Penalized Regression Splines in Mixture Density Networks

Quentin Edward Seifert (quentinedward.seifert@uni-goettingen.de)  
Chair of Spatial Data Science and Statistical Learning, Georg-August-Universität Göttingen

Anton Thielmann  
Chair of Data Science and Applied Statistics, TU Clausthal

Elisabeth Bergherr  
Chair of Spatial Data Science and Statistical Learning, Georg-August-Universität Göttingen

Benjamin Säfken  
Chair of Data Science and Applied Statistics, TU Clausthal

Jakob Zierk  
Department of Pediatrics and Adolescent Medicine, University Hospital Erlangen

Manfred Rauh  
Department of Pediatrics and Adolescent Medicine, University Hospital Erlangen

Tobias Hepp  
Department of Medical Informatics, Biometry and Epidemiology, Friedrich-Alexander-Universität Erlangen-Nürnberg

Research Article

Keywords: Neural Networks, Distributional Regression, Finite Mixture Models, Regression Splines

Posted Date: December 23rd, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2398185/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

Additional Declarations: No competing interests reported.
Penalized Regression Splines in Mixture Density Networks

Quentin Edward Seifert\(^1\)*, Anton Thielmann\(^2\), Elisabeth Bergherr\(^1\), Benjamin Säffen\(^2\), Jakob Zierk\(^3\), Manfred Rauh\(^3\) and Tobias Hepp\(^1,4\)

\(^1\)Chair of Spatial Data Science and Statistical Learning, Georg-August-Universität Göttingen, Germany.
\(^2\)Chair of Data Science and Applied Statistics, TU Clausthal.
\(^3\)Department of Pediatrics and Adolescent Medicine, University Hospital Erlangen, Germany.
\(^4\)Department of Medical Informatics, Biometry and Epidemiology, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany.

*Corresponding author. E-mail: quentinedward.seifert@uni-goettingen.de.

Abstract

Mixture Density Networks (MDN) belong to a class of models that can be applied to data which cannot be sufficiently described by a single distribution since it originates from different components of the main unit and therefore needs to be described by a mixture of densities. In some situations, however, MDNs seem to have problems with the proper identification of the latent components. While these identification issues can to some extent be contained by using custom initialization strategies for the network weights, this solution is still less than ideal since it involves subjective opinions. We therefore suggest replacing the hidden layers between the model input and the output parameter vector of MDNs and estimating the respective distributional parameters with penalized cubic regression splines. Applying this approach to data from Gaussian mixture distributions as well gamma mixture distributions proved to be successful with the identification issues not playing a role anymore and the splines reliably converging to the true parameter values.

Keywords: Neural Networks, Distributional Regression, Finite Mixture Models, Regression Splines

1 Introduction

With the broad availability of programming libraries such as Tensorflow (Abadi et al., 2015), Keras (Chollet et al., 2015) or Pytorch (Paszke et al., 2019), artificial neural networks have become increasingly popular in recent years. Their flexibility, scalability as well as the impressive results they can achieve in many different applications make them a powerful tool in many fields. However, their ability to model complex relationships between covariates using multiple layers of densely connected artificial neurons is also the source of their biggest weakness, their uninterpretability.

Mixture Density Networks (MDN), as introduced in Bishop (1994) are a special type of Neural Network that is designed to deal with data that cannot be sufficiently described by one distribution but rather has to be considered to originate from multiple components. The goal of a MDN is to estimate all parameters of a mixture distribution dependent on one or multiple covariates, hence they show some similarities to distributional...
regression models such as generalized additive models for location scale and shape (GAMLSS) (Rigby and Stasinopoulos, 2005) as well as mixture regression models (Gn{"u}n and Leisch, 2004).

