Maximum-Likelihood Estimation of Neural Mixture Densities: Model, Algorithm, and Preliminary Experimental Evaluation

Edmondo Trentin(✉)

Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche,
Università degli Studi di Siena, Siena, Italy
trentin@dii.unisi.it

Abstract. Unsupervised estimation of probability density functions by means of parametric mixture densities (e.g., Gaussian mixture models) may improve significantly over plain, single-density estimators in terms of modeling capabilities. Moreover, mixture densities (and even mixtures of mixture densities) may be exploited for the statistical description of phenomena whose data distributions implicitly depend on the distinct outcomes of a number of non-observable, latent states of nature. In spite of some recent advances in density estimation via neural networks, no proper mixtures of neural component densities have been investigated so far. The paper proposes a first algorithm for estimating Neural Mixture Densities based on the usual maximum-likelihood criterion, satisfying numerically a combination of hard and soft constraints aimed at ensuring a proper probabilistic interpretation of the resulting model. Preliminary results are presented and their statistical significance is assessed, corroborating the soundness of the approach with respect to established statistical techniques.

Keywords: Density estimation · Mixture density
Unsupervised learning · Constrained learning · Mixture of experts

1 Introduction

Density estimation is fundamental to a number of (apparently unrelated) tasks. Firs and foremost, it is at the core of the search for a statistical description of populations represented in terms of a sample of data distributed according to an unknown probability density function (pdf) [10]. Then, it is involved (possibly only implicitly) in the estimation of the probabilistic quantities that are necessary in order to apply Bayes decision rule for pattern classification, in particular the class-conditional probabilities [10]. Other tasks include data compression and model selection [11], coding [3], etc. Even the estimation of regression models may rely implicitly on density estimation, since it can be described as the

L. Pancioni et al. (Eds.): ANNPR 2018, LNAI 11081, pp. 178–189, 2018.
https://doi.org/10.1007/978-3-319-99978-4_14

estimation of a model $p(\mathbf{y}|\mathbf{x})$ that captures the statistical relationship between an independent random vector $\mathbf{x}$ and the corresponding output vector $\mathbf{y}$ [4]. As pointed out by Vapnik [28], 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 (the reader is referred to [25] for a list of the major drawbacks of the statistical techniques), and by the technical difficulties that arise from attempting to use artificial neural networks (ANNs) or machine learning for pdf estimation. Such difficulties stem from: (1) the unsupervised nature of the learning task, (2) the numerical instability problems entailed by pdfs, whose codomains may span the interval $[0, +\infty)$ in the general case, and (3) the requirement of mathematical plausibility of the estimated model, i.e. the respect of the axioms of probability. Furthermore, the use of maximum-likelihood (ML) training in ANNs tends to result in the so-called "divergence problem", observed first in the realm of hybrid ANN/hidden Markov models [20]. 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 [26], or in the RBF/graph neural network presented in [6] 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.

In spite of these difficulties, several approaches to pdf estimation via ANNs are found in the literature [23]. First of all, a ML technique is presented in [13] where the "integral equals 1" requirement is satisfied numerically 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 [13]. Nonetheless, this approach is related to the technique presented in this paper, insofar that ML will be exploited herein. Differently from [13], a multi-dimensional ad-hoc numeric integration method will be used in the following, jointly with hard constraints, over a mixture of ANNs. Other approaches found in the literature translated the estimation of univariate pdfs to the (theoretically equivalent) estimation of the corresponding cumulative distribution functions (cdf) [12,27]. Regular backpropagation (BP) is applied, relying on the empirical cdf of the data for generating synthetic target outputs. After training the MLP model $\phi(\cdot)$ of the cdf, the pdf can be recovered by applying differentiation to $\phi(\cdot)$. The idea is sound, since the requirements that $\phi(\cdot)$ has to satisfy to be interpretable as a proper cdf (namely, that it ranges between 0 and 1, and that it is monotonically non-decreasing) appear to be more easily met than the corresponding constraints on pdf models (that is, the unit integral). Unfortunately, there are drawbacks to the cdf-based

