Rügamer, David; Pfisterer, Florian; Bischl, Bernd; Grün, Bettina

Mixture of experts distributional regression: implementation using robust estimation with adaptive first-order methods

AStA Advances in Statistical Analysis

Provided in Cooperation with:
Springer Nature

Suggested Citation: Rügamer, David; Pfisterer, Florian; Bischl, Bernd; Grün, Bettina (2023) : Mixture of experts distributional regression: implementation using robust estimation with adaptive first-order methods, AStA Advances in Statistical Analysis, ISSN 1863-818X, Springer, Berlin, Heidelberg, Vol. 108, Iss. 2, pp. 351-373, https://doi.org/10.1007/s10182-023-00486-8

AStA Advances in Statistical Analysis (2024) 108:351–373
https://doi.org/10.1007/s10182-023-00486-8

ORIGINAL PAPER

Mixture of experts distributional regression: implementation using robust estimation with adaptive first-order methods

David Rügamer[1,2,3] · **Florian Pfisterer**[1,3] · **Bernd Bischl**[1,3] · **Bettina Grün**[4]

Received: 17 November 2022 / Accepted: 24 October 2023 / Published online: 15 November 2023

Abstract

In this work, we propose an efficient implementation of mixtures of experts distributional regression models which exploits robust estimation by using stochastic first-order optimization techniques with adaptive learning rate schedulers. We take advantage of the flexibility and scalability of neural network software and implement the proposed framework in *mixdistreg*, an R software package that allows for the definition of mixtures of many different families, estimation in high-dimensional and large sample size settings and robust optimization based on TensorFlow. Numerical experiments with simulated and real-world data applications show that optimization is as reliable as estimation via classical approaches in many different settings and that results may be obtained for complicated scenarios where classical approaches consistently fail.

Keywords Mixture models · Deep learning · Structured additive regression · Neural networks

✉ David Rügamer
david.ruegamer@stat.uni-muenchen.de

Florian Pfisterer
florian.pfisterer@stat.uni-muenchen.de

Bernd Bischl
bernd.bischl@stat.uni-muenchen.de

Bettina Grün
bettina.gruen@wu.ac.at

[1] Department of Statistics, LMU Munich, Munich, Germany

[2] Department of Statistics, TU Dortmund, Dortmund, Germany

[3] Munich Center for Machine Learning, Munich, Germany

[4] Institute for Statistics and Mathematics, WU Vienna, Vienna, Austria

1 Introduction

Mixture models are a common choice to model the joint distribution of several subpopulations or subclasses on the basis of data from the pooled population where the subclass memberships are not observed (for an introduction see McLachlan and Peel 2004). Each subclass is assumed to follow a parametric probability distribution such that the pooled observations are from a mixture of these distributions. Many applications of mixture models aim at estimating the distributions of the latent subclasses and identifying subclass memberships of the observations (McLachlan et al. 2019). Mixture models have also been used in machine learning, e.g., for clustering (Viroli and McLachlan 2019), to build generative models for images (Van den Oord and Schrauwen 2014) or as a hybrid approach for unsupervised outlier detection (Zong et al. 2018).

Different kinds of mixture models are used depending on the available data structure and the parametric model which is assumed for each subpopulation or subclass. In the following, we assume a supervised learning task and that regression models are used to model the subpopulations. This leads to mixture regression models which define a mixture of (conditional) models for the outcome of interest, where the mixture components are still unknown and thus considered latent variables. This model class is also referred to as mixtures of experts (Gormley and Frühwirth-Schnatter 2019). Mixtures of experts allow for the inclusion of covariates when modeling the mixture components as well as for the mixture weights. In the following, we consider a mixture of experts model where the regression models do not only include the mean parameter but also other distributional parameters that may depend on the covariates or features. In addition, we assume that the subpopulation sizes vary with covariates (or features). This leads to the class of mixture of experts distributional regression models.

1.1 Mixture models and their estimation

As for classical statistical regression models, the goal of *mixtures of regressions* or *mixture regression models* is to describe the conditional distribution of a response (or outcome), conditional on a set of covariates (or features). Mixtures of regressions have been first introduced by Quandt (1958) under the term *switching regimes* where only two-component mixtures of linear regression models were considered, i.e., the number of subclasses was fixed to two. This was extended to general mixtures of linear regression models by DeSarbo and Cron (1988) who referred to this approach as a model-based version of *clusterwise regression* (Späth 1979). The extension to mixtures of generalized linear regression models was proposed in Wedel and DeSarbo (1995). Aiming for a setting beyond mean regression (Kneib 2013), a distributional regression setting can be used such as generalized additive models for location, scale and shape (GAMLSS; Rigby and Stasinopoulos 2005) which also allow for nonlinear smooth relationships between covariates and the distributional parameter of interest (Stasinopoulos et al. 2018).

Mixture models can be estimated using various techniques, with the expectation–maximization (EM) algorithm based on the maximum-likelihood principle being the most prominent one. Other approaches include Bayesian methods such as MCMC algorithms with data augmentation (Diebolt and Robert 1994). An alternative way of model specification and estimation was proposed by Bishop (1994) who introduced mixture density networks (MDNs). MDNs use a mixture of experts distributional regression model specification. But the relationship between the covariates (or features) and the mixture weights or the distributional regression models is learned using a neural network. The training of MDNs is done using highly optimized stochastic gradient descent (SGD) routines with adaptive learning rates and momentum. These procedures have proven to be very effective in the optimization of large neural networks with millions of parameters and complex model structures. While often MDNs are advantageous in terms of prediction performance, they lack interpretability as the relationship between inputs (features) and distribution parameters is modeled by a deep neural network.

1.2 Our contribution

1.2.1 Novel modeling approach

In this work, we combine the ideas of interpretable mixtures of regression models and MDNs to allow for a mixture of experts distributional regression models in a very general setup. In particular, this approach enables modeling mixtures of subpopulations where the distribution of every subpopulation is modeled using a distributional regression. Predictors of every distribution parameter in every subpopulation can be defined by linear effects or (tensor product) splines and thereby allow not only for complex relationships between the features and the distributions' mean but also for other distribution characteristics such as scale or skewness. Furthermore, the subpopulation sizes may vary with features in a flexible way using again a combination of linear effects as well as (tensor product) splines.

1.2.2 Robust estimation

Estimating mixture regression models within a maximum-likelihood framework on the basis of the EM algorithm works well for smaller problems. However, higher-dimensional settings where the number of parameters is similar to or exceeds the number of observations are often infeasible to estimate. In fact, the convergence and stability of classical algorithms are more and more adversely affected by an increase in model complexity. Maximum-likelihood estimation of distributional regression models itself induces a non-convex optimization problem in all but special cases. Thus, extending mixtures of mean regression models to mixtures of distributional regression models further complicates the optimization.

In this work, inspired by MDNs, we suggest and analyze the usage of stochastic first-order optimization techniques using adaptive learning rate schedulers for (regularized) maximum-likelihood estimation. Our results show that this

approach is as reliable as estimation via classical approaches in many different settings and even provides estimation results in complex cases where classical approaches consistently fail.

1.2.3 Flexible and scalable implementation

Common implementations of EM optimization routines are limited in their flexibility to specify a mixture of (many) potentially different distributions but in general focus on the case where all components have a parametric distribution from the same distributional family. Another (computational) limitation of existing approaches is faced for large amounts of data (observations) as methods usually scale at least quadratic in the number of observations. On the other hand, SGD optimizers from the field of deep learning are trained on mini-batches of data allowing large dataset applications and the use in a generic fashion for all model classes. We take advantage of this flexibility and scalability and implement the fitting of the proposed model class using *mixdistreg*, an R (R Core Team 2022) software package based on the R package *deepregression* (Rügamer et al. 2023) that allows for the definition of mixtures with components from many different distributional families, estimation in high-dimensional and large sample size settings and robust optimization based on the deep learning platform TensorFlow (Abadi et al. 2015). In order to use TensorFlow within R, the package *reticulate* (Ushey et al. 2022) is used to connect R to Python (Van Rossum and Drake Jr 1995).