A recent important application area for distributional regression mixture models concerns the so-called indirect estimation of reference distributions for medical decision making (Hepp et al., 2020, 2022). Reference distributions can be used to derive reference intervals for specific analytes which are needed to help decide whether a patient can be considered healthy or not. The established direct approaches for the identification of reference intervals usually involve carefully designed studies and meticulous choice of participants and thereby avoid having to work with databases that can be assumed to include unhealthy patients (Solberg, 1987), whereas indirect estimation approaches make use of data that gets collected in the everyday routine of clinics and laboratories. However, databases from such sources are assumed to be "contaminated" with pathological observations. While indirect estimation approaches bear great potential for the estimation of reliable reference intervals, especially if one is interested in estimating reference intervals depending on continuous covariates, indirect estimation necessitates the separation of the distributions of healthy and unhealthy patients.

While the usage MDNs in this context might seem promising in concept, the simulation studies in Hepp et al. (2022), reveal a strong influence of the network initialization on the results, eventually resulting in severe identification problems which lead to the model switching up the latent components in a large proportion of the simulation runs. Considering the rather simple setup of the study, these results raise doubts about the reliability of MDNs in more complex applications with multiple input variables and where less prior knowledge about the true latent components might be available to assess the plausibility of the estimates.

To solve the identifiability problem encountered in Hepp et al. (2022), we suggest to enhance the conventional MDN architecture by estimating the respective parameters using penalized regression splines with the intuition that the regularization might keep the model from misidentifying the components. Another advantage of our proposed network architecture is that using regression splines would also allow for easier interpretability in more complex models as compared to the conventional MDN.

The next section provides an overview of the basics of generalized additive models and the general concept of mixture density networks and how those two model classes can be fused. In section 3, the revised model architecture is applied to simulated datasets to see how it performs compared to the conventional MDN. In section 4, the model is applied to a medical database of hemoglobin levels of female patients between 1 and 18 years, which is part of the PEDREF reference interval initiative (PED). Section 5 concludes.

2 Methods

With the general idea behind the methodology of this paper being the application of non-parametric regression within the context of artificial neural networks, this section gives a brief introduction into the basic structure and ideas behind neural networks as well as an introduction to non-parametric regression before merging both ideas and introducing the novel network architecture which will be used throughout this paper.

2.1 Neural Networks

Artificial neural networks are generally regarded as incredibly flexible models, which can be constructed in ways to work with any kind of data, be it structured or unstructured, that perform astonishingly well on a wide variety of tasks, such as image and object classification (Krizhevsky et al., 2012; Qi et al., 2017) but also conventional economic applications such as loan default risk prediction (Lin et al., 2022). The major disadvantage, however, is that neural networks are generally considered to be non-interpretable black boxes (Shwartz-Ziv and Tishby, 2017; Agarwal et al., 2021). The data is fed into a multi-layer network which consists of a large amount of densely connected neurons, which makes it hard to tell how the input data affects the network output. While neural networks have proven to be of great use for prediction and classification, their lack of interpretability makes them unsuited if the intent is to actually understand the relationship between input data and model output.
Generally speaking, artificial neural networks are a collection of interconnected neurons that communicate by receiving inputs and forwarding outputs. The neurons are organized in layers, starting with the input layer and ending with the output layer, with the individual neurons being connected between layers (Goodfellow et al., 2016). A neuron, which is not in the input layer, receives a weighted sum of the outputs from the neurons to which it is connected in the layer below as an input, based on which it creates an output using an activation function (Ramachandran et al., 2017). This output is then forwarded to the neurons in the next layer. The output layer determines the output a network creates for a given input. Artificial neural networks get their name from the fact that the general idea of such a network architecture is loosely based on our understanding of how information is processed in the brain (Svozil et al., 1997).

