W_i)$ for all $i = 1, \ldots, K$. In so doing, numerical integration algorithms are viable to compute $\int \varphi_i(\mathbf{x}, W_i)d\mathbf{x}$, as well as the other integrals required shortly. Equation (2) reduces to $\tilde{p}_i(\mathbf{x} \mid W_i) = \frac{\varphi_i(\mathbf{x}, W_i)}{\int_S \varphi_i(\mathbf{x}, W_i)d\mathbf{x}}$.

The choice of the form of the activation function $f_i(.)$ used in the output layer of the i-th DNN requires some precautions. Due to the very nature of pdfs, $f_i(.)$ is expected to have (at least in principle) a codomain defined as $[0, +\infty)$. This may be granted in several different ways. In this paper we use a logistic sigmoid with component-specific adaptive amplitude $\lambda_i \in \mathbb{R}^+$, that is $f_i(a_i) = \lambda_i/(1 + \exp(-a_i))$ as described in Ref. [21], where a_i represents the current activation value for the output neuron of the i-th DNN in the mixture. Therefore, each neural component in the DNMM can stretch its output over any (component-specific) range $[0, \lambda_i)$, which, instead of being predefined by the user, is learned from the data (just like any other parameter in W_i) in compliance with the nature of the corresponding component pdf.[2]

The DNMM training algorithm revolves around a learning rule for the mixture parameters W given $\mathcal{T}$, such that at the end of the training process the quantity $\tilde{p}(\mathbf{x} \mid W)$ results in a robust estimate of $p(\mathbf{x})$. This is achieved by pursuing two purposes: (1) exploiting the information encapsulated in $\mathcal{T}$ to approximate the unknown pdf; (2) preventing the DNNs in the DNMM from developing spurious solutions, by enforcing the constraints $\int_S \varphi_i(\mathbf{x}, W_i)d\mathbf{x} = 1$ for all $i = 1, \ldots, K$. To this aim, a constrained algorithm is devised that builds on the stochastic gradient-ascent maximization of the point-wise likelihood $\tilde{p}(\mathbf{x}_j \mid W)$ of the DNMM given the current training pattern $\mathbf{x}_j$. The stochastic optimization step has to be applied iteratively for $j = 1, \ldots, n$. This is achieved by means of an on-line, differentiable loss function $C(.)$ defined as

$$C(W, \mathbf{x}_j) = \tilde{p}(\mathbf{x}_j \mid W) - \rho \sum_{i=1}^{K} \frac{1}{2} \left(1 - \int_S \varphi_i(\mathbf{x}, W_i)d\mathbf{x}\right)^2 \tag{3}$$

to be maximized with respect to the DNMM parameters W under the (hard) constraints that $c_i \in [0, 1]$ for $i = 1, \ldots, K$ and $\sum_{i=1}^{K} c_i = 1$. The second term in the loss function is rather a "soft" constraint that enforces a unit integral of $\tilde{p}_i(\mathbf{x}, W_i)$ over S for all $i = 1, \ldots, K$, as sought, such that $\int_S \tilde{p}(\mathbf{x} \mid W)d\mathbf{x} \simeq 1$. The hyper-parameter $\rho \in \mathbb{R}^+$ balances the importance of the constraints, and may be used in real-life applications in order to tackle potential numerical issues. The learning rule Δw for a generic parameter w in the DNMM is written

[1] It is seen that, in practical applications, any data normalization approach may be applied to the training sample in order to ensure respect of this assumption.

[2] Other advantages entailed by the use of adaptive amplitudes are pointed out in Ref. [21].

as $\Delta w = \eta \frac{\partial C(.)}{\partial w}$, where $\eta \in \mathbb{R}^+$ is the learning rate. Different calculations are needed, depending on w being either (i) a mixing coefficient, i.e. $w = c_k$, or (ii) a parameter[3] within any of the DNN-based component densities. In case (i), we introduce K unconstrained latent variables $\gamma_1, \ldots, \gamma_K$, and we let

$$c_k = \frac{\varsigma(\gamma_k)}{\sum_{i=1}^{K} \varsigma(\gamma_i)} \tag{4}$$

for $k = 1, \ldots, K$, where $\varsigma(x) = 1/(1 + e^{-x})$. In so doing, hereafter each γ_k can be treated as the unknown parameter to be estimated, in place of the corresponding c_k. Consequently, higher-likelihood mixing parameters c_k that satisfy the required constraints are implicitly yielded by the application of the learning rule. The latter can be written as:

$$\begin{aligned}
\Delta\gamma_k &= \eta \frac{\partial C(.)}{\partial \gamma_k} \\
&= \eta \frac{\partial \tilde{p}(\mathbf{x}_j \mid W)}{\partial \gamma_k} \\
&= \eta \frac{\partial \sum_{i=1}^{K} c_i \tilde{p}_i(\mathbf{x}_j \mid W_i)}{\partial \gamma_k} \\
&= \eta \sum_{i=1}^{K} \tilde{p}_i(\mathbf{x}_j \mid W_i) \frac{\partial}{\partial \gamma_k} \left(\frac{\varsigma(\gamma_i)}{\sum_{\ell=1}^{K} \varsigma(\gamma_\ell)} \right) \\
&= \eta \left\{ \tilde{p}_k(\mathbf{x}_j \mid W_k) \frac{\varsigma'(\gamma_k)}{\sum_{\ell=1}^{K} \varsigma(\gamma_\ell)} - \sum_{i=1}^{K} \tilde{p}_i(\mathbf{x}_j \mid W_i) \frac{\varsigma(\gamma_i)\varsigma'(\gamma_k)}{[\sum_{\ell} \varsigma(\gamma_\ell)]^2} \right\} \\
&= \eta \frac{\varsigma'(\gamma_k)}{\sum_{\ell} \varsigma(\gamma_\ell)} \left\{ \tilde{p}_k(\mathbf{x}_j \mid W_k) - \tilde{p}(\mathbf{x}_j \mid W) \right\}
\end{aligned} \tag{5}$$