approaches (see [25]). In particular, a good approximation of the cdf does not necessarily translate into a similarly good estimate of its derivative. In fact, a small squared error between $\phi(\cdot)$ and the target cdf does not mean that $\phi(\cdot)$ is free from steep fluctuations that imply huge, rapidly changing values of its derivative. Negative values of $\frac{\partial \phi(x)}{\partial x}$ may occasionally occur, since a linear combination of logistics is not necessarily monotonically increasing. Besides, cdf-based algorithms naturally apply to univariate cases, whilst extension to multivariate pdfs is far less realistic. The idea of generating empirical target outputs was applied to non-parametric ANN-based pdf estimation in [22,24]. The former resorts to the k_n-Nearest Neighbor (k_n-NN) technique [10] for generating unbiased pdf estimates that are used to label the training set for a MLP. Like in the k_n-NN, the resulting model is not a proper pdf (the axioms of probability are not satisfied in the general case). On the other way around, the algorithm presented in [22] uses a modified criterion function to be minimized via gradient descent for pdf estimation via MLP. The criterion involves two terms: a loss between the MLP output and a synthetically-generated non-parametric estimate of the corresponding input pattern, and a loss between the integral of the function computed by the MLP and its target (i.e., unity) value. Numerical integration methods are used to compute the integral at hand and its derivatives w.r.t. the MLP parameters within the gradient-descent via backpropagation. The ideas behind such integration methods are exploited in this paper, as well.

A generalization of plain pdf estimation 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 [10]. GMMs are the most popular instance of mixture densities [5]. Traditionally, mixture densities were intended mostly as real-life extensions of the single-pdf parametric model, e.g. along the following line: one Gaussian may not be capable to explain the whole data distribution but K Gaussian pdfs might as well be, as long as K is large enough. Nevertheless, there is much more than this to the very notion of mixture density. In fact, different components 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. Examples of practical relevance include (among many others) the statistical study of heterogeneity in meta-analysis [7], 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; the modeling of unsupervised or partially-supervised [17] data samples in statistical pattern recognition [10], where each sub-population corresponds to a class or category; the distribution of financial returns on the stock market depending on latent phenomena such as a political crisis or a war [8]; 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 [18], etc. In general, the sub-populations in a mixture are

unlikely to be individually distributed according to simple (e.g., Gaussian) pdfs, therefore parametric models (e.g., GMMs) are seldom a good fit to these scenarios. 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 $p(\mathbf{x}|\xi_i)$ 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 the distributed according to the mixture $p(\mathbf{x}) = \sum_{i=1}^{K} P(\xi_i)p(\mathbf{x}|\xi_i)$. 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 $p(\mathbf{x}|\xi_i)$. In general, at the very least, more than one Gaussian component will be needed to model $p(\mathbf{x}|\xi_i)$. 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].

Given the aforementioned relevance and difficulties of estimating pdfs in general and mixture models in particular, and in the light of the above-named shortcomings of the established approaches, the paper contributes (for the first time) a plausible solution in the form of a mixture model built on ANNs. The model, presented in Sect. 2 and called neural mixture model (NMM), is a convex combination of component densities estimated by component-specific MLPs. The NMM is intrinsically non-parametric, since no prior assumptions on the form of the underlying component densities is made [10]. In fact, due to the "universality" of MLPs [9], the model may approximate any (bounded and continuous) multimodal multivariate pdf to any degree of precision[1]. Besides, due to the learning and generalization capabilities of ANNs, the NMM can actually learn a smooth and 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 [29] having a specific task, a ML-based unsupervised training algorithm, and a particular probabilistic strategy for assigning credit to its individual experts. A preliminary experimental evaluation is reported in Sect. 3, while Sect. 4 draws some pro tempore conclusions.