2.2 Non-parametric Regression

In a simple univariate model with response \( y_i \), covariate \( x_i \) and normally distributed error term \( \epsilon_i \) where

\[
y_i = f(x_i) + \epsilon_i, \quad i = 1, \ldots, n
\]

\( f(x_i) \) can be estimated by treating it as a linear combination of basis functions evaluated such that

\[
f(x_i) = \sum_{j=0}^{k} \gamma_j B_j(x_i).
\]

With \( y = (y_1, \ldots, y_n)' \), \( \epsilon = (\epsilon_1, \ldots, \epsilon_n)' \), \( \gamma = (\gamma_1, \ldots, \gamma_k)' \) and

\[
Z = \begin{pmatrix} B_1(x_1) & \cdots & B_k(x_1) \\ \vdots & \ddots & \vdots \\ B_1(x_n) & \cdots & B_k(x_n) \end{pmatrix}, \tag{2}
\]

the model in (1) can now be expressed as

\[
y = Z \gamma + \epsilon. \tag{3}
\]

For continuous data, \( f(x) \) is usually estimated using regression splines, such as cubic regression splines, thin plate regression splines or P-Splines (Wood, 2003; Eilers and Marx, 1996). Smooth terms for discrete data include Gauss Markov random fields or random effects (Wood, 2017). For the model above, the parameter vector \( \gamma \) could be estimated by minimizing the penalized least squares criterion

\[
\text{PLS}(\lambda) = (y - Z \gamma)'(y - Z \gamma) + \lambda \gamma' S \gamma, \tag{3}
\]

which is simply the ordinary least squares criterion with an additional penalty term that controls the smoothness of the corresponding smooth term via the smoothing parameter \( \lambda \geq 0 \) and the penalty matrix \( S \).

Regression splines are often used in the context generalized additive models (GAM, see Hastie and Tibshirani (1986)), models in which the predictor is the sum of one or multiple smooth terms and an optional parametric part such that

\[
\eta_i = X_i \beta + f_1(z_{1i}) + f_2(z_{2i}) + f_3(z_{3i}) + \ldots,
\]

and \( \eta_i = g(\mu_i) \) and \( h(\eta_i) = \mu_i \) with \( g() \) being the link function, \( h() \) being the response function and \( \mu_i \) being the canonical parameter of a random variable from the exponential family.

A notable extension of GAMs are GAMLSS, a framework which allows for the estimation of multiple distributional parameters, so with distribution parameters \( \theta_1, \ldots, \theta_J \), the parameters are estimated as

\[
f_j(\theta_j) = X_j \beta_j + \sum_{k=1}^{k} Z_{jk} \gamma_{jk}.
\]

For the purpose of this work, the spline regression model will be interpreted as a neural network. Considering that the input a single neuron in a neural network receives is simply a weighted sum of the previous outputs, the similarities between splines and neural networks are quite obvious. One can simply regard the basis functions as inputs of the output neurons, the corresponding regression parameters as the network weights and biases. Instead of using classical fitting methods, these models can be trained as a simple neural network with the common first-order gradient descent based optimizers. This reinterpretation of the spline regression model as neural networks allows us to apply concepts of spline regression in the context of neural networks and allows their
integration into more complex network architectures.

2.3 Mixture Density Networks

As we are dealing with data where we assume the overall population to consist of multiple components, we are in the context of finite mixture models and mixture densities. Formally, mixture densities are defined as a weighted sum of the densities of its components. A mixture density consisting of $M$ components is described as

$$g(y_i) = \sum_{m=1}^{M} \alpha_m g_m(y_i, \theta_m),$$  \hspace{1cm} (4)

with mixture weights $\alpha_m$ and $\sum_{m=1}^{M} \alpha_m = 1$ and the distribution parameters of the $m$-th component $\theta_m$, for the mixture of normal example in Figure 1 $\theta_m = (\mu_m, \sigma_m)$, with mean $\mu_m$ and standard deviation $\sigma_m$. The corresponding likelihood function of the mixture density is simply the product over the $N$ observations, so

$$L(\alpha, \theta) = \prod_{n=1}^{N} \sum_{m=1}^{M} \alpha_m g_m(y_i, \theta_m).$$  \hspace{1cm} (5)