where, in order to obtain the fifth step of the calculations from the fourth, we applied the quotient rule (observing that $\frac{\partial \varsigma(\gamma_i)}{\partial \gamma_k} = 0$ if $i \neq k$), and the last step is obtained by exploiting Eq. (4) as well as the very definition of DNMM, that is Eq. (1).

Next, let us focus on scenario (ii), i.e. let w be a parameter within one of the DNNs. Taking the partial derivative of $C(W, \mathbf{x}_j)$ with respect to w requires calculating the derivatives of the first and the second terms in the right-hand side of Eq. (3), respectively. For notational convenience, hereafter we assume that w belongs to the generic k-th DNN. The partial derivative of the first term can be written as

$$\begin{aligned}
\frac{\partial \tilde{p}(\mathbf{x_j} \mid W)}{\partial w} &= \frac{\partial}{\partial w} \sum_{i=1}^{K} c_i \tilde{p}_i(\mathbf{x_j} \mid W_i) \\
&= \frac{\partial}{\partial w} \{c_k \tilde{p}_k(\mathbf{x_j} \mid W_k)\} \\
&= c_k \frac{\partial}{\partial w} \left\{ \frac{\varphi_k(\mathbf{x}_j, W_k)}{\int_S \varphi_k(\mathbf{x}, W_k) d\mathbf{x}} \right\}
\end{aligned}$$

[3] A connection weight, bias, adaptive amplitude, or any other trainable parameter.

$$= c_k \left\{ \frac{1}{\int_S \varphi_k(\mathbf{x}, W_k) d\mathbf{x}} \frac{\partial \varphi_k(\mathbf{x}_j, W_k)}{\partial w} - \frac{\tilde{p}_k(\mathbf{x}_j, W_k)}{\int_S \varphi_k(\mathbf{x}, W_k) d\mathbf{x}} \frac{\partial}{\partial w} \int_S \varphi_k(\mathbf{x}, W_k) d\mathbf{x} \right\}$$

$$= \frac{c_k}{\int_S \varphi_k(\mathbf{x}, W_k) d\mathbf{x}} \left\{ \frac{\partial \varphi_k(\mathbf{x}_j, W_k)}{\partial w} - \frac{\varphi_k(\mathbf{x}_j, W_k)}{\int_S \varphi_k(\mathbf{x}, W_k) d\mathbf{x}} \int_S \frac{\partial \varphi_k(\mathbf{x}, W_k)}{\partial w} d\mathbf{x} \right\} \quad (6)$$

where Eq. (2) and the quotient rule were exploited for obtaining the fourth step of the calculations from the third, while Leibniz rule was applied in the last step of the calculations. Note that these calculations are formally identical to those we presented in Ref. [23]. In fact, the depth of the k-th DNN and, in turn, the specific layer within the corresponding architecture where the generic weight w is located, are implicitly accounted for by the regular BP-based computation of $\frac{\partial \varphi_k(\mathbf{x}, W_k)}{\partial w}$. In passing, it is worth noting that Eq. (6) is the mathematical statement of the very rationale behind the specific impact that the current training pattern $\mathbf{x}_j$ has on the learning process for distinct neural components of the DNMM. In fact, the amount of parameter change Δw is proportional to the probabilistic "credit" c_k of the neural component at hand. Moreover, the quantities within brackets in Eq. (6) depend on the value of the k-th DNN output over $\mathbf{x}_j$, as well as on its derivative. If, at any given time during the training process, $\varphi_k(.)$ does not change significantly in a neighborhood of $\mathbf{x}_j$ (e.g. if $\mathbf{x}_j$ lies in a high-likelihood plateau or, vice versa, in a close-to-zero plateau of $\varphi_k(.)$) then the contribution of the first quantity within brackets is neglectable. Moreover, if $\varphi_k(\mathbf{x}_j) \simeq 0$ then the second term within brackets turns out to be neglectable, as well. To the contrary, the contribution of $\mathbf{x}_j$ to the parameter adaptation of k-th component DNN turns out to be paramount if $\varphi_k(.)$ returns a high likelihood over $\mathbf{x}_j$ and significant variations in its surroundings. Roughly speaking, this explains how the estimate of each separate DNN $\varphi_k(.)$ works.