1.2.4 Summary of our approach and overview on the paper structure

Our framework unites neural density networks (Magdon-Ismail and Atiya 1998) with (distributional) regression approaches, extends MDNs by incorporating penalized smooth effects and comprises various frameworks proposed in the statistical community such as Leisch (2004); Grün and Leisch (2007); Stasinopoulos and Rigby (2007) to estimate mixtures of linear, generalized linear, generalized additive or distributional regression models. In contrast to many existing approaches, our framework further allows to estimate the mixture weights themselves on the basis of an additive structured predictor. We refer to this combination of the model class of mixtures of experts distributional regression with structured additive predictors and the estimation using neural network software as the *neural mixture of experts distributional regression* (NMDR) approach.

The remainder of this paper is structured as follows. In Sect. 2, we present our model definition. In Sect. 3, we introduce the architecture for SGD-based optimization in neural networks and discuss penalized estimation approaches including mixture sparsification. We then demonstrate the framework's properties using extensive numerical experiments in Sect. 4 and its application to real-world data in Sect. 5. We conclude with a discussion in Sect. 6.

2 Methodology

Our goal is to model the conditional distribution of $Y \mid \boldsymbol{x}$ where Y is the univariate outcome (or response) of interest. We assume that $Y \mid \boldsymbol{x} \sim \mathcal{F}$, where $\mathcal{F}$ is a mixture of parametric distributions $\mathcal{F}_m, m \in \{1, \ldots, M\} =: \mathcal{M}$. These distributions, in turn, depend on unknown parameters $\boldsymbol{\theta}_m(\boldsymbol{x}) \in \mathbb{R}^{k_m}$ that are influenced by a set of features (or covariates) $\boldsymbol{x} \in \mathbb{R}^p$. Furthermore, the nonnegative mixture weights are also allowed to depend on features (or covariates) $\boldsymbol{x} \in \mathbb{R}^p$ such that $\pi_m(\boldsymbol{x})$ for all $m = 1, \ldots, M$ with $\sum_{m=1}^{M} \pi_m(\boldsymbol{x}) = 1$.

2.1 Model definition

For an observation y in (a suitable subset of) $\mathbb{R}$ constituting the support from the conditional distribution of $Y \mid \boldsymbol{x}$, we define the conditional density by

$$f_{Y|\boldsymbol{x}}(y \mid \boldsymbol{\vartheta}(\boldsymbol{x})) = \sum_{m=1}^{M} \pi_m(\boldsymbol{x}) f_m(y \mid \boldsymbol{\theta}_m(\boldsymbol{x})), \tag{1}$$

i.e., by a mixture of density functions f_m of the distributions $\mathcal{F}_m$. Each component density f_m has its own k_m distribution parameters $\boldsymbol{\theta}_m = (\theta_{m,1}, \ldots, \theta_{m,k_m})^\top$. $\pi_m \in [0, 1]$ are the mixture weights with $\sum_{m=1}^{M} \pi_m = 1$. The vector $\boldsymbol{\vartheta} = (\boldsymbol{\theta}^\top, \boldsymbol{\pi}^\top)^\top \in \mathbb{R}^{K+M}$ with $K = \sum_{m=1}^{M} k_m$ comprises all distribution parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M)^\top$ and all mixture weights $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_M)^\top$.

Each of the parameters ϑ_j in $\boldsymbol{\vartheta}$ is assumed to depend on the features $\boldsymbol{x}$ through an additive predictor $\eta_j, j = 1, \ldots, K + M$. For the distribution parameters $\boldsymbol{\theta}$, a monotonic and differentiable function $h_j, j = 1, \ldots, K$, is assumed to provide a map between the additive structured predictor and the distribution parameter, i.e., $\vartheta_j = h_j(\eta_j(\boldsymbol{x}))$. The parameter-free transformation function h_j ensures the correct domain of each ϑ_j. For example, if ϑ_j represents a scale parameter the function, h_j ensures that ϑ_j is positive while the additive predictor η_j may take arbitrary values in $\mathbb{R}$.

For the mixture weights, $\boldsymbol{\pi}$ a single monotonic and differentiable function h_{K+1} is assumed to map the M additive predictors to the $(M - 1)$-dimensional simplex, i.e., h_{K+1} maps $\mathbb{R}^M \to [0, 1]^M$ under the condition that the sum of the weights is 1. This links the last M additive predictors $\boldsymbol{\eta}_\pi := (\eta_{K+1}, \ldots, \eta_{K+M})^\top$ to the set of mixture weights $\boldsymbol{\pi}$. The most common choice in this respect for h_{K+1} is the softmax function

$$h_{K+1}(\boldsymbol{\eta}_\pi) = (\text{softmax}_1(\boldsymbol{\eta}_\pi), \ldots, \text{softmax}_M(\boldsymbol{\eta}_\pi))$$

with

$$\text{softmax}_j(\boldsymbol{\eta}) = \frac{\exp(\eta_j)}{\sum_{l=1}^{M} \exp(\eta_l)}.$$

This implies

$$\pi_m(\boldsymbol{x}_i) = \frac{\exp(\eta_{K+m}(\boldsymbol{x}_i))}{\sum_{l=1}^{M} \exp(\eta_{K+l}(\boldsymbol{x}_i))}, \quad \text{for} \quad m \in \mathcal{M}. \tag{2}$$

Effects in the additive predictors $\eta_{K+l}, l = 1, \ldots, M$, in (2) are not identifiable without further constraints. We do not enforce any constraints during model training. Model interpretation is still possible in relative terms (e.g., using a log-odds interpretation). However, some constraints would need to be imposed if identifiable regression coefficients for the additive predictors are to be obtained.

For all parameters in $\boldsymbol{\vartheta}$, the additive structured predictors $\eta_j(\boldsymbol{x})$ ensure interpretability of the relationship between the parameter ϑ_j and the covariates. For example, if η_{K+M} is a linear model, i.e., $\eta_{K+M}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\beta}$, the regression coefficients $\boldsymbol{\beta}$ can be interpreted as linear contributions of each of the features to the logits of the mixture weight for the last mixture component M.

2.2 Additive predictor structure

The model (1) relates all model parameters $\boldsymbol{\vartheta}$ to features $\boldsymbol{x}$ through additive predictors $\eta_j, j = 1, \ldots, K + M$. As different densities f_m and also different parameters $\boldsymbol{\theta}_m$ have potentially different influences on the conditional distribution of $Y \mid \boldsymbol{x}$, every parameter in $\boldsymbol{\theta}$ is defined by its own additive structured predictor η_j. Here we assume the additive predictors to have the following structure:

$$\eta_j = \beta_{0,j} + \boldsymbol{x}_{\mathcal{L}(j)}^\top \boldsymbol{\beta}_j + \sum_{l \in \mathcal{S}(j)} \phi_{l,j}(\boldsymbol{x}), \tag{3}$$

where $\beta_{0,j}$ corresponds to the model intercept, $\boldsymbol{\beta}_j$ are the linear effects for pre-defined covariates $\boldsymbol{x}_{\mathcal{L}(j)}$ with $\mathcal{L}(j) \subseteq \{1, \ldots, p\} \cup \emptyset$ being a subset of all possible predictors and $\phi_{l,j}$ are nonlinear smooth functions for one or more covariates in $\boldsymbol{x}$ with $\mathcal{S}(j)$ being a (potentially empty) set of indices indicating the covariates with nonlinear predictor effects. We assume that every $\phi_{l,j}(\boldsymbol{x})$ can be represented by (tensor product) basis functions $B_{l,j,o}, o = 1, \ldots, O$, taking one or several columns of $\boldsymbol{x}$ as input and mapping these onto the space spanned by the basis functions. Denoting $\boldsymbol{z}_{l,j} := \boldsymbol{B}_{l,j}(\boldsymbol{x}) = (B_{l,j,1}(\boldsymbol{x}), \ldots, B_{l,j,O}(\boldsymbol{x}))^\top \in \mathbb{R}^O$, the nonlinear smooth terms can be written as $\phi_{l,j}(\boldsymbol{x}) = \boldsymbol{z}_{l,j}^\top \boldsymbol{\gamma}_{l,j}$ where $\boldsymbol{\gamma}_{l,j} \in \mathbb{R}^O$ are the corresponding basis coefficients for $\boldsymbol{z}_{l,j}$.