In many cases, the assumption of constant distribution parameters and mixture weights is rather restrictive and the resulting models may not be able to accurately estimate the true underlying distributions. Therefore, we extend (4) by assuming the distribution parameters as well as the mixture weights to depend on one or multiple covariates in $x$:

$$g(y_i | x_i) = \sum_{m=1}^{M} \alpha_m(x_i) g_m(y_i, \theta_m(x_i)).$$

Mixture density networks (Bishop, 1994) are a special type of artificial neural network that allow for the modelling of all parameters of mixture distributions dependent on covariates, including its mixture weights. The general idea is to feed the observations $x_i$ into a network of one or more hidden layers that lead into an output layer consisting of as many neurons as there are parameters to be estimated. The loss function to be minimized is the negative log-likelihood of the mixture density. An exemplary structure of MDNs is depicted in the left panel of Figure 1.

For a mixture of normal distributions, the scale parameters are usually activated using the softplus activation function, the mixture weights $\alpha_m$ are activated using the softmax activation function which normalizes an input vector into a probability distributions so $\sum \alpha_m = 1$ with the respective $\alpha_m$ being restricted to the interval $[0, 1]$, and the means are simply activated linearly. While not implemented this way in Tensorflow, for the case $M = 2$ it suffices to only model one of the mixture weights explicitly using for instance the sigmoid-function or any other activation that limits it to $[0, 1]$ and then simply setting $\alpha_2 = 1 - \alpha_1$. For the sake of creating sparser models, we take this approach.

The framework allows to easily adjust the output layer for other distributions, such as for example the gamma distribution. Apart from the loss function, the gamma model only differs in that the mean parameters are also activated using softplus since the gamma distribution does not allow for negative values.

2.3.1 Revised MDN-Architecture

To tackle the component switching issue that was encountered in Hepp et al. (2022) when using conventional mixture density networks, we propose a new network architecture which uses penalized regression splines. Hepp et al. (2022) observed their networks to converge in local minima of the loss function, the intuition behind the inclusion of penalized splines is that the penalty might keep the estimated curves from making the jumps from one component to the other which ultimately leads to said convergence in local minima.

The revised network architecture is as depicted in the right panel of Figure 1, hence the input is now $k$-dimensional and comprised of the basis function evaluated at $x$ and feeds directly into the output layer. For our networks, we use cubic regression splines which were implemented in python closely following the implementation in the R-package mgcv and Wood (2017). Furthermore, the splines are "centered" around zero such that $\sum_i f(x_i) = 0$. Other spline types such as penalized B-splines (P-splines) or thin plate regression splines could also be used.

In matrix notation, the model can be expressed with design matrix $Z$ and the weight matrix $W$,
whose first row contains the biases $b$ and the matrix of predictors $H$, such that

$$
\begin{pmatrix}
1 & B_1(x_1) & \ldots & B_k(x_1) \\
\vdots & \vdots & \ddots & \vdots \\
1 & B_1(x_n) & \ldots & B_k(x_n)
\end{pmatrix}
\begin{pmatrix}
b_1 \\
b_2 \\
\vdots \\
b_5 \\
w_{11} \\
w_{12} \\
\vdots \\
w_{15} \\
w_{k1} \\
w_{k2} \\
\vdots \\
w_{k5}
\end{pmatrix} =
\begin{pmatrix}
1 \\
\vdots \\
1
\end{pmatrix}
\begin{pmatrix}
\eta_{\mu_{11}} & \eta_{\mu_{21}} & \ldots & \eta_{\mu_{1k}} \\
\eta_{\mu_{12}} & \eta_{\mu_{22}} & \ldots & \eta_{\mu_{1k}} \\
\vdots & \vdots & \ddots & \vdots \\
\eta_{\mu_{1n}} & \eta_{\mu_{2n}} & \ldots & \eta_{\mu_{1n}} \\
\eta_{\sigma_{11}} & \eta_{\sigma_{21}} & \ldots & \eta_{\sigma_{1k}} \\
\eta_{\sigma_{12}} & \eta_{\sigma_{22}} & \ldots & \eta_{\sigma_{1k}} \\
\vdots & \vdots & \ddots & \vdots \\
\eta_{\sigma_{1n}} & \eta_{\sigma_{2n}} & \ldots & \eta_{\sigma_{1n}} \\
\eta_{\alpha_{11}} & \eta_{\alpha_{21}} & \ldots & \eta_{\alpha_{1k}} \\
\eta_{\alpha_{12}} & \eta_{\alpha_{22}} & \ldots & \eta_{\alpha_{1k}} \\
\vdots & \vdots & \ddots & \vdots \\
\eta_{\alpha_{1n}} & \eta_{\alpha_{2n}} & \ldots & \eta_{\alpha_{1n}}
\end{pmatrix}
$$