Next, Leibniz rule is used again in the calculation of the derivative of the second term in the right-hand side of Eq. (3), which can be written as

$$\frac{\partial}{\partial w} \left\{ \rho \sum_{i=1}^{K} \frac{1}{2} \left(1 - \int_S \varphi_i(\mathbf{x}, W_i) d\mathbf{x} \right)^2 \right\}$$

$$= \frac{\partial}{\partial w} \left\{ \frac{\rho}{2} \left(1 - \int_S \varphi_k(\mathbf{x}, W_k) d\mathbf{x} \right)^2 \right\}$$

$$= -\rho \left(1 - \int_S \varphi_k(\mathbf{x}, W_k) d\mathbf{x} \right) \frac{\partial}{\partial w} \int_S \varphi_k(\mathbf{x}, W_k) d\mathbf{x}$$

$$= -\rho \left(1 - \int_S \varphi_k(\mathbf{x}, W_k) d\mathbf{x} \right) \int_S \frac{\partial \varphi_k(\mathbf{x}, W_k)}{\partial w} d\mathbf{x}. \quad (7)$$

In order to compute the right-hand side of Eqs. (6) and (7) we need suitable algorithms for the computation of $\frac{\partial \varphi_k(\mathbf{x}_j, W_k)}{\partial w}$, $\int_S \varphi_k(\mathbf{x}, W_k) d\mathbf{x}$, and $\int_S \frac{\partial}{\partial w} \varphi_k(\mathbf{x}, W_k) d\mathbf{x}$. As regards the computation of $\frac{\partial \varphi_k(\mathbf{x}_j, W_k)}{\partial w}$ we proceed as in traditional BP, or as in Ref. [21] in case $w = \lambda_k$. As for the integrals, deterministic numerical quadrature integration techniques (e.g., Simpson's method, trapezoidal rule, etc.) are viable only if $d = 1$. In fact, in terms of computational burden, they cannot realistically scale up to higher dimensions ($d \geq 2$). This is all the more critical in the light of the fact that $\int_S \frac{\partial}{\partial w} \varphi_k(\mathbf{x}, W_k) d\mathbf{x}$ shall be iteratively computed for each parameter of each DNN in the DNMM. Moreover, deterministic integration methods do not exploit at all the nature of the function under integration. Roughly speaking, herein the integrand is expected to be an "approximation" of the pdf (say, $p_k(\mathbf{x})$) that explains the distribution of that specific sub-sample of $\mathcal{T}$ that is drawn from the k-th component of the mixture.

To the contrary, accounting for the pdf of the data should drive the integration algorithm towards integration points that cover "interesting" regions[4] of the domain of the integrand. For these reasons, we apply a component-oriented version of the approach we presented in Ref. [24]. The resulting approach can be seen as an instance of Markov chain Monte Carlo [1]. It is a non-deterministic, multi-dimensional integration technique which accounts for the component-specific probability distribution of the data. Let $\phi_k(\mathbf{x})$ denote the integrand at hand (i.e., $\varphi_k(\mathbf{x}, W_k)$ or $\frac{\partial \varphi_k(\mathbf{x}, W_k)}{\partial w}$). Monte Carlo with importance sampling [16] yields an approximation of the integral of $\phi_k(\mathbf{x})$ over S in the form $\int_S \phi_k(\mathbf{x}) d\mathbf{x} \simeq \frac{V(S)}{m} \sum_{\ell=1}^{m} \phi_k(\dot{\mathbf{x}}_\ell)$ where m properly sampled integration points $\dot{\mathbf{x}}_1, \ldots, \dot{\mathbf{x}}_m$ are used. Sampling of the ℓ-th integration point $\dot{\mathbf{x}}_\ell$ (for $\ell = 1, \ldots, m$) is obtained by drawing it at random from the mixture pdf $p_u^{(k)}(\mathbf{x})$ defined as

$$p_u^{(k)}(\mathbf{x}) = \alpha(t) u(\mathbf{x}) + (1 - \alpha(t)) \tilde{p}_k(\mathbf{x} \mid W_k) \tag{8}$$

where $u(\mathbf{x})$ is the uniform distribution over S, and $\alpha : \mathbb{N} \rightarrow (0, 1)$ is a decaying function of the number t of the DNMM training epochs[5] for $t = 1, \ldots, T$, such that $\alpha(1) \simeq 1.0$ and $\alpha(T) \simeq 0.0$. As in [24] we let $\alpha(t) = 1/(1 + e^{\frac{t/T - 1/2}{\theta}})$, where θ is a hyperparameter. Equation (8) is such that the importance sampling mechanism it implies accounts for the (estimated) component density $\tilde{p}_k(\mathbf{x} \mid W_k)$ of the k-th latent subsample of the data, therefore respecting the natural distribution of such sub-population and focusing integration on the relevant integration points (i.e., the points having high component-specific likelihood). At the same time, since the estimates of this component pdfs are unfit during the early stage of the DNMM training, Eq. (8) prescribes a (safer) sampling from a uniform distribution at the beginning (practically behaving like a plain Monte Carlo algorithm). As long as the robustness of the DNMM estimate increases, i.e. as t increases, sampling from $\tilde{p}_k(\mathbf{x} \mid W_k)$ replaces progressively the sampling from $u(\mathbf{x})$, ending up in purely non-uniform importance sampling. The form of $\alpha(t)$ is defined accordingly. Since $\varphi_k(\mathbf{x}, W_k)$ is intrinsically non-negative, for $t \rightarrow T$ the sampling occurs substantially from $|\varphi_k(\mathbf{x}, W_k)| \,/\, \int_S |\varphi_k(\mathbf{x}, W_k)| \, d\mathbf{x}$, that is a sufficient condition for granting that the variance of the estimated integral vanishes and the corresponding error goes to zero [13].