2 Model and Estimation Algorithm

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 identically distributed according to an unknown pdf $p(\mathbf{x})$. In order to estimate $p(\mathbf{x})$ from $\mathcal{T}$ we introduce a neural mixture model $\tilde{p}(\mathbf{x}|W)$ defined as

[1] According to the meaning of "approximation" and under the conditions required in order for (e.g.) Cybenko's theorem to hold true [9].

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

where W denotes the overall set of parameters in the NMM (that is $c_1, \ldots, c_K, W_1, \ldots, W_k$). The mixing coefficients c_i are such that $c_i \in [0,1]$ for $i = 1, \ldots, K$ and $\sum_{i=1}^{K} c_i = 1$, and the generic i-th component density $\tilde{p}_i(\mathbf{x}|W_i)$ is defined, in turn, as

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

where $\varphi_i(\mathbf{x}, W_i)$ represents the function computed by a component-specific MLP having adaptive parameters W_i. We say that this MLP realizes the i-th neural component of the NMM. A constraint on $\int \varphi_i(\mathbf{x}, W_i) d\mathbf{x}$ will be imposed shortly to enure satisfaction of the axioms of probability. Clearly, each MLP in the NMM has d input units and a single output unit, and it is expected to have one or more hidden layers. Without loss of generality for all the present intents and purposes, we assume that the patterns of interest are confined within a compact $S \subset \mathbb{R}^d$ (in practice, any data normalization technique may be applied in order to guarantee that this assumption holds true) such that, in turn, S can be seen as the definition domain of $\varphi_i(\mathbf{x}, W_i)$ for all $i = 1, \ldots, K$. As a consequence, numerical integration techniques can be used to compute $\int \varphi_i(\mathbf{x}, W_i) d\mathbf{x}$ and the other integrals required shortly. In so doing, Eq. (2) reduces to $\tilde{p}_i(\mathbf{x}|W_i) = \frac{\varphi_i(\mathbf{x}, W_i)}{\int_S \varphi_i(\mathbf{x}, W_i) d\mathbf{x}}$.

Some precautions are to be taken in regard to the nature of the activation function $f_i(.)$ used in the output layer of the i-th MLP. In fact, $f_i(.)$ shall be capable of spanning a codomain that fits the general definition of pdf, that is (in principle) any range in $[0, +\infty)$. Although this may be granted in several different ways, herein we opt for a logistic sigmoid with component-specific adaptive amplitude $\lambda_i \in \mathbb{R}^+$, namely $f_i(a_i) = \lambda_i / (1 + \exp(-a_i))$ as described in [19], where a_i represents the current activation value for the output unit of the i-th neural component. Consequently, each MLP in the NMM can stretch its output over any required component-specific interval $[0, \lambda_i)$, which is not bounded a priori but is rather learned (along with the other parameters in W_i) so as to fit the nature of the specific component density at hand. Other general advantages entailed by the use of adaptive amplitudes are pointed out in [19].

The training algorithm is expected to revolve around a proper learning rule for the mixture parameters W given the unlabeled sample $\mathcal{T}$, such that eventually $\tilde{p}(\mathbf{x}|W)$ results in a proper estimate of $p(\mathbf{x})$. This requires pursuing two purposes: (1) exploiting the information encapsulated in $\mathcal{T}$ to approximate the unknown pdf; (2) preventing the MLPs in the NMM 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 end, a constrained stochastic gradient-ascent algorithm is devised that aims at the maximization of the point-wise likelihood $\tilde{p}(\mathbf{x}_j|W)$ of the NMM given the current training pattern $\mathbf{x}_j$, to be applied iteratively for $j = 1, \ldots, n$. This is achieved by means of an on-line, differentiable criterion function $C(.)$ defined as