In principle, such a flexible specification may also be used for $\eta_{K+1}, \ldots, \eta_{K+M}$, i.e., for the additive structured predictors related to $\boldsymbol{\pi}$. While this is technically possible (using a different subnetwork for every π_m), these predictors are usually assumed to share one and the same additive structure, i.e., $\boldsymbol{x}_{\mathcal{L}(j)}$ and $\boldsymbol{z}_{l,j}$ with $l \in \mathcal{S}(j)$ are the same for all j related to the M mixture weights. This further enables a straightforward interpretation of the predictor–mixture weight relationship, as sharing one additive predictor across all π_ms resembles a multinomial logistic regression model. We will thus assume the same specification for all these predictors. We summarize all

model coefficients of the linear terms by $\boldsymbol{\beta} = (\beta_{0,1}, \boldsymbol{\beta}_1^\top, \beta_{0,2}, \boldsymbol{\beta}_2^\top, \ldots, \beta_{0,K+M}, \boldsymbol{\beta}_{K+M}^\top)^\top$ and all model coefficients for representing the nonlinear smooth functions by $\boldsymbol{\gamma} = (\gamma_{0,1}, \boldsymbol{\gamma}_1^\top, \gamma_{0,2}, \boldsymbol{\gamma}_2^\top, \ldots, \gamma_{0,K+M}, \boldsymbol{\gamma}_{K+M}^\top)^\top$. In the following, we summarize these two parameter vectors as $\boldsymbol{\psi} = (\boldsymbol{\beta}^\top, \boldsymbol{\gamma}^\top)^\top$.

2.3 Model log likelihood

Based on model (1) and the structures imposed on the predictors in (3), the negative log likelihood of the model parameters $\boldsymbol{\psi}$ for n independent observations $\boldsymbol{y} = (y_1, \ldots, y_n)^\top \in \mathbb{R}^n$ and their corresponding n observed feature vectors $\boldsymbol{x} := (\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_n^\top)^\top \in \mathbb{R}^{n \times p}$ is given by the sum of the negative log-likelihood contributions $\ell_i(\boldsymbol{\psi})$ of each observation $i = 1, \ldots, n$:

$$\ell(\boldsymbol{\psi}) := \sum_{i=1}^{n} \ell_i(\boldsymbol{\psi}) = -\sum_{i=1}^{n} \log \left\{ \sum_{m=1}^{M} \pi_m(\boldsymbol{x}_i) f_m(y_i \mid \boldsymbol{\theta}_m(\boldsymbol{x}_i)) \right\}. \tag{4}$$

The negative log likelihood can be rewritten as the negative sum of the exponentiated log likelihoods of both the mixture weights π_m and the component densities f_m using the log-sum-exp (LSE) function:

$$-\sum_{i=1}^{n} \log \left\{ \sum_{m=1}^{M} \exp \left[\log \pi_m(\boldsymbol{x}_i) + \log f_m(y_i \mid \boldsymbol{\theta}_m(\boldsymbol{x}_i)) \right] \right\}. \tag{5}$$

In practice, formulation (5) is often preferred over (4) as it is less affected by under- or overflow problems.

2.4 Identifiability

Identifiability is of concern for the mixture of experts distributional regression model because of the identifiability problems which could occur due to the mixture specification, due to the additive structured predictors and due to the softmax mapping to the mixture weights.

2.4.1 Mixture models

For any mixture model, trivial identifiability problems may arise due to label switching and overfitting with either empty or duplicated components. In addition, also generic identifiability problems may occur for mixtures of distributions but also for mixtures of regressions. A detailed overview of identifiability problems in the finite mixture case is given in Frühwirth-Schnatter (2006).

2.4.2 Additive structured predictors

The components of the additive model with predictor structures of the form (3) are in general only identifiable up to a constant and also require further restrictions if both linear and nonlinear smooth effects are defined for one and the same covariate. For example, $\eta_j = \beta_{0,j} + \phi_{1,j}(x_1)$ can be equally represented by $\tilde{\beta}_{0,j} + \tilde{\phi}_{1,j}(x_1)$ by defining $\tilde{\beta}_{0,j} = \beta_{0,j} + c$ and $\tilde{\phi}_{1,j}(x_1) = \phi_{1,j}(x_1) - c$ with $c \in \mathbb{R}$, i.e., by adding and subtracting a constant c in both terms. We ensure the identifiability of nonlinear smooth terms $\phi_{l,j}$ by using sum-to-zero constraints for all nonlinear additive functions such as splines or tensor product splines. The identifiability of linear effects $x\beta$ in the presence of a univariate nonlinear effect $\phi(x)$ must also be ensured. In this case, several different options exist (see, e.g., Rügamer et al. 2023b). The most straightforward way is to use a nonlinear effect $\phi(x)$ with a basis representation that includes the linear effect as null space (and hence, for enough penalization as discussed in Sect. 3.3, results in a linear effect). In this case, $\mathcal{L}(j)$ only consists of variables that are only modeled using a linear component (e.g., categorical effects) and $\mathcal{L}(j) \cap \mathcal{S}(j) = \emptyset$.

3 Model representation and robust estimation

The observed data log likelihood is, in general, not concave and thus difficult to optimize. We suggest to use optimizers from the field of deep learning by framing our model as a neural network. This enables the fitting of the full model class of mixture of experts distributional regression models with additive structure predictors in a straightforward way. Numerical experiments confirm that this choice—when properly trained—is not only more flexible and robust than EM-based optimization but also makes large dataset applications feasible due to mini-batch training.

3.1 Neural network representation

Models described in (1) and (3) can be represented as neural networks in the following way. An exemplary architecture is depicted in Fig. 1. The network architecture implementing model (1) defines at most K subnetworks, where each subnetwork models one or more additive structured predictors η_j of a distribution parameter θ_j. Using an appropriate parameter-free transformation function, these additive structured predictors are mapped to the distribution parameters and passed to a distribution layer (Dillon et al. 2017). Each distribution layer corresponds to a mixture component f_m that is further passed to a multinomial or categorical distribution layer, modeling the mixture of all defined distributions. The mixture weights can either be directly estimated or also learned on the basis of input features using additive structured predictors in another subnetwork. A classical linear mixture regression combining M linear regressions would, e.g., be given by M subnetworks, each learning the expectation of a normal distribution and a mixture subnetwork that only takes a constant input (a bias) and learns the M mixture weights. The individual additive predictors for each subnetwork are, in most cases, fully connected layers with only