hence the predictors of the distribution parameters of the first component for the $i$-th observation are

$$
\begin{align*}
\eta_{\mu_{1i}} &= b_1 + \sum_{j=1}^{k} w_{1j} B_j(x_i) \\
\eta_{\sigma_{1i}} &= b_3 + \sum_{j=1}^{k} w_{3j} B_j(x_i)
\end{align*}
$$

With $w = (w_{11}, \ldots, w_{1k}, w_{21}, \ldots, w_{kM})'$ and the penalty matrix $S_\lambda$

$$
S_\lambda = \begin{pmatrix}
\lambda_1 S & \ldots & 0 \\
\vdots & \ddots & \vdots \\
0 & \ldots & \lambda_S S
\end{pmatrix},
$$

a block diagonal matrix with the penalty matrix of the spline multiplied with the corresponding $\lambda_m$ on its diagonal, the penalized log-likelihood is

$$
l_{pen}(b, w) = \ln(L(b, w)) - \frac{1}{2} w' S_\lambda w
$$

In the following we will use splines with $k = 10$ knots. Bearing in mind that the parameters can and probably will originate from functions with varying degrees of smoothness, one could also use splines with different numbers of knots for the parameters. However, this is limited to the type of parameter. Since the components are interchangeable, $k$ cannot be selected component specific.
This simply means, that it is not possible to tell beforehand which component the model will identify as component 1 and which as component 2. Using some prior knowledge about the respective components, one could of course try to notch the model into a specific direction by initializing the network weights accordingly and thereby control which component it identifies as which at least to some extent, but that would introduce some unnecessary subjectivity into the model. For that reason, we simply use the same number of knots for every spline and let the smoothing parameter choice handle the task of suppressing unnecessary degrees of freedom, as is common in the context of non-parametric regression (Fahrmeir et al., 2013). This also brings the benefit of only having to create one single design matrix as seen in (6).

The smoothing parameter choice is done using a gridsearch where we choose the set of smoothing parameters with the lowest GAIC, a generalized version of the AIC (Akaike, 1998). Using criteria such as the AIC is common when dealing with hyperparameter choice concerning the allowed degrees of freedoms of splines in a comparable context (Mohanty and Fahnestock, 2021). For each spline, we consider 3 different smoothing parameters which are based on the effective degrees of the splines similar to Li and Cao (2023). The lowest possible smoothing parameter leaves the spline unpenalized, the highest of which leads to almost linear splines.

The GAIC is defined as $GAIC = -2 \cdot \hat{l} + 2 \cdot df$, $\hat{l}$ being the log-likelihood and $df$ the degrees of freedom of the model. The effective degrees of freedom of a single spline are defined as $edf = \text{tr}((Z'Z + \lambda S)^{-1}Z'Z)$ (Wood, 2017). For larger sample sizes, it might be necessary to use the BIC, defined as $BIC = -2\hat{l} + \log(n) \cdot df$ in order to penalize additional degrees of freedom more heavily.

Throughout, the networks are trained using the ADAM-optimizer introduced in Kingma and Ba (2014), an extension of the simple gradient-descent optimizer which creates individual learning rates for the parameters by not only using the gradient of the current training epoch but also estimates of its first and second moments based on gradients of the past training epochs. Due to its reputation of being a reliable optimizer for any kind of network architecture it is a popular and widely used default optimizer.