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

that has to be maximized with respect to the NMM 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 criterion, instead, is 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, resulting in $\int_S \tilde{p}(\mathbf{x}|W) d\mathbf{x} \simeq 1$. The hyper-parameter $\rho \in \mathbb{R}^+$ controls the importance of the constraints, and it is used in practical applications to tackle numerical issues. The gradient-ascent learning rule Δw for a generic parameter w in the NMM is then defined as $\Delta w = \eta \frac{\partial C(.)}{\partial w}$, where $\eta \in \mathbb{R}^+$ is the learning rate. Different calculations are needed, according to the fact that w is either: (i) a mixing coefficient, say $w = c_k$; or (ii) a parameter (connection weight, bias, or adaptive amplitude) within any of the neural component densities. In case (i), we first 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 $\varsigma(x) = 1/(1 + e^{-x})$. Each γ_k is then treated as the unknown parameter to be actually estimated instead of the corresponding c_k. In so doing, higher-likelihood mixing coefficients that satisfy the required constraints are implicitly obtained from application of the learning rule. The latter takes the following form:

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

Secondly, let us move to scenario (ii), that is where w is a parameter within one of the neural components. In this case, 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). In the following calculations we assume that w belongs to the (generic) k-th neural component. For the first term we have:

$$\begin{aligned}\frac{\partial \tilde{p}(\mathbf{x_j}|W)}{\partial w} &= \frac{\partial}{\partial w}\sum_{i=1}^{K} c_i \tilde{p}_i(\mathbf{x_j}|W_i) \\ &= \frac{\partial}{\partial w}\{c_k \tilde{p}_k(\mathbf{x_j}|W_k)\} \\ &= c_k \frac{\partial}{\partial w}\left\{\frac{\varphi_k(\mathbf{x}_j, W_k)}{\int_S \varphi_k(\mathbf{x}, W_k)d\mathbf{x}}\right\} \\ &= c_k \left\{\frac{1}{\int_S \varphi_k(\mathbf{x}, W_k)d\mathbf{x}}\frac{\partial \varphi_k(\mathbf{x}_j, W_k)}{\partial w} - \frac{\tilde{p}_k(\mathbf{x}_j, W_k)}{\int_S \varphi_k(\mathbf{x}, W_k)d\mathbf{x}}\frac{\partial}{\partial w}\int_S \varphi_k(\mathbf{x}, W_k)d\mathbf{x}\right\} \\ &= \frac{c_k}{\int_S \varphi_k(\mathbf{x}, W_k)d\mathbf{x}}\left\{\frac{\partial \varphi_k(\mathbf{x}_j, W_k)}{\partial w} - \frac{\varphi_k(\mathbf{x}_j, W_k)}{\int_S \varphi_k(\mathbf{x}, W_k)d\mathbf{x}}\int_S \frac{\partial \varphi_k(\mathbf{x}, W_k)}{\partial w}d\mathbf{x}\right\}\end{aligned} \tag{6}$$

where Leibniz rule was exploited in the last step of the calculations. Note that Eq. (6) is a mathematical statement of the rationale behind the different impact that current training pattern $\mathbf{x}_j$ has on the learning process in distinct neural components of the NMM. First, the amount of parameter change Δw is proportional to the probabilistic "credit" c_k of the component at hand. Second (and foremost), the quantities within brackets in Eq. (6) depend on the value of the k-th MLP output over $\mathbf{x}_j$, and on its derivative. If, at any time during the training, $\varphi_k(.)$ 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 network will be paramount if $\varphi_k(.)$ returns high likelihood over $\mathbf{x}_j$ and significant variations in its surroundings.