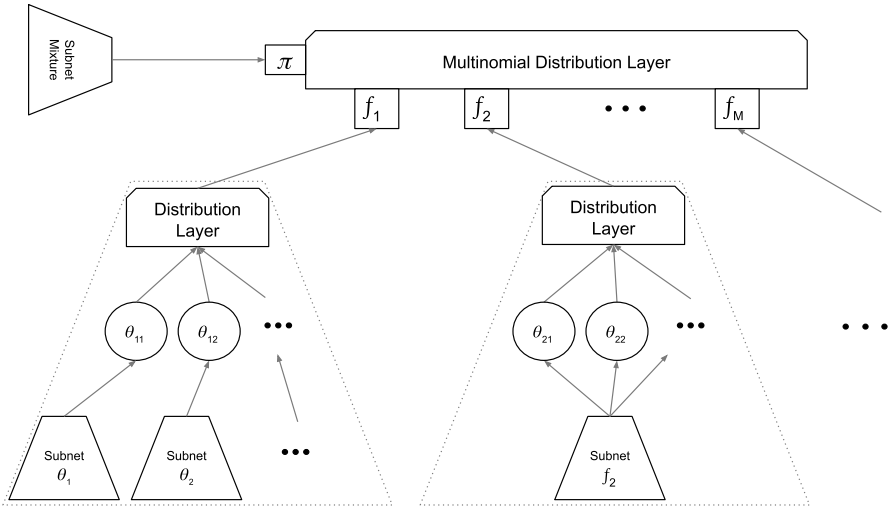

Fig. 1 Example of an architecture. Smaller subnetworks (subnet) learn one or more parameters of a distribution which is defined in the respective distribution layer. For the first distribution in this example, each distribution parameter in $\boldsymbol{\theta}_1$ is learned through a separate network while the second distribution is learned by a network that outputs all parameters $\boldsymbol{\theta}_2$ together (e.g., used when $\boldsymbol{\theta}_2$ are constrained parameters such as probabilities). Each distribution layer thus corresponds to a distributional regression model. The mixture model is then defined by an additional subnetwork that learns the mixture weights $\boldsymbol{\pi}$ as well as by the M learned distributions $f_1, \ldots, f_K$

one neuron. Such a layer describes a weighted linear combination of inputs $\boldsymbol{x}$ with weights (here the regression coefficients $\boldsymbol{\beta}$). The same representation can also be used for basis function evaluations $\boldsymbol{z}$ for nonlinear functions $\boldsymbol{\phi}$. For more details, we refer to Rügamer et al. (2023). Due to the unifying network structure of our approach, the use of other layers or deep neural networks as subnetworks is also possible, but not further discussed in this article (see Rügamer et al. 2023b, for details).

3.2 Optimization routines

Representing the model (1) and (3) as a neural network makes a plethora of optimization routines from deep learning readily available for model estimation. Various first-order stochastic optimization routines exist for neural networks. Most of these optimizers work with mini-batches $J_1, \ldots, J_B \subset \{1, \ldots, n\}$ of data and hence perform stochastic gradient descent (SGD). The update of parameters $\boldsymbol{\psi}^{[t]}$ in iteration $t \in \mathbb{N}$ and mini-batch J_b is

$$\boldsymbol{\psi}^{[t]} = \boldsymbol{\psi}^{[t-1]} - \nu^{[t]} \frac{1}{|J_b|} \sum_{i \in J_b} \nabla_{\boldsymbol{\psi}} \ell_i(\boldsymbol{\psi}^{[t-1]}), \tag{6}$$

where $\boldsymbol{\psi}^{[0]}$ is some starting value, $\nabla_{\boldsymbol{\psi}} \ell_i(\boldsymbol{\psi}^{[t-1]})$ are the individual gradients of the negative log-likelihood contributions of the observations in the batch evaluated at the parameters of the previous iteration and $\nu^{[t]}$ is a learning rate updated using a learning

rate scheduler. We determine the starting value $\boldsymbol{\psi}^{[0]}$ with the Xavier uniform initialization scheme (Glorot and Bengio 2010). For the update of $\nu^{[t]}$, we found the following learning rate schedulers useful for the optimization of mixture of experts distributional regression models: *RMSprop* (Hinton et al. 2012), *Adadelta* (Zeiler 2012), *Adam* (Kingma and Ba 2014) and *Ranger* (Wright 2019). These optimizers in turn often come with hyperparameters such as their initial learning rate which need to be adjusted. We will investigate their influence in our numerical experiments section.

3.3 Penalized estimation

Estimating the model class of mixtures of experts distributional regressions with additive structured predictors can benefit from a penalized log-likelihood specification. Such a penalization allows to control the smoothness or wiggliness of the nonlinear smooth functions in the additive structured predictors and to induce sparsity in the estimation of the additive structured predictors as well as the mixture weights.

3.3.1 Additive structured predictors

In order to estimate suitable nonlinear smooth additive structured predictors based on splines within the neural network, the respective coefficients in each layer may require regularization using appropriate penalties or penalty matrices. The smoothness or wiggliness of the nonlinear additive structured predictor can be calibrated by either selecting a suitable dimension of the basis representation or imposing a penalization on the respective coefficients in combination with a generous basis representation. Using the later approach, the penalized version of (4) is given by

$$\ell_{pen}(\boldsymbol{\psi}) = \ell(\boldsymbol{\psi}) + \sum_{l \in \Lambda} \lambda_l \boldsymbol{\gamma}_l^\top \boldsymbol{P}_l \boldsymbol{\gamma}_l, \tag{7}$$

with sets of index sets Λ defining the weights that are penalized using a quadratic penalty with individual smoothing parameters λ_l and individual quadratic penalty matrices $\boldsymbol{P}_l$ (see, e.g., Wood 2017). While estimating the tuning parameters is possible by including a more elaborated optimization routine in the neural network such as the Fellner–Schall method (Wood and Fasiolo 2017), we use the approach suggested by Rügamer et al. (2023b) to tune the different smooth effects by relating λ_l to their respective degrees of freedom df_l. This has the advantage of training the network in a simple backpropagation procedure with only little to no tuning, by, e.g., setting the df_l equal to a pre-specified value for all $l \in \Lambda$. More specifically, while setting the same λ_l value for all smooth terms would result in (very) different penalization strengths for every term, choosing their λ_l value through one shared df value, i.e., $\text{df}_l \equiv \text{df}$, yields equally penalized smooth terms for all $l \in \Lambda$. Setting this shared df value to a moderate number, e.g., $\text{df} = 9$, proved to be a well-working prior assumption for each smooth term and circumvents expensive tuning schemes by fixing all λ_l values a priori.

3.3.2 Entropy-based penalization of mixtures

In order to penalize an excessive amount of nonzero mixture weights, we further introduce an entropy-based penalty for the mixture weights that can be simply added to the objective function in (7):

$$\ell_{ent}(\boldsymbol{\psi}) = \ell_{pen}(\boldsymbol{\psi}) - \xi \sum_m \pi_m \log \pi_m. \tag{8}$$

The second part of (8) is controlled by a tuning parameter $\xi \in \mathbb{R}_0^+$ and corresponds to the entropy induced by the (estimated) marginal mixture weights $\boldsymbol{\pi}$, i.e., in case the mixture weights depend on covariates $\boldsymbol{x}$ the mixture weights obtained when averaging over the observed $\boldsymbol{x}$. A large value of ξ enforces the weight distribution to be sparse in the amount of nonzero elements in $\boldsymbol{\pi}$, while smaller values will result in an (almost) uniform distribution of $\boldsymbol{\pi}$. As the entropy is permutation-invariant w.r.t. the ordering of the components in $\boldsymbol{\pi}$, this penalty is particularly suitable for mixtures that are only identified up to a permutation of the component labels. We investigate the effects of this tuning parameter ξ in Sect. 4.5. While it is technically possible to allow feature dependency of the conditional mixture weights which depend on $\boldsymbol{x}$ in (8), further research is required to investigate the effect of the entropy-based penalty in this case.