Sampling from $p_u^{(k)}(\mathbf{x})$ requires a viable technique for sampling from the k-th DNN in the DNMM. A variant of Markov chain Monte Carlo, namely the Metropolis–Hastings (M-H) algorithm [12], is exploited in this paper. M-H is robust to the fact that, during training, $\varphi_k(\mathbf{x}, W_k)$ may not be properly normalized but it is proportional by construction to the corresponding pdf estimate (which is normalized properly, instead, by definition) [12]. Due to its efficiency and ease of sampling, a multivariate logistic pdf with radial basis, having location $\mathbf{x}$ and scale σ, is used as the *proposal* pdf $q(\mathbf{x}' \mid \mathbf{x})$ required by M-H to generate a new candidate sample $\mathbf{x}' = (x'_1, \ldots, x'_d)$ from the current sample $\mathbf{x} = (x_1, \ldots, x_d)$. Formally, such a proposal pdf is defined as $q(\mathbf{x}' \mid \mathbf{x}, \sigma) = \prod_{i=1}^{d} \frac{1}{\sigma} e^{(x'_i - x_i)/\sigma} (1 + e^{(x'_i - x_i)/\sigma})^{-2}$ which can be easily sampled via the inverse transform sampling technique. The hyperparameters needed (i.e., the scale σ of the proposal pdf and the burn-in period for M-H) are fixed empirically as part of the overall model selection process (see Sect. 4). Note that the equations presented so far are a compact and univocal representation of the DNMM estimation algorithm, which is readily implementable in software provided that any regular BP software is used to compute

[4] That is, regions having high component-specific likelihood.

[5] In the present context, a training epoch is a completed re-iteration of Eqs. (6) and (7) for all the parameters of the DNMM over all the observations in $\mathcal{T}$.

$\frac{\partial \varphi_k(\mathbf{x}, W_k)}{\partial w}$ for all possible values of k and w. Like any DNN training algorithm based on the gradient method, the present technique is guaranteed to increase locally the likelihood of the parameters W but it may end up being trapped into local maxima of the criterion function, depending on the initial value of W. This may affect significantly the quality of the resulting pdf estimate. An empirical workaround for this issue consists in initializing the DNMM with different random values for W and to select the pdf model that results in the highest likelihood after training. The seed of any pseudo-randomization algorithm used to this aim may be fixed (like any other hyperparameter) via the automatic likelihood-driven model selection strategy presented in Sect. 4.

3 Class of Pdfs that can be Modeled Accurately via DNMMs

It is seen that not all pdfs can be estimated in an accurate manner via DNMMs. A couple of simple, univariate examples should be more than enough to convince the reader of this: the Dirac's Delta (which is not continuous and not bounded) and the standard exponential pdf (defined as $p_e(x) = 0$ if $x < 0$, $p_e(x) = \exp(-x)$ otherwise). In Ref. [27] we introduced the class of *nonpaltry pdfs*, that basically embrace all the "interesting" pdfs that are of practical interest (and, that can be estimated to any degree of precision by means of Parzen Neural Networks). The arguments handed out by Ref. [27] can be extended to DNMMs, as well. The formal definition of this class of function goes as follows. Let S be a compact subset of I_d, where $I_d = [0, 1]^d$ (the symbol S has the same practical meaning it had in the previous section). A continuous pdf $p : \mathbb{R}^d \rightarrow \mathbb{R}$ is said to be *nonpaltry* over S if $\int_S p(\mathbf{x})d\mathbf{x} = 1$. Accordingly, we formally define the class $\mathcal{P}(S)$ of nonpaltry pdfs over S as the set of all the density functions that are nonpaltry over S. In passing, note that mixture densities built on nonpaltry component densities are, in turn, nonpaltry. Theorem 3.1 in Ref. [27] states that for any nonpaltry pdf $p(\cdot)$ over S at least one DNN exists that computes $\phi^{(\epsilon)}(\cdot)$ s.t. $\|\phi^{(\epsilon)}(\cdot) - p(\cdot)\|_{\infty,S} < \epsilon$ for any $\epsilon \in \mathbb{R}^+$. This (trivially) implies that at least one DNMM (with $c = 1$) exists that computes $\phi^{(\epsilon)}_{DNMM}(\cdot)$ s.t. $\|\phi^{(\epsilon)}_{DNMM}(\cdot) - p(\cdot)\|_{\infty,S} < \epsilon$ for any $\epsilon \in \mathbb{R}^+$. Roughly speaking, we can estimate any nonpaltry pdf to any degree of precision by means of a DNMM.

4 Automatic Likelihood-Driven Model Selection Strategy

The former section presented a modeling (or, approximation) capability of the family of pdfs realized by DNMMs, regardless of the actual convergence (during training) of a specific DNMM to the unknown pdf $p(\cdot)$ underlying the data $\mathcal{T}$ at hand. A formal treatment of the asymptotic convergence of DNMMs is beyond the scope of the paper. In practice, roughly speaking, as the cardinality of the training sample $\mathcal{T}$ increases (ideally, for $n \rightarrow +\infty$) the resulting estimate realized by the DNMM tends (in probability) to approach $p(\cdot)$. This means that, for sufficiently large values of n, the likelihood of the DNMM given a separate validation set $\mathcal{V}$ (independently drawn from $p(\cdot)$, as well) tends to "increase[6]" with n. This informal reasoning suggests the adoption of the maximum-likelihood criterion as an evaluation function that can be used throughout the DNMM model selection process. This is made viable by the

[6] The term "increase" is used hereby with its qualitative meaning, that is to indicate a trend and not a strictly monotonic behavior: in fact, the likelihood of the DNMM does not necessarily increase monotonically as the training process proceeds.

very probabilistic nature of the DNMM, and it is not available to generic, non-probabilistic DNNs or mixtures of experts.