At this point 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

$$\begin{aligned}&\frac{\partial}{\partial w}\left\{\rho\sum_{i=1}^{K}\frac{1}{2}\left(1-\int_S \varphi_i(\mathbf{x}, W_i)d\mathbf{x}\right)^2\right\} \\ &= \frac{\partial}{\partial w}\left\{\frac{\rho}{2}\left(1-\int_S \varphi_k(\mathbf{x}, W_k)d\mathbf{x}\right)^2\right\} \\ &= -\rho\left(1-\int_S \varphi_k(\mathbf{x}, W_k)d\mathbf{x}\right)\frac{\partial}{\partial w}\int_S \varphi_k(\mathbf{x}, W_k)d\mathbf{x} \\ &= -\rho\left(1-\int_S \varphi_k(\mathbf{x}, W_k)d\mathbf{x}\right)\int_S \frac{\partial \varphi_k(\mathbf{x}, W_k)}{\partial w}d\mathbf{x}.\end{aligned} \tag{7}$$

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 $\int_S \frac{\partial}{\partial w}\varphi_k(\mathbf{x}, W_k)d\mathbf{x}$ are needed in order to compute the right-hand side of Eqs. (6) and (7). The quantity $\frac{\partial \varphi_k(\mathbf{x}_j, W_k)}{\partial w}$ is the usual partial derivative of the output of a MLP with respect to one of its parameters, and it is computed via plain BP (or, as in [19] if $w = \lambda_k$). As for the integrals, deterministic numerical quadrature integration techniques (e.g., Simpson's method, trapezoidal rule, etc.) are viable

only for $d = 1$, since they do not scale up computationally to higher dimensions ($d \geq 2$) of the feature space. This is all the more critical if we bear in mind that $\int_S \frac{\partial}{\partial w}\varphi_k(\mathbf{x}, W_k)d\mathbf{x}$ has to be computed iteratively and individually for each parameter of each neural component in the NMM. Besides, deterministic integration methods do not exploit the very nature of the function to be integrated. In fact, in the present context the integrand is expected to be closely related to the pdf (say, $p_k(\mathbf{x})$) that explains the distribution of the specific sub-sample of data drawn from the k-th component of the mixture. In fact, accounting for the pdf of the data should drive the integration algorithm towards integration points that cover "interesting" (i.e., having high component-specific likelihood) regions of the domain of the integrand. For these reasons, we use a component-oriented variant of the approach we proposed in [21], that can be seen as an instance of Markov chain Monte Carlo [1]. It is a non-deterministic, multi-dimensional integration technique which actually accounts for the component-specific probability distribution of the data. Let $\phi_k(\mathbf{x})$ denote the integrand of interest (either $\varphi_k(\mathbf{x}, W_k)$ or $\frac{\partial \varphi_k(\mathbf{x}, W_k)}{\partial w}$). An approximation of the integral of $\phi_k(\mathbf{x})$ over S is obtained via Monte Carlo with importance sampling [16] as $\int_S \phi_k(\mathbf{x})d\mathbf{x} \simeq \frac{V(S)}{m}\sum_{\ell=1}^{m}\phi_k(\dot{\mathbf{x}}_\ell)$ 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 attained by drawing it at random from the mixture density $p_u^{(k)}(\mathbf{x})$ defined as

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

where $u(\mathbf{x})$ is the uniform distribution over S, and $\alpha : \mathbb{N} \to (0, 1)$ is a decaying function of the number t of the NMM training epochs (a training epoch is a completed re-iteration of Eqs. (6) and (7) for all the parameters of the NMM over all the observations in $\mathcal{T}$) for $t = 1, \ldots, T$, such that $\alpha(1) \simeq 1.0$ and $\alpha(T) \simeq 0.0$. As in [21] we let $\alpha(t) = 1/(1 + e^{\frac{t/T - 1/2}{\theta}})$, where θ is a hyperparameter. Eq. (8) is such that the importance sampling mechanism it implies accounts for the (estimated) component density $\tilde{p}_k(\mathbf{x}|W_k)$ of the k-th latent subsample of the data, therefore respecting the natural distribution of such sub-population and focusing integration on the relevant (i.e., having high component-specific likelihood) integration points. On the other hand, since the estimates of this component densities are not reliable during the early stage of the NMM training process, Eq. (8) prescribes a (safer) sampling from a uniform distribution at the beginning, like in the plain Monte Carlo algorithm. As the robustness of the NMM estimates increases, i.e. as t increases, sampling from $\tilde{p}_k(\mathbf{x}|W_k)$ replaces progressively the sampling from $u(\mathbf{x})$, ending up in entirely non-uniform importance sampling. The form of $\alpha(t)$ is such that the algorithm basically sticks with the uniform sampling for quite some time before beginning crediting the estimated pdfs, but it ends up relying mostly on the neural component densities throughout the advanced stage of training. Since $\varphi_k(\mathbf{x}, W_k)$ is non-negative by construction, for $t \to T$ 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 [15].