4 Numerical experiments

We now investigate our framework in terms of predictive and estimation performance. To this end, we first compare our neural mixture of experts distributional regression (NMDR) approach with EM-based optimization routines to demonstrate competitiveness with state-of-the-art procedures. For this comparison, the model class considered is restricted to only contain constant mixture weights and linear predictors for the distributional parameters. We then evaluate our approach considering a mixture of generalized additive regression models with additional noise variables to demonstrate the framework's efficacy when including additive predictor structures. Finally, we investigate an overfitting mixture setting, in which we simulate situations where EM-based optimization fails and NMDR with entropy-based penalty leads to superior performance. In Appendix, we also perform an empirical investigation of different optimization routines in the neural network context to provide a practical guideline for users of our framework. Results show that adaptive methods are more robust compared to a vanilla stochastic gradient descent optimization. A second simulation study in Appendix further investigates the model performance of different approaches when using concomitant variables.

4.1 Evaluation metrics

We measure the estimation performance of both the regression coefficients and the mixture weights $\boldsymbol{\pi}$ using the root mean squared error (RMSE), the prediction

performance using the predictive log scores (PLS; Gelfand and Dey 1994) on an independent test dataset of the same size as the training dataset and the differences between the true and estimated class memberships using the adjusted Rand index (ARI; Hubert and Arabie 1985) as well as the accuracy (ACC) by deriving the estimated class memberships based on the maximum a posteriori probabilities for each of the mixture components. In order to deal with the problem of label switching for the non-label-invariant performance measures, we first determine the labeling which induces the accuracy-optimal assignment and then calculate these performance metrics using this labeling. The accuracy-optimal labeling is computed using the true class memberships (known in this case as data is simulated) and the estimated class memberships as induced by the mixture posterior probabilities estimated by each model.

4.2 Initialization and optimization

The weights in all our experiments are initialized using a Xavier uniform initialization (Glorot and Bengio 2010). If not specified otherwise, we use RMSprop as optimizer with a learning rate of 0.001, a maximum of 10,000 iterations, early stopping with a patience of 250 epochs, an additional reduction of the learning rate when reaching a plateau for 150 epochs, a train validation split of 0.1 and a batch size of 32.

4.3 Comparison with EM-based optimization

We first compare the NMDR framework to an EM-based algorithm implemented in the R package `gamlss.mx` (Stasinopoulos and Rigby 2016) allowing for mixtures of various distributional regressions. We use $n \in \{300, 2500\}$ observations, $M \in \{2, 3, 5, 10\}$ identically distributed mixture components, either following a Gaussian, a Laplace or a logistic distribution, each defined by location and scale parameter, and mixture weights are randomly drawn such that the smallest weight is not less than 3%. We use $p_m \in \{2, 10\}$ features for each distribution and distribution parameter in the mixture, and uniformly sampled regression coefficients from a $\mathcal{U}(-2, 2)$-distribution. While we test `gamlss.mx` with a fixed budget of 20 restarts, we compare these results to NMDR using 1 and 3 random initializations to assess the effect of multiple restarts. Each experimental configuration is replicated 10 times. All fitted models in this simulation are correctly specified, i.e., they correspond to the data generating process.

4.3.1 Results

Results are visualized in Fig. 2 and yield four important findings. First, the EM-based approach only provides results in case $p_m = 2$. The EM-based approach is not able to converge to any meaningful solution in all settings with $p_m = 10$, whereas NMDR's performance is affected by the increased number of predictors, but still yields reasonable results for $n = 2500$, also without restarts. Second, in the case

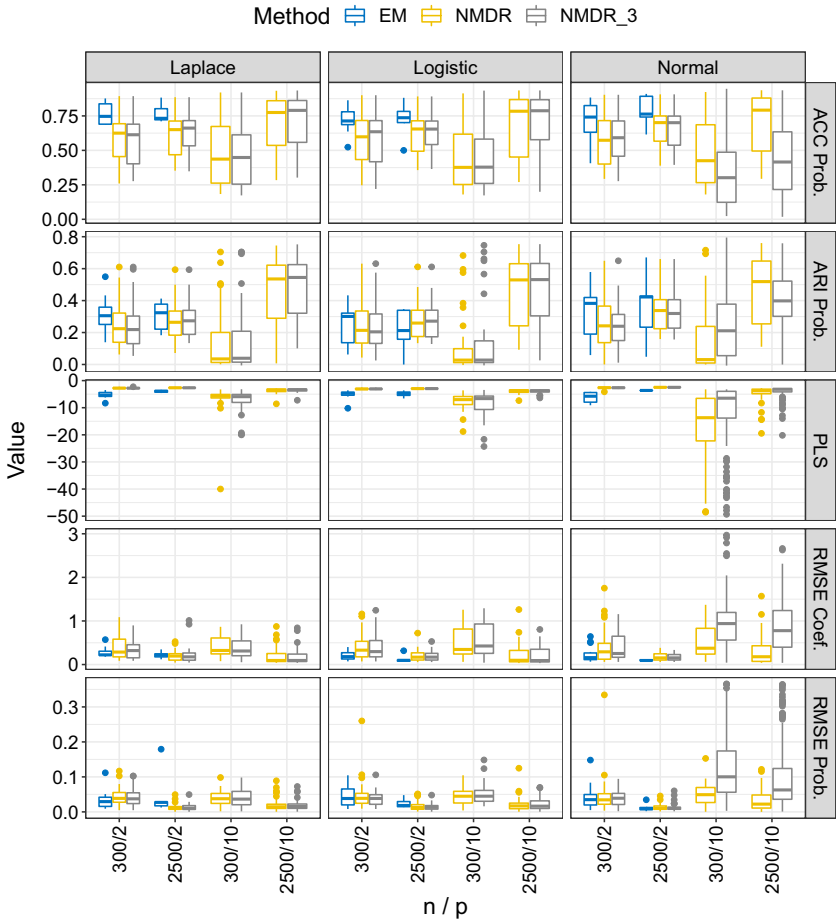

Fig. 2 Comparison of EM-based optimization (EM), NMDR with one (NMDR) or three (NMDR_3) optimization runs for different distributions (columns), measures (rows) and combinations of n and $p = p_m$. Boxplots contain all 10 runs and the four different settings for M (i.e., in total 40 data points per boxplot). Missing boxplots for EM are due to missing solutions caused by missing values when comparing current results to a convergence threshold. RMSEs for coefficients > 3 and PLS values < -50 are omitted to improve readability

$p_m = 2$, the EM-based approach provides in general a better classification performance than NMDR (as indicated by ACC Prob. and ARI Prob.). Third, the difference in estimation performance in case $p_m = 2$ between the EM-based and the NMDR approach is often negligible in terms of the RMSE between the estimated parameters. Fourth, while the induced regularization of the SGD-based routine induces a bias in the estimation and hence typically larger estimation errors, the predictive performance of NMDR is always better compared to the EM-based approach. Note that the best model for NMDR with multiple restarts is chosen based

on the in-sample log score which does not necessarily imply better out-of-sample performance compared to a single optimization run.