In short, an iterative model/hyperparameter search strategy may be developed as follows. To fix ideas, let us first consider the issue of fixing the number of neurons in a certain hidden layer of a DNN within the DNMM.[7] We start with an initial number of hidden neurons, and we increase it by u units (for a certain integer u, e.g. $u = 1$) at each of the following steps of the selection procedure. At each iteration (starting from the initialization) the DNMM is trained on $\mathcal{T}$, and the likelihood of the resulting model given $\mathcal{V}$ is evaluated. The procedure is iterated for as long as this likelihood increases. We stop searching when, for τ consecutive iterations, the relative gain in likelihood between consecutive steps is lower than a fixed percentage $\nu\%$. The minimum description length principle is then applied [15], selecting the simplest model amongst those (explored throughout the last τ iterations) which yielded comparable values of the corresponding validation likelihood given $\mathcal{V}$ (where "comparable" means that their difference is below a user-defined threshold). It is worth noting that this approach relies on an instance of the so-called cross-validated likelihood criterion [18]. The same incremental strategy is viable for selecting the number of layers within the DNNs in the mixture (e.g., starting from a single hidden layer and deepening the architecture by adding one more layer at each step). In fact, such a strategy is used for selecting "optimal" (i.e., maximum-likelihood) depths of the DNNs involved in the mixtures used in the experiments presented in Sect. 5.2.

The present cross-validated likelihood search strategy can be applied, in a straightforward manner, to the problem of selecting the other hyperparameters controlling the behavior of the DNMM learning process. In order to limit the computational burden of such an automatic model selection process, random search of the values of the hyperparameters (each choice characterized by the corresponding likelihood given $\mathcal{V}$) is recommended instead of going through the step-by-step incremental approach we proposed for selecting the umber of neurons.

5 Experiments

Experiments are conducted on random samples drawn from multimodal mixtures having known form. Section 5.1 reports on the results obtained on univariate data using mixtures of 3-layer DNNs, while Sect. 5.2 presents the results of experiments involving multivariate data (of different dimensionalities) using deeper DNMMs.

5.1 Experiments with Univariate Data

In the present, univariate case, the random samples were drawn from mixtures $p(x)$ of c Fisher–Tippett pdfs, i.e. $p(x) = \sum_{i=1}^{c} \frac{P_i}{\beta_i} \exp\left(-\frac{x-\mu_i}{\beta_i}\right) \exp\left\{-\exp\left(-\frac{x-\mu_i}{\beta_i}\right)\right\}$. The mixing coefficients $P_1, \ldots, P_c$ were drawn at random from the uniform distribution over $[0, 1]$ and normalized in such a way that $\sum_{i=1}^{c} P_i = 1$. The component densities of the Fisher-Tippett mixture are identified by their scales β_i and locations μ_i, for $i = 1, \ldots, c$. The scales were drawn at random from the uniform distribution over $(0.01, 0.9)$, and the locations were randomly and uniformly distributed over $(0, 10)$. Estimation tasks were created using 1200

[7] Note that there is no assumption in the definition of DNMM that forces the practitioner to use the same number of layers and/or neurons in the different DNNs realizing the components of the mixture, although this may be a computationally realistic choice.

Table 1 Estimation of the Fisher–Tippett mixture $p(x)$ (with $n = 800$) in terms of integrated squared error as a function of the number c of the Fisher-Tippett component densities. Best results are shown in boldface. (*Legend*: k-GMM and k-DNMM denote the GMM and the DNMM with k components, respectively)

c	5	10	15	20	Avg. $\pm$ std. dev.
8-GMM	9.60e − 3	1.12e − 2	4.57e − 2	7.99e − 2	(3.66 ± 2.89)e − 2
16-GMM	6.33e − 3	9.29e − 3	3.78e − 2	4.24e − 2	(2.40 ± 1.63)e − 2
32-GMM	7.15e − 3	9.82e − 3	2.41e − 2	3.03e − 2	(1.78 ± 0.97)e − 2
k_n-NN	6.54e − 3	8.70e − 3	2.03e − 2	2.36e − 2	(1.48 ± 0.73)e − 2
PW	6.02e − 3	8.94e − 3	2.14e − 2	1.98e − 2	(1.40 ± 0.67)e − 2
4-DNMM	6.41e − 3	7.06e − 3	1.09e − 2	1.40e − 2	(9.59 ± 3.07)e − 3
8-DNMM	**5.89**e − **3**	**6.02**e − **3**	8.11e − 3	1.01e − 2	(**7.53** ± 1.73)e − 3
12-DNMM	6.38e − 3	6.27e − 3	**8.05**e − **3**	**9.64**e − **3**	(7.59 ± **1.38**)e − 3

random patterns each, drawn from $p(x)$, and a variable number c of component densities (namely $c = 5$, 10, 15 and 20). Each c-specific sample was split into a training set (with $n = 800$ patterns) and a validation set (the remaining 400 patterns). The integrated squared error (ISE) between $p(x)$ and its estimate $\tilde{p}(x)$, i.e. $\int (p(x) - \tilde{p}(x))^2 \, dx$, is adopted in order to assess the performance of the different estimators. Simpson's method was applied to compute numerically the ISE.