Sampling from $p_u^{(k)}(\mathbf{x})$ requires an effective method for sampling from the output of the k-th MLP in the NMM. A specialization of Markov chain Monte Carlo, namely the Metropolis–Hastings (M-H) algorithm [14], is used herein. 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) [14]. 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 $q(\mathbf{x}'|\mathbf{x})$ 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 $q(\mathbf{x}'|\mathbf{x}, \sigma) = \prod_{i=1}^{d} \frac{1}{\sigma} e^{(x'_i - x_i)/\sigma} (1 + e^{(x'_i - x_i)/\sigma})^{-2}$ which can be sampled readily by means of the inverse transform sampling technique. Any approach to model selection can be applied to fix empirically the values for the present hyperparameters (that is, the scale σ of the proposal pdf and the burn-in period for M-H).

3 Preliminary Experimental Evaluation

This Section presents a preliminary evaluation of the NMM behavior on random samples generated synthetically from a multimodal pdf $p(\cdot)$ having known form. The random samples used in the experiments were drawn from mixtures $p(x)$ of c Fisher-Tippett pdfs, that is $p(x) = \sum_{i=1}^{c} \frac{P_i}{\beta_i} \exp\left(-\frac{x-\mu_i}{\beta_i}\right) \exp\left\{-\exp\left(-\frac{x-\mu_i}{\beta_i}\right)\right\}$. The mixing parameters $P_1, \ldots, P_c$ were drawn at random (any time a new dataset had to be generated) from the uniform distribution over [0.1] and normalized such that $\sum_{i=1}^{c} P_i = 1$. The component densities of the Fisher-Tippett mixture are identified by their locations μ_i and scales β_i, respectively, for $i = 1, \ldots, c$. The locations were drawn at random from the uniform distribution over $(0, 10)$, while the scales were randomly and uniformly distributed over $(0.01, 0.9)$. The estimation tasks we faced involved 1200 patterns randomly drawn from $p(x)$, and a variable number c of component densities, namely $c = 5$, 10, 15 and 20, respectively. Each c-specific dataset was split into a training set ($n = 800$ patterns) and a validation set (the remaining 400 patterns). The integrated squared error (ISE) between $p(x)$ and its estimate $\tilde{p}(x)$, that is $\int (p(x) - \tilde{p}(x))^2 \, dx$, is used as the criterion of assessment of the performance of the estimation techniques. The ISE is computed numerically via Simpson's method.

In the present experiments we used NMMs involving MLPs with a single hidden layer of 9 units. Logistic sigmoid activation functions were used, having layer-wise [19] trainable amplitudes λ. All the parameters in the NMM were initialized at random over small intervals centered at zero, except for the amplitudes λ that were initially set to 1 and the mixing coefficients that were set to $c_i = 1/K$ for $i = 1, \ldots, K$. As in [22] we let $\theta = 0.07$ in the definition of the function $\alpha(t)$, and the number of integration points was set to $m = 400$, 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 hyperparameters of the NMM were fixed via random-search based on the cross-validated likelihood criterion exploiting the validation set. Input data were not normalized. Results are reported in Table 1. NMMs having different values of K (that is $K = 4$, 8 and 12) were evaluated, and compared with the most prominent statistical techniques, parametric and non-parametric, namely: 8-, 16- and 32-GMM initialized via k-means and refined via iterative maximum-likelihood re-estimation [10], k_n-NN with unbiased $k_n = 1\sqrt{n}$ [10], and Parzen Window (PW) with unbiased bandwidth $h_n = 1/\sqrt{n}$ of the Gaussian kernels [10].