4.4 Mixture of additive regression models

Next, we investigate mixtures of mean regression models with nonlinear smooth effects in the additive predictors of the mean distribution parameter. We generate $n = 2500$ data points from $M = 3$ mixtures of Poisson or normal distributions with the additive structured predictor of the means defined by $\eta_1(\boldsymbol{x}) = \beta_0 + \phi_1(x_1) + \phi_2(x_2)$, $\eta_2(\boldsymbol{x}) = \beta_0 + \phi_2(x_1) + x_2$ and $\eta_3(\boldsymbol{x}) = \beta_0 + x_1 + \phi_3(x_2)$ with $\beta_0 = 0.5$, $\phi_1(x) = 2\sin(3x)$, $\phi_2(x) = \exp(2x)$ and $\phi_3(x) = 0.2x^{11}(10(1-x))^6 + 10(10x)^3(1-x)^{10})$. All covariates are independently drawn from a uniform distribution $\mathcal{U}(0, 1)$. We model all nonlinear effects using thin-plate regression splines from Wood (2017) for all methods. Note that for large data sets, other basis functions are preferred as the preprocessing step for setting up thin-plate regression spline bases scales quadratically with the number of observations (Wood 2003). Since the preprocessing is done prior to setting up the neural network, our proposed approach can be flexibly combined with any of the existing spline basis function approaches. For Poisson data, we use $h(\cdot) = \exp(\cdot)$ and the identity for the Gaussian case. We vary $\boldsymbol{\pi}$ to be either (1/3, 1/3, 1/3) or (1/10, 3/10, 6/10) and add 3 or 10 noise variables that are also included in the model as nonlinear smooth predictors for the expectation parameter. We use two different scale values 2 or 4, which either define the Gaussian variance in each mixture component or a multiplicative offset effect in the Poisson case. Our method is compared with a state-of-the-art implementation of mixtures of additive models using the R package *flexmix* (Leisch 2004) and, as an oracle, a generalized additive model with varying coefficients for all smooth effects, where the class label (unknown to the other two approaches) is used as the varying parameter. These two methods determine the smoothing parameters via an outer optimization loop as implemented in `mgcv` (see Wood 2017, for details). For NMDR, the smoothing parameters are determined as described in Sect. 3.3 via the respective degrees of freedom which are set equally for all smooths to 10 for normal and 6 for Poisson distribution. This process results in "equally smooth" nonlinear effects a priori. The combination of gradient descent updates and early stopping introduces additional regularization. This process favors parameters that result in a low prediction error, akin to other out-of-sample or model selection information criteria commonly employed to determine smoothing parameters (Wood and Fasiolo 2017). The adaptive learning rates designated for individual parameters can further lead to variable penalization across different additive components.

4.4.1 Results

The comparison for all settings is depicted in Fig. 3 showing the average log score (LS; calculated on the training dataset using the estimated model parameters) and PLS, as well as the ARI and ACC. The boxplots summarize all 10 simulation

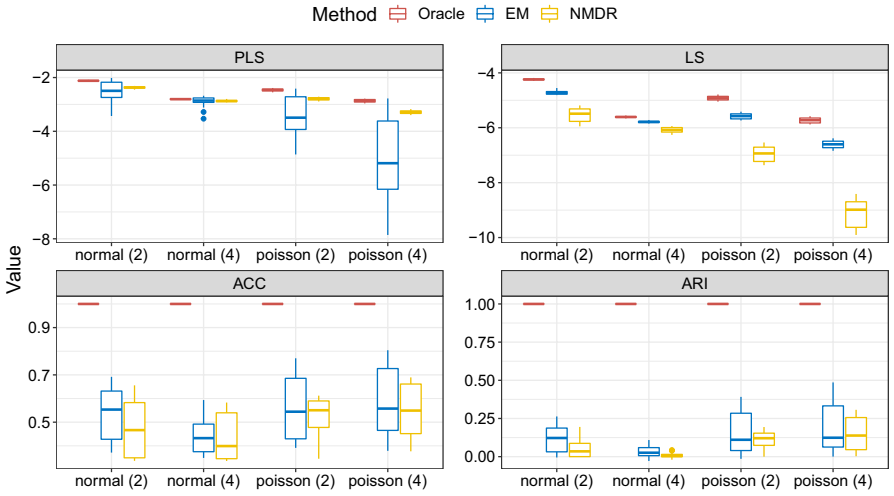

Fig. 3 Comparison of average PLS and LS as well as ACC and ARI of a state-of-the-art implementation (EM), an oracle varying coefficient model with known class memberships and our approach (NMDR) in different colors for the two distributions and two scales (*x*-axis). The boxplots contain the results for 10 replications over two settings for the mixture weights and two settings for the number of noise variables

replications for the two different mixture weights and the two number of noise variables settings. Results suggest that our approach leads to better predictions measured by the PLS, but is inferior in terms of LS. The smaller LS values are possibly due to fewer data points to fit the model because of the need for a validation set and due to the shrinkage induced by early stopping the procedure. The median performance of the clustering induced based on the estimated posterior probabilities is in general on par with the EM-based approach in terms of ARI and ACC with *flexmix*, while showing even slightly better performance in the Gaussian case.

4.5 Misspecified mixtures and sparsity

In this simulation, we use a normal mixture with $p_m = 10$ fixed linear predictors for each distribution and distribution parameter (mean and variance), where all features are again drawn from a standard normal distribution and regression coefficients from a uniform $\mathcal{U}(-2, 2)$-distribution. The data are then generated using $M = 2$ actual mixture components with π_1 drawn (independently of features) from a uniform distribution on the interval $(0.06, 0.094)$ and $\pi_2 = 1 - \pi_1$ to ensure that the minimum value of both probabilities is at least 6%. We then evaluate the estimation of mixture probabilities by NMDR for $n \in \{300, 2500\}$ when increasing the number of specified distributions $M^\dagger \in \{3, 5, 10\}$. To allow for sparsity in $\boldsymbol{\pi}$, we use the objective function ℓ_{ent} introduced in Sect. 3.3. For each scenario, 10 replications are performed.

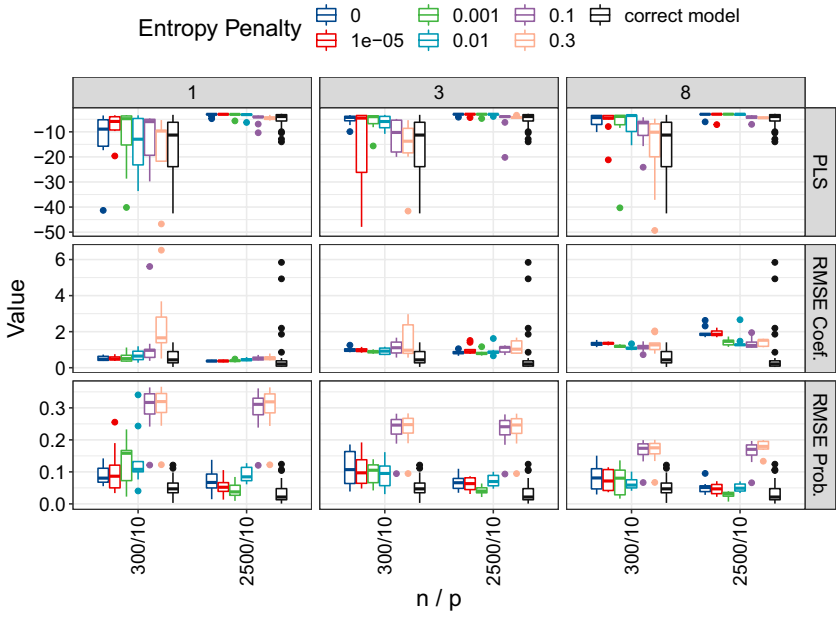

Fig. 4 Model quality for misspecified models with specified mixtures $M^{\dagger} \in \{3, 5, 10\}$ (columns, counting the additional components) instead of actual $M = 2$ mixtures and different goodness of fit measures (rows) for 10 replications (boxes). Colors correspond to different settings of ξ or represent estimation results of the correct model (black)

4.5.1 Results

Results for various settings of the entropy penalty parameter ξ are depicted in Fig. 4. While setting ξ to small values larger than zero can improve the predictive performance and even outperform the correctly specified model without

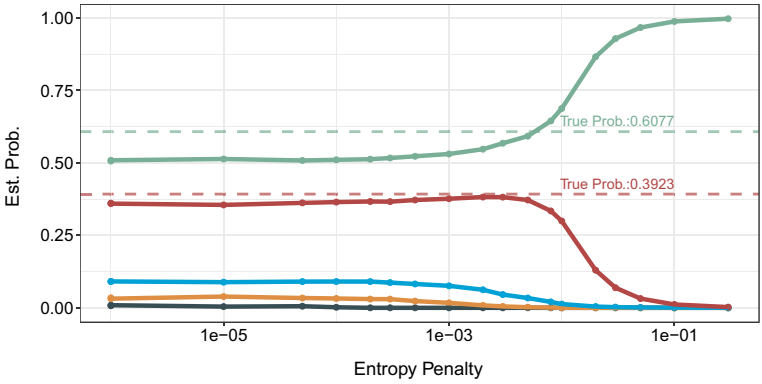

Fig. 5 Coefficient path (estimated mixture probabilities) for different entropy penalties. A value of around 1e-02 yields the best trade-off between sparsity and estimation performance

additional distribution components, the bias induced by the penalty generally decreases the estimation performance. In practice, an appropriate amount of penalization can be found by running cross-validation along a grid of different ξ values as, e.g., done for the Lasso (Tibshirani 1996).