3 Simulations

The two different types of MDN, the conventional MDN used in (Hepp et al., 2022) and the Spline-MDN are now applied to simulated datasets in order to see whether the inclusion of splines actually improves the model. For the comparison we use the exact same network architecture as in Hepp et al. (2022).

3.1 Setup

For our simulation study, we consider two different simulation setups, one using a mixture of Gaussian distributions, the other one a mixture of gamma distributions. The Gaussian setup is the same as in Hepp et al. (2022) and consists of $M = 2$ components both with individual curves for $\mu_m$ and $\sigma_m$, and $\alpha_m$ as pictured in Figure 2. The gamma simulation setup is adapted from Hepp et al. (2020). Deviating from the setup in that paper, we have slightly changed the scale parameter of the second component, since in the original setup, both components share the same scale parameter. Furthermore, we used the same mixture weights $\alpha_m(x)$ as in our Gaussian setup, since the gamma setup in Hepp et al. (2020) uses constant mixture weights and we are interested in modelling all parameters dependent on $x$. Instead of the more common parametrization with shape parameter $\alpha$ and scale parameter $\beta$, we use the parametrization of the gamma distribution with

$$\alpha = \frac{1}{\sigma^2} \text{ and } \beta = \frac{1}{\mu \sigma^2},$$

since it makes the interpretation of the results and the comparison of the models more intuitive.

For both simulation setups, we generate datasets of two different sizes: $n = 5,000$ and $n = 10,000$ observations, with $x \sim U(0, 1)$. We draw 100 datasets from both mixture distributions and then apply the respective models to the datasets. As in Hepp et al. (2020) and Hepp et al. (2022), the performance of the models is evaluated using the integrated squared error (ISE) of the parameters of the lower component. The ISE is defined as

$$ISE(\hat{\theta}) = \int_0^1 (\theta(x) - \hat{\theta}(x))^2 \, dx,$$
Fig. 2 True parameters of the components of the mixture distribution consisting of two Gaussian distributions. The upper three panels show the means, standard deviations and mixture weights of the two components, the scatter plot is an exemplary sample from the mixture. The components are colour coded, such that curves of the same color belong to the same component.

θ being the respective parameter of interest. For the comparison we consider the mixture weight α₁, the mean μ₁, the scale parameter σ₁ as well as the 95%-quantile Q_{0.95,1} of the lower component.

3.2 Results

As illustrated in Figure 3, the model using penalized regression splines does in fact appear to solve the component crossing issue observed with the conventional MDN. As far as the lower component is concerned, the spline model seems to estimate the parameters much more accurate than the MDN. For both models, the estimated curves for the upper component spread much more than the estimated curves for the lower component. This is due to the fact that the variance of the upper component is higher than the variance of the lower component at any point and furthermore, the lower component is the dominant component in the sample, meaning the relative sample size of the second component is also smaller.

Despite the Spline-MDN seemingly performing much better, there are some issues to be addressed. Looking at the average prediction errors over the range of x in Figure 4 for models trained on datasets with n = 10,000, it becomes apparent that for the most part, the prediction errors for the parameters are lower than for the MDN. However, it also seems that the spline MDN might be somewhat biased. The mixture weight for the first component, for instance, seems to be underestimated systematically. For σ₁ we see an almost linear trend in the average prediction error, μ₁ seems to be underestimated for high values of x.

In Table 1, we compare the average integrated squared errors of the estimated parameters. Due to the fact that the issue of component switching obviously would raise the average ISEs of the conventional MDN above those of the spline models, the runs in which it occurred are excluded. For the Gaussian setting, the Spline-MDN performs much better for all parameters with the ISEs being much lower throughout. In the gamma setting, the ISEs of the Spline-MDN are much lower for all parameters except the mean, which is slightly higher for n = 5,000 and only slightly lower than the ISE of the conventional MDN for n = 10,000. Nevertheless, given that the other parameters seem to be estimated much more accurately, the spline model still seems to estimate the true distribution...
Fig. 3 Estimated distribution parameters and mixture weights for both components. The upper three plots show the results using the conventional MDN, the lower plots show the results using the Spline-MDN. The grey curves depict models that switch up the components.