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 Fisher-Tippett component densities. Best results are shown in boldface. (*Legend*: k-GMM and k-NMM denote the GMM and the NMM with k components, respectively).

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

It is seen that all the models yield values of the resulting ISE that tend to increase as a function of c. Nevertheless, the NMM improves systematically and significantly over all the statistical approaches regardless of c. On average, both the 8-NMM and the 16-NMM offer a 46% relative ISE reduction over the PW (the PW turns out to be the closest competitor to NMM in the present setup). Welch's t-test returns a level of confidence >90% on the statistical significance of the gap between the results yielded by the 8-NMM (or, by the 12-NMM) and by the PW, respectively. Moreover, the NMM results in the stablest estimators overall, as shown by the values of the standard deviations (last column of the table). This is evidence of the fact that the degree of fitness of the NMMs to the true pdf is less sensitive to the complexity of the underlying Fisher-Tippett mixture (that is, to c) than that yielded by the traditional statistical models. As expected, ISE differences are observed among the k-NMMs depending on the different values of k. In the present setup, differences between the 8-NMM and the 12-NMM turn out to be mild (especially if compared to the overall gaps between NMMs and the established statistical techniques), and they depend on the complexity of the underlying pdf to be estimated, at least to some extent, as expected.

4 Pro Tempore Conclusions and On-Going Research

Density estimation remains an open, non-trivial task: in fact, significant improvement over established approaches may still be expected of novel techniques, capable of increased robustness of the resulting pdf estimates. In spite of the relative simplicity of the data used in the previous Section, the empirical evidence reported therein confirms that the traditional statistical techniques may yield inaccurate estimates, whereas the NMM may result in a model of the data that is closer to the true pdf underlying the unlabeled samples at hand. For the time being, we are in the early stages of investigating the behavior of the technique over multivariate random vectors. Model selection (in particular, selection of proper ANN architectures) is under further investigation, as well, by exploiting the implicit availability of a mathematically grounded assessment criterion, namely the likelihood of the model (e.g., of its architecture) given a validation set. Finally, another facet of the NMM that is currently under development lies in the initialization procedure: non-uniform initialization of the mixing coefficients may turn out to be helpful in breaking potential ties, and initializing the individual neural components via BP learning of a subset of the components of a pre-estimated reference mixture model (i.e., a GMM) may also fruitfully replace the bare random initialization of the MLPs parameters.

References

1. Andrieu, C., de Freitas, N., Doucet, A., Jordan, M.I.: An introduction to MCMC for machine learning. Mach. Learn. **50**(1–2), 5–43 (2003)
2. Aste, M., Boninsegna, M., Freno, A., Trentin, E.: Techniques for dealing with incomplete data: a tutorial and survey. Pattern Anal. Appl. **18**(1), 1–29 (2015)
3. Beirami, A.: Wireless network compression via memory-enabled overhearing helpers. IEEE Trans. Wirel. Commun. **15**(1), 176–190 (2016)
4. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
5. Bishop, C.M.: Pattern Recognition and Machine Learning. Information Science and Statistics. Springer, Heidelberg (2006). https://doi.org/10.1007/978-1-4615-7566-5
6. Bongini, M., Rigutini, L., Trentin, E.: Recursive neural networks for density estimation over generalized random graphs. IEEE Trans. Neural Netw. Learn. Syst. (2018). https://doi.org/10.1109/TNNLS.2018.2803523
7. Borenstein, M., Hedges, L.V., Higgins, J.P.T., Rothstein, H.R.: Introduction to MetaAnalysis. Wiley-Blackwell, New York (2009)
8. Cuthbertson, K., Nitzsche, D.: Quantitative Financial Economics: Stocks, Bonds and Foreign Exchange, 2nd edn. Wiley, New York (2004)
9. Cybenko, G.: Approximation by superposition of sigmoidal functions. Math. Control Signal Syst. **2**(4), 303–314 (1989)
10. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley-Interscience, New York (2000)
11. Liang, F., Barron, A.: Exact minimax strategies for predictive density estimation, data compression, and model selection. IEEE Trans. Inf. Theory **50**(11), 2708–2726 (2004)