We additionally investigate the coefficient path obtained from varying values of the penalty parameter ξ for one simulated example. Results for the $n = 2500, M^{\dagger} = 5$ setting are depicted in Fig. 5. ξ is varied between 0 and 1 on a logarithmic scale. The true model has two nonzero probabilities 0.6077 and 0.3923 while the other 3 entries in $\boldsymbol{\pi}$ are 0.

5 Cell cycle-regulated genes of yeast

In order to demonstrate the flexibility of our approach, we investigate its application to the yeast cell cycle dataset from Spellman et al. (1998). In this study, genome-wide mRNA levels were measured for 6178 yeast open reading frames (ORFs) for 119 min at 7-min intervals. We here analyze the subset of data where all 18 time points for the alpha factor arrest are available. The resulting longitudinal dataset consists of 80,802 observations of the standardized expression levels. A subset of this dataset was also analyzed using mixture models in Grün et al. (2011).

5.1 Distributional mixture regression

As both the mean and standard deviation of the standardized expression levels of genes change over time, we apply a mixture of distributional regressions model where the mean μ and the standard deviation σ of the normally distributed mixture components depend on time, i.e., $Y_t \sim \sum_{m=1}^{M} \pi_m \mathcal{N}(\mu_{m,t}, \sigma_{m,t}^2)$ for $t \in [0, 119]$. The additive structured predictors for these distribution parameters are defined as

$$h^{-1}(\mu_{m,t}) = \beta_{0,m,1} + \phi_{m,1}(t) \quad \text{and} \quad h^{-1}(\sigma_{m,t}) = \beta_{0,m,2} + \phi_{m,2}(t),$$

where the nonlinear smooth functions ϕ are modeled by thin-plate regression splines (Wood 2003). Previous approaches for modeling this dataset investigated the use of a mixture of mixed models, i.e., the inclusion of gene-specific random effects (Luan and Li 2003; Grün et al. 2011). We investigate here an alternative option for modeling this additional heterogeneity by allowing for time-varying standard deviations. We use $M = 6$ which corresponds to the number of mixture components identified by Spellman et al. (1998).

5.2 Results

In order to plot the estimated smooth effects together with the true observations, we first derive the component membership of every gene. As done in the E-step of mixture model approaches (see, e.g., Grün et al. 2011), we calculate the a posteriori

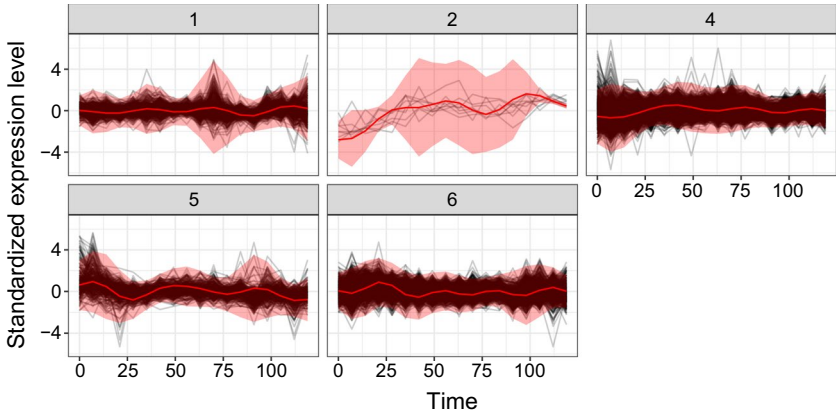

Fig. 6 Trajectories of ORFs (y-axis) for all 4489 genes (black lines) over the course of the 18 time points (x-axis) with the estimated mean trend (red solid line) per component (facets) and uncertainty visualized by two times the estimated (time-varying) standard deviation (shaded red area)

probability for every gene to belong to component m and then take the maximum of all components $1, \ldots, M = 6$. For this application, observations were only assigned to 5 of the 6 assumed components. Note that due to the nature of our optimization routine, not all components necessarily contain at least one observation.

Comparing the number of genes assigned to each cluster, one sees that the most common component in our results is cluster 6 with 39,078 observations. Cluster 4 contains 25,722 observations, cluster 1 9306 and cluster 5 6552 observations. Least number of observations is assigned to cluster 2 which contains only 144 observations.

Figure 6 visualizes the results obtained in a panel plot where in each panel the trajectories of the ORFs for all genes assigned to this cluster are shown together with the component-specific estimates of the time-varying means and standard deviations. The identified clusters clearly vary in showing either an initial decrease or increase in their means. In addition, one also sees that the standard deviations of the clusters vary over time with some clusters exhibiting a particularly large amount of heterogeneity at later time points.

6 Outlook

We have introduced the class of mixtures of experts distributional regression with additive structural predictors and investigated its embedding into neural networks for robust model estimation. Overall this leads to the neural mixture of experts distributional regression (NMDR) approach. We show that popular first-order adaptive update routines are well suited for learning these mixture of experts (distributional) regression models and also highlight that the embedding into a neural network estimation framework allows for straightforward extensions of the

general mixture model class and (regularized) maximum-likelihood estimation using optimization routines suitable also for big data applications due to mini-batch training. Using the proposed architecture for mixture of experts distributional regression, a possible extension of our approach is therefore the combination with other (deep) neural networks. This allows learning both the distribution components and the mixture weights either by (a) a structured model, such as a linear or additive model, (b) a custom (deep) neural network or (c) a combination thereof. A similar approach has been investigated by Fritz et al. (2022) using a zero-inflated Poisson model (i.e., a mixture including a point mass distribution) which includes both additive effects and a graph neural network in the additive predictor.

Appendix A: Optimization routines

In order to provide insights into the various optimizers' performance, we conduct a small benchmark study to assess the influence of the choice of an optimizer and to find a good default. Figure 7 visualizes the comparison based on the ranks for each optimizer across all 48 settings from Sect. 4.3. In all our experiments, we use the Glorot initializer to initialize weights (Glorot and Bengio 2010) (initialized differently for every optimizer and restart) a batch size of 50 and a maximum of 1500 epochs.

Results

Figure 7 indicates that convergence problems are primarily encountered for SGD with a substantial amount of runs diverging during optimization. An overall performance assessment based on ranks indicates that the *RMSprop* optimizer achieves the lowest overall rank. However, the figure highlights that in fact, no clear best optimizer emerges across all scenarios. The ranks obtained for the different optimizers also vary considerably with the performance criterion.

Overall one might conclude that both the RMSprop optimizer and the Adam optimizer perform in general well, also when used with their default settings. We note, however, that tuning the optimizer and its learning rate would in general also be beneficial in terms of both predictive performance and estimation quality. A further speed-up for some of these routines can be achieved by additionally incorporating momentum, which also proved to be effective in the optimization of additive models using boosting (Schalk et al. 2022).

Appendix B: Concomitant variables

Finally, we compare our approach with `flexmix` in terms of estimation quality when the model includes concomitant variables.