In the present, illustrative setup we used DNMMs involving DNNs with a 3-layer architecture (input layer, hidden layer of 9 neurons, output layer). Sigmoid activation functions were used in the hidden and the output layers, having layer-wise [21] adaptive λ. All the DNMM parameters underwent pseudo-random initialization over zero-centered intervals, except for the values of λ (that were initialized to 1) and the mixing parameters that were initialized as $c_i = 1/K$ for $i = 1, \ldots, K$. As in Ref. [24] we used a function $\alpha(t)$ having $\theta = 0.07$, and we relied on $m = 400$ integration point. The latter ones were sampled at the beginning of each training epoch using a scale $\sigma = 9$ for the logistic proposal pdf in M-H. The burn-in period of the Markov chain in M-H was stretched over the first 500 states. The other hyper-parameters of the DNMM (including the seeds for the pseudo-random initialization of the DNMM parameters) were fixed via random-search based on the cross-validated likelihood criterion presented in Sect. 4. No normalization was applied to the values of the input data. Table 1reports the results. DNMMs having different values of K (that is $K = 4$, 8 and 12) were tested and compared with 8-GMM, 16-GMM, and 32-GMM (initialized via k-means and refined via iterative ML [8]), k_n-NN with unbiased $k_n = 1\sqrt{n}$ [8], and Parzen Window (PW) with standard width $h_n = 1/\sqrt{n}$ of its Gaussian kernels [8].

The Table shows that, regardless of the model used, the corresponding ISE increase as a function of c, as expected. The DNMMs improve systematically (and, significantly) over the statistical techniques, whatever the value of c. On average, both the 8-DNMM and the 12-DNMM yield a 46% relative ISE reduction over the PW (the latter turns out to be the most robust non-neural estimation algorithm). Welch's t-test (used in order to account for the different variances of the models) [9] returns a level of confidence $> 90\%$ on the statistical significance of the gap between the 8- (or, 12-) DNMM and the PW. Furthermore, the DNMMs turn out to be the stablest models overall, as proved by the values of the corresponding standard deviations (last column of the table). This is evidence of the fact that the estimation accuracy offered by the DNMMs is less sensitive to the complexity of the underlying Fisher-Tippett mixture (i.e., to c) than the accuracies yielded by the traditional statistical techniques. Finally,

differences in terms of ISE are observed among the DNMMs depending on K. Nonetheless, differences between the 8-DNMM and the 12-DNMM turn out to be mild, and they depend on the complexity of the underlying pdf to be estimated (at least to some extent), as expected.

5.2 Experiments with Multivariate Data

Hereafter we use multivariate data drawn from mixtures of generalized extreme value distributions (GEV) [6] with null slope and having the following parametric form:

$$m\text{-}GEV(\mathbf{x};\boldsymbol{\Theta}) = \sum_{k=1}^{c_T} \frac{1}{c_T} \prod_{i=1}^{d} \frac{1}{\beta_{ki}} \exp\left(-\frac{x_i-\mu_{ki}}{\beta_{ki}}\right) \exp\left\{-\exp\left(-\frac{x_i-\mu_{ki}}{\beta_{ki}}\right)\right\}$$

where the GEV parameters $\boldsymbol{\Theta} = (d, c_T, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{c_T}, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{c_T})$ are defined as follows:

1. d is the dimensionality of the feature space as usual. Therefore, the generic real valued random vector $\mathbf{x} \in \mathbb{R}^d$ can be written as $\mathbf{x} = (x_1, \ldots, x_d)$;
2. c_T is the number of component GEVs in the mixture, and $c_T = c^d$ where c denotes the (fixed) number of diverse modes of $m\text{-}GEV(\mathbf{x};\boldsymbol{\Theta})$ along each one of the dimensions in the definition domain of the pdf;
3. $\boldsymbol{\mu}_k = (\mu_{k1}, \ldots, \mu_{kd})$ and $\boldsymbol{\beta}_k = (\beta_{k1}, \ldots, \beta_{kd})$ are the location and the scale vectors of k-th component GEV, respectively. They are defined in such a manner that the set $\{(\boldsymbol{\mu}_k, \boldsymbol{\beta}_k) \mid k = 1, \ldots, c^d\}$ embraces all possible combinations of c dimension-specific parameters $(\bar{\mu}_{i1}, \bar{\beta}_{i1}), \ldots, (\bar{\mu}_{ic}, \bar{\beta}_{ic})$ for $i = 1, \ldots, d$ (for a total of $c_T = c^d$ combinations).

For each $i = 1, \ldots, d$ and $j = 1, \ldots, c$, the values $\bar{\mu}_{ij}$ are random quantities, independently and uniformly drawn from the interval $(0.1, 0.9)$, and the iid random values $\bar{\beta}_{ij}$ are drawn from the interval $(0.03, 0.05)$. In the following, numerical integration is computed over the interval $S = [0, 1.1]^d$.

We generated different estimation tasks involving *m-GEVs* with an increasing number of components ($C_T = 4, 9, 16$ and 25, respectively) and an increasing dimensionality of the feature space ($d = 2, 4, 6$ and 8, respectively). For each combination of such values for C_T and d, a sample of 1200 patterns was randomly drawn from $m\text{-}GEV(\mathbf{x};\boldsymbol{\Theta})$. Each such data sample was split into a training set (800 random vectors) and a validation set (the remaining 400 patterns), as we did before. The validation set was used to select the architectures and hyperparameters for the algorithms via ML-based random search, on a d- and c_T-specific basis, using the model selection procedure presented in Sect. 4. This cross-validated likelihood criterion was exploited also in order to fix a statistical baseline for assessing the performance of the DNMM, as follows. First, three established and popular statistical estimation techniques suitable for multivariate, multimodal pdf estimation were evaluated, namely the PW, the k_n-NN, and the GMM (the latter was evaluated several times, for an increasing number of component Gaussians ranging from 4 to 32). In PW we used Gaussian kernels with bandwidth $h_n = h_1/\sqrt{n}$, where h_1 was selected over the $[10^{-1}, 10^1]$ range. For the k_n-NN we let $k_n = k_1\sqrt{n}$, as usual, with k_1 selected via cross-validated ML. As for the GMMs, the k-means algorithm was used for the initialization of the parameters, which were then refined via iterative ML re-estimation [8]. After ML selection of suitable hyperparameters for PW, k_n-NN, and GMM, the best amongst these statistical models (in terms of likelihood on the validation set) was adopted as the baseline for the specific combination of d and c_T at hand. The whole procedure was then repeated for the other combinations of d and c_T, fixing the corresponding baselines. Each such baseline was quantified as the ISE with respect to