Fig. 4 Average prediction errors ($\hat{\theta}(x) - \theta(x)$) for the mixture weight, mean, standard deviation and 95% Quantile of the blue component over the range of $x$ for the Gaussian setup with $n = 10,000$. The prediction errors of the conventional MDN are plotted in orange, for the Spline-MDN in blue.
Table 1 Average integrated squared errors of conventional MDN and Spline-MDN for Gaussian and gamma setups

<table>
<thead>
<tr>
<th>n</th>
<th>Average ISE($\hat{\alpha}_1$)</th>
<th>Average ISE($\hat{\mu}_1$)</th>
<th>Average ISE($\hat{\sigma}_1$)</th>
<th>Average ISE($\hat{Q}_{0.95,1}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>conventional MDN</td>
<td>Spline MDN</td>
<td>conventional MDN</td>
<td>Spline MDN</td>
</tr>
<tr>
<td>5,000</td>
<td>0.0092</td>
<td>0.0063</td>
<td>1.9421</td>
<td>0.8134</td>
</tr>
<tr>
<td>10,000</td>
<td>0.0064</td>
<td>0.0038</td>
<td>1.2055</td>
<td>0.5255</td>
</tr>
<tr>
<td></td>
<td>conventional MDN</td>
<td>Spline MDN</td>
<td>conventional MDN</td>
<td>Spline MDN</td>
</tr>
<tr>
<td>5,000</td>
<td>0.0033</td>
<td>0.0022</td>
<td>0.0185</td>
<td>0.0191</td>
</tr>
<tr>
<td>10,000</td>
<td>0.0018</td>
<td>0.0012</td>
<td>0.0133</td>
<td>0.0132</td>
</tr>
</tbody>
</table>

much more reliably as indicated by the ISEs of the 95%-quantile.

The overall takeaway from this is that our approach does not only solve the component switching issue, but also clearly outperforms the conventional MDN model, even when ignoring the runs in which it fails completely.

4 Application to Data of Hemoglobin Levels

Having shown that the revised MDN outperforms the conventional MDN, we now apply our model to a dataset consisting of $n = 60,423$ hemoglobin concentration values of female patients between 1 and 18 years old, which is part of the PEDREF reference interval initiative (PED). Given that this dataset covers the whole range of child development from infant to young adult, this dataset especially necessitates the use of a model that treats the distribution parameters as functions of the covariates and is therefore a prime example of a use case for MDNs.

For both networks, we model the data as originating from a mixture of normal distributions and specify the respective network architectures the same way as in the simulation study, hence the amount of knots and hidden layers, the activation functions and optimizers remain the same. Deviating from the simulation study, we switched from the GAIC to the BIC. This is simply due to the fact that we wanted a model choice criterion that takes the increased sample size into account.

As we cannot assess the performance of the networks regarding the true parameters as we did before, we test the stability of both models by running each model 100 times with randomly initialized weights for each run. The results still seem to be in line with the results from the simulation study and Hepp et al. (2022). As indicated in Figure 5, the conventional MDN still appears to have an issue with component switching, albeit much less pronounced than in the
Fig. 5 Estimated distributions of Hemoglobin concentrations. The upper three panels show the results using the conventional MDN, the lower three panels the results using our model. For the conventional MDN, component crossing happens on fewer occasions than in the simulation study. The grey curves represent models that switch up the components. The Spline-MDN is remarkably stable. Curves of the same color belong to the same class, so contrary to the simulation setup, the upper mean belongs to the lower standard deviation and vice versa.

Fig. 6 Standard deviations of the estimated 5%-quantiles over the range of age in years. The orange line depicts the standard errors of the original MDN architecture, the blue line those of the revised network architecture.