Data generating process

We simulate a mixture regression model with $M \in \{2, 3\}$ mixture components, each following a normal distribution with fixed variance, $\sigma^2 = 1$, and mean defined by $p = 10$ covariates that are drawn from a standard normal distribution and multiplied by fixed coefficient also generated from a standard normal distribution. In contrast to previous simulations, this numerical experiment is based on a data generating process that uses additional 2, 4 or 8 covariates to define the additive predictors

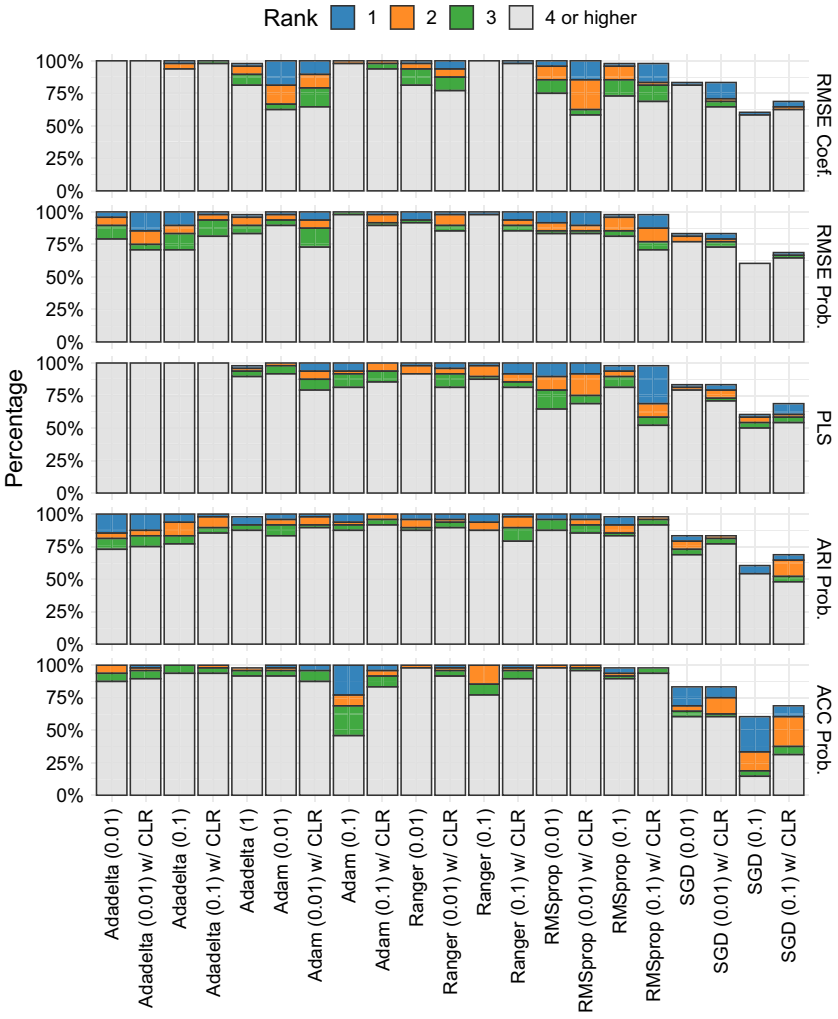

Fig. 7 Comparison of various optimizers (x-axis; with learning rate in brackets) ranked by performance for different metrics (different rows), potentially with additional cyclic learning rate schedule (CLR), across simulation settings studied in Sect. 4.3. Ranks are computed on performance averaged over 3 repetitions. Bars from SGD runs do not sum up to 100% as some models diverged during optimization

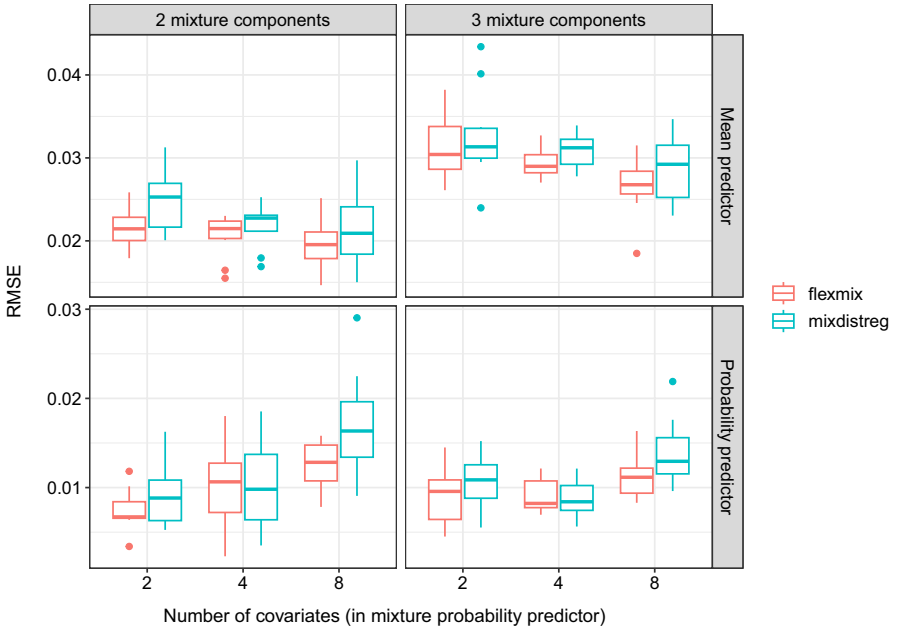

Fig. 8 Comparison of `flexmix` and `mixdistreg` estimation performance using the RMSE (visualized by boxplots summarizing the 10 different runs) for coefficients in the mean predictors of distribution members (top row) as well as additive predictor of mixture probabilities (bottom row) for a mixture of $M \in \{2, 3\}$ distributions (columns)

$\eta_{K+1}, \ldots, \eta_{K+M}$. In other words, we do not set the mixture probabilities to a fixed constant but make them covariate-dependent. The predictors $\eta_{K+1}, \ldots, \eta_{K+M}$ are generated independently from predictors $\eta_1, \ldots, \eta_K$ by drawing covariates and effects from a standard normal distribution and defining their effect to be linear.

Experiment

We set $n = 5000$ and run the experiment 10 times to investigate the estimation performance of the two methods for all regression coefficients using the RMSE. `mixdistreg` is optimized using Adam with a learning rate 1e-3, batch size of 32, early stopping on a 10% validation data set and a patience of 250 epochs.

Results

Figure 8 shows the resulting RMSE values for different values of M and the two additive predictor types (means and probabilities). Both approaches perform well with RMSE values in the range 0.01 to 0.02, which is comparably small compared to the true coefficient values which range from -2 to 1.5. As in previous studies, the estimation performance of the EM-based approach is slightly better than the one using SGD.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Competing financial interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abadi, M., Agarwal, A., Barham, P., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). https://www.tensorflow.org/

Bishop, C.M.: Mixture density networks. Aston University (1994)

DeSarbo, W.S., Cron, W.L.: A maximum likelihood methodology for clusterwise linear regression. J. Classif. **5**(2), 249–282 (1988)

Diebolt, J., Robert, C.P.: Estimation of finite mixture distributions through Bayesian sampling. J. R. Stat. Soc.: Ser. B (Methodol.) **56**(2), 363–375 (1994)

Dillon, J.V., Langmore, I., Tran, D., et al.: TensorFlow distributions. (2017). https://doi.org/10.48550/arXiv.1711.10604

Fritz, C., Dorigatti, E., Rügamer, D.: Combining graph neural networks and spatio-temporal disease models to improve the prediction of weekly covid-19 cases in Germany. Sci. Rep. **12**(1), 1–18 (2022)

Frühwirth-Schnatter, S.: Finite Mixture and Markov Switching Models. Springer, Berlin (2006)

Gelfand, A.E., Dey, D.K.: Bayesian model choice: asymptotics and exact calculations. J. R. Stat. Soc.: Ser. B (Methodol.) **56**(3), 501–514 (1994)

Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, vol. 9, pp. 249–256. PMLR (2010)

Gormley, I.C., Frühwirth-