Table 2 Estimation of the multivariate mixture of generalized extreme value distributions *m-GEV*($\mathbf{x}$; $\boldsymbol{\Theta}$) (with $n = 800$) in terms of relative (%) ISE reduction over the baseline, as a function of the number C_T of the component densities of the *m-GEV*($\mathbf{x}$; $\boldsymbol{\Theta}$) and of the dimensionality d of the feature space

	$C_T = 4$	$C_T = 9$	$C_T = 16$	$C_T = 25$	*Avg.*
$d = 2$	11.31	10.52	7.38	9.02	9.56 ± 1.50
$d = 4$	8.44	-0.07	5.75	7.01	5.28 ± 3.23
$d = 6$	4.98	-1.64	5.80	8.13	4.32 ± 3.63
$d = 8$	6.20	7.34	8.63	8.00	7.54 ± 0.90
Avg.	7.73 ± 2.41	4.04 ± 5.05	6.89 ± 1.20	8.04 ± 0.71	6.67 ± 3.29

the true *m-GEV* underlying the data sample for those specific values of d and c_T. Next, a DNMM was selected, again via the procedure presented in Sect. 4. Selection involved fixing the architecture of the DNNs (number of layers between 3 and 6, number of neurons per layer, type of activation functions) and the hyperparameters used for training the DNMM. The same architectures and hyperparameters were applied for all the DNNs in the DNMM at any step of the random search process. DNMMs were initialized as in Sect. 5.1, using the same parameters selected therein for executing the numeric integration, including the burn-in period, the form of $\alpha(t)$, the value of m, etc. Eventually, the ISE between the DNMM selected this way (after training) and the true *m-GEV* was computed and compared with the baseline. In so doing, the results (reported in Table 2) can be expressed in terms of the percentage of relative ISE reduction (%) yielded by the best DNMM over the baseline. To fix ideas, a 0% relative ISE reduction means that the DNMM equals the baseline, a negative relative ISE reduction means that the DNMM performance worsened the baseline, while positive reductions represent gains over the baseline (the larger, the better: ideally, in the limit case, a 100% relative ISE reduction would be observed if the DNMM could result in a perfect model of the true *m-GEV* in presence of an imperfect estimation baseline).. Expressing the results of the experiments in this way has two significant advantages: (1) keeping the amount of values to be read and compared, and the size of the table, to a human-readable scale, and (2) conveying the sense of the gap between the DNMM and any other established statistical estimation technique.

It is seen that, in a quasi-systematic manner, the DNMM improves significantly over the statistical approaches. Only in two cases out of 16 the ISE yielded by the DNMM is just in line with (or, slightly worse than) the baseline. Overall, the DNMM offers an average 6.67 ± 3.29 (%) relative ISE reduction over its closest competitor. The statistical significance of the improvement offered by the DNMM over the baseline (and, in turn, over any statistical technique) computed via Welch's t-test is high, namely $\geq 90\%$. The values of the standard deviations, reported in the last column and last row of the table, show that the DNMM results also in a substantially stable behavior overall.

Note that the gap between the DNMM and the best statistical estimates is not as wide as in the univariate setup. The rationale behind the reduced gap is likely to be found in the model selection procedure we used: the random-search strategy applied within the cross-validated ML model selection algorithm over the multivariate data (much needed, in place of a more accurate grid-search, in order to keep the overall computational burden to a realistic scale) proved capable of finding suitable yet sub-optimal architectures and hyperparameters. This affects significantly the outcome of the model selection process in the DNMM, due to the many hyperparameters to be fixed (each choice resulting in a profoundly different model, viz.

performance). The issue is much less severe in the case of the statistical estimators, which involve selecting just a few hyperparameters (even one only in the case of PW or k_n-NN). Another reason for the reduced gap between the DNMM and the baseline lies in our inheriting from the univariate setup the hyperparameters for M-H: it is seen that the effectiveness of (say) a certain number m of integration points (which is suitable to univariate data) tend to reduce as the dimensionality of the feature space increases. Nevertheless, also in the present setup the results are evidence of the improved estimation and modeling capabilities of the DNMM even in complex multivariate density estimation tasks.

6 Conclusions

The paper presented the DNMM as a sound DNN-based model for multivariate pdf estimation that satisfies Kolmogorov's axioms. The DNMM overcomes all the usual shortcomings of traditional statistical techniques. In particular, its nonparametric modeling capabilities allow for realizing flexible component densities that are not constrained to have a predefined form, as it happens in parametric statistics. Moreover, the DNMM does not suffer from the limitations of nonparametric statistical estimators, as well, insofar that it is a learning machine (instead of a memory-based algorithm), which entails generalization, smooth and compact solutions, as well as reduced time- and space-complexity. Furthermore, the probabilistic nature of the DNMM allows for the exploitation of the cross-validated likelihood criterion in order to carry out the model selection relying on the ML of the hyperparameters given the validation data.