Simulation study, where it occurred in about half the runs. Here, it only occurred in 6 out of our 100 runs. This is a vast improvement compared to the simulation study and can probably attributed to the larger dataset. Nevertheless the conventional MDN still proves to be much less stable than the Spline-MDN.

In contrast, the Spline-MDN is remarkably stable. For 99 out of 100 runs, the same set of smoothing parameters is chosen, in the other run, we see that the estimated mean of the pathological component remains unpenalized. As can be seen in Figure 5, given that the same set of smoothing parameters is chosen, the network converges to very close solutions. The one case in which a different set of smoothing parameters is chosen also only seems to concern the pathological component.

This improved stability becomes even clearer in Figure 6, which compares the standard deviations of the estimated 5%-quantile for both models. While not perfectly stable, the standard deviation of the spline deviation is pretty close to zero throughout, especially compared to the conventional MDN.

Overall, both models seem to estimate the respective location and shape parameters of both
components quite similarly, although the Spline-MDN allows more flexibility for the means. Furthermore, the mean of the pathological component seems to slightly increase for higher ages, whereas the curves seem to point in the opposite direction for the conventional MDN. It is to be noted, that the conventional MDN becomes much more unstable in this area.

Given that especially the healthy component is estimated quite similarly, it is not surprising that the estimated average 5%-quantiles plotted in Figure 7 are also quite similar for both models. The advantage of our method becomes very clear when looking at the standard deviations illustrated by the shaded areas. Even for the higher values of age, where the standard deviation of the estimated quantiles is at its highest level, the shaded area is barely visible.

The application to the hemoglobin dataset confirms the advantages of our new approach over the approach used in Hepp et al. (2022). In addition to the obvious advantage of eliminating the component switching issue, our model proves to be much more stable. This improved stability is especially important for models which are to be used for medical decision making.

5 Conclusion

In this work, we extended an existing neural network architecture using penalized regression splines. For our settings, we showed our model is not only able solve the major issue that MDNs appear to have, but is also able to outperform the conventional network architecture. Borrowing from the GAM and GAMLSS frameworks, we created models that are straightforward as far as interpretability is concerned while making use of the numerical tools from the field of deep learning. Since we are using stochastic gradient descent with mini batches, our approach is also easily scalable.

The usefulness of our approach becomes especially apparent in our data application, since we created a model that could make the tedious process of designing studies and collecting data for the identification of reference intervals unnecessary. With the massive amount of data that is accumulated in laboratories, our approach could prove to be a reliable tool for the estimation of appropriate reference intervals.

While already showing great promise, our approach still offers plenty of room for extensions. Since our approach is clearly rooted in GAMLSS, the inclusion of multiple covariates is an obvious next step. Going back to our data example, the assumption that hemoglobin levels depend on age only is extremely naive and restrictive, especially considering that laboratory databases provide plenty of usable information. While the inclusion of multiple covariates is of course also possible and common in conventional MDNs, our model has the big advantage of providing interpretable covariate effects, something that is notoriously difficult to achieve in conventional neural networks (Agarwal et al., 2021).

To improve our model and to get even more stable results, one could also try to incorporate the smoothing parameter choice into the training
process rather than treating it as a hyperparameter. In general, the smoothing parameter choice is not an issue that is exclusive to our approach. Latent class distributional regression (Hepp et al., 2020), which uses B-Splines for the estimation of the parameters, simply circumvents the smoothing parameter choice by simply fixing the degrees of freedoms of the respective splines, the MDN approach in Hepp et al. (2022) basically fixes the allowed flexibility using the number of hidden neurons.

6 Acknowledgments

This work was supported by the Freigeist-Fellowships of Volkswagen Stiftung, project “Bayesian Boosting - A new approach to data science, unifying two statistical philosophies”. Quentin E. Seifert performed the present work in partial fulfilment of the requirements for obtaining the degree “Dr. rer. pol.” at the Georg-August-Universität Göttingen. Funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) within project 450330162 is gratefully acknowledged.

References