12. Magdon-Ismail, M., Atiya, A.: Density estimation and random variate generation using multilayer networks. IEEE Trans. Neural Netw. **13**(3), 497–520 (2002)
13. Modha, D.S., Fainman, Y.: A learning law for density estimation. IEEE Trans. Neural Netw. **5**(3), 519–23 (1994)
14. Newman, M.E.J., Barkema, G.T.: Monte Carlo Methods in Statistical Physics. Oxford University Press, Oxford (1999)
15. Ohl, T.: VEGAS revisited: adaptive Monte Carlo integration beyond factorization. Comput. Phys. Commun. **120**, 13–19 (1999)
16. Rubinstein, R.Y., Kroese, D.P.: Simulation and the Monte Carlo Method, 2nd edn. Wiley, Hoboken (2012)
17. Schwenker, F., Trentin, E.: Pattern classification and clustering: a review of partially supervised learning approaches. Pattern Recognit. Lett. **37**, 4–14 (2014)
18. Spall, J.C., Maryak, J.L.: A feasible bayesian estimator of quantiles for projectile accuracy from non-i.i.d. data. J. Am. Stat. Assoc. **87**(419), 676–681 (1992)
19. Trentin, E.: Networks with trainable amplitude of activation functions. Neural Netw. **14**(45), 471–493 (2001)
20. Trentin, E.: Maximum-likelihood normalization of features increases the robustness of neural-based spoken human-computer interaction. Pattern Recognit. Lett. **66**, 71–80 (2015)
21. Trentin, E.: Soft-constrained nonparametric density estimation with artificial neural networks. In: Schwenker, F., Abbas, H.M., El Gayar, N., Trentin, E. (eds.) ANNPR 2016. LNCS (LNAI), vol. 9896, pp. 68–79. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46182-3_6
22. Trentin, E.: Soft-constrained neural networks for nonparametric density estimation. Neural Process. Lett. (2017). https://doi.org/10.1007/s11063-017-9740-1
23. Trentin, E., Freno, A.: Probabilistic interpretation of neural networks for the classification of vectors, sequences and graphs. In: Bianchini, M., Maggini, M., Scarselli, F., Jain, L.C. (eds.) Innovations in Neural Information Paradigms and Applications. SCI, vol. 247, pp. 155–182. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04003-0_7
24. Trentin, E., Freno, A.: Unsupervised nonparametric density estimation: a neural network approach. In: Proceedings of the International Joint Conference on Neural Networks, IJCNN 2009, pp. 3140–3147 (2009)
25. Trentin, E., Lusnig, L., Cavalli, F.: Parzen neural networks: fundamentals, properties, and an application to forensic anthropology. Neural Netw. **97**, 137–151 (2018)
26. Trentin, E., Scherer, S., Schwenker, F.: Emotion recognition from speech signals via a probabilistic echo-state network. Pattern Recognit. Lett. **66**, 4–12 (2015)
27. Vapnik, V.N., Mukherjee, S.: Support vector method for multivariate density estimation. In: Advances in Neural Information Processing Systems, pp. 659–665. MIT Press (2000)
28. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995). https://doi.org/10.1007/978-1-4757-2440-0
29. Yuksel, S.E., Wilson, J.N., Gader, P.D.: Twenty years of mixture of experts. IEEE Trans. Neural Netw. Learn. Syst. **23**, 1177–1193 (2012)