The experimental results show significant improvements yielded by the DNMM over the baseline of the statistical techniques on both the univariate and the multivariate data drawn from several complex pdfs. Future work will revolve around the following points: (1) the initialization procedure. Non-uniform initialization of the mixing coefficients may turn out to be helpful in breaking potential ties, and initializing the individual DNN-based components via supervised learning of a subset of the components of a pre-estimated reference mixture model (i.e., a GMM) may improve the random initialization of the DNNs parameters; (2) an in-depth analysis of sensitivity of the DNMM to the different hyperparameters and architectural aspects that may affect the outcome of the estimation algorithm.

Funding Open access funding provided by Universit[Pleaseinsertintopreamble] degli Studi di Siena within the CRUI-CARE Agreement.

References

1. Andrieu C, de Freitas N, Doucet A, Jordan MI (2003) An introduction to MCMC for machine learning. Mach Learn 50(1–2):5–43
2. Aste M, Boninsegna M, Freno A, Trentin E (2015) Techniques for dealing with incomplete data: a tutorial and survey. Pattern Anal Appl 18(1):1–29

3. Bishop CM (2006) Pattern recognition and machine learning (information science and statistics). Springer, Berlin, Heidelberg
4. Bongini M, Rigutini L, Trentin E (2018) Recursive neural networks for density estimation over generalized random graphs. IEEE Trans Neural Netw Learn Syst 29(11):5441–5458
5. Borenstein M, Hedges LV, Higgins JPT, Rothstein HR (2009) Introduction to meta-analysis. Wiley, New York
6. Castillo E, Hadi A, Balakrishnan N, Sarabia J (2004) Extreme value and related models with applications in engineering and science. Wiley series in probability and statistics. Wiley
7. Cuthbertson K, Nitzsche D (2004) Quantitative financial economics: stocks, bonds and foreign exchange, 2nd edn. Wiley, New York
8. Duda RO, Hart PE, Stork DG (2000) Pattern classification, 2nd edn. Wiley, New York
9. Japkowicz N, Shah M (2011) Evaluating learning algorithms: a classification perspective. Cambridge University Press, Cambridge
10. Magdon-Ismail M, Atiya A (2002) Density estimation and random variate generation using multilayer networks. IEEE Trans Neural Netw 13(3):497–520
11. Modha DS, Fainman Y (1994) A learning law for density estimation. IEEE Trans Neural Netw 5(3):519–23
12. Newman MEJ, Barkema GT (1999) Monte Carlo methods in statistical physics. Oxford University Press
13. Ohl T (1999) VEGAS revisited: adaptive Monte Carlo integration beyond factorization. Comput Phys Commun 120:13–19
14. Peerlings D, van den Brakel J, Bastürk N, Puts M (2022) Multivariate density estimation by neural networks. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2022.3190220
15. Rissanen J (1978) Modeling by shortest data description. Automatica 14(5):465–471
16. Rubinstein RY, Kroese DP (2012) Simulation and the Monte Carlo method. Wiley
17. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536
18. Rust R, Schmittlein D (1985) A Bayesian cross-validated likelihood method for comparing alternative specifications of quantitative models. Market Sci 4(1):20–40
19. Schwenker F, Trentin E (2014) Pattern classification and clustering: a review of partially supervised learning approaches. Pattern Recognit Lett 37:4–14
20. Spall JC, Maryak JL (1992) A feasible Bayesian estimator of quantiles for projectile accuracy from non-i.i.d data. J Am Stat Assoc 87(419):676–681
21. Trentin E (2001) Networks with trainable amplitude of activation functions. Neural Netw 14(4–5):471–493
22. Trentin E (2015) Maximum-likelihood normalization of features increases the robustness of neural-based spoken human-computer interaction. Pattern Recognit Lett 66:71–80
23. Trentin E (2018) Maximum-likelihood estimation of neural mixture densities: Model, algorithm, and preliminary experimental evaluation. In: Pancioni L, Schwenker F, Trentin E (eds.) Artificial Neural Networks in Pattern Recognition—Proceedings of ANNPR 2018 (8th IAPR TC3 Workshop, Springer, pp 178–189
24. Trentin E (2018) Soft-constrained neural networks for nonparametric density estimation. Neural Process Lett 48(2):915–932
25. Trentin E (2020) Asymptotic convergence of soft-constrained neural networks for density estimation. Mathematics 8(4):572
26. Trentin E, Freno A (2009) Probabilistic interpretation of neural networks for the classification of vectors, sequences and graphs. Innovations in neural information paradigms and applications. Springer, pp 155–182
27. Trentin E, Lusnig L, Cavalli F (2018) Parzen neural networks: fundamentals, properties, and an application to forensic anthropology. Neural Netw 97:137–151
28. Trentin E, Scherer S, Schwenker F (2015) Emotion recognition from speech signals via a probabilistic echo-state network. Pattern Recognit Lett 66:4–12
29. Vapnik V (1995) The nature of statistical learning theory. Springer, New York
30. Vapnik VN, Mukherjee S (2000) Support vector method for multivariate density estimation. Advances in neural information processing systems. MIT Press, pp 659–665
31. Yuksel SE, Wilson JN, Gader PD (2012) Twenty years of mixture of experts. IEEE Trans Neural Netw Learn Syst 23:1177–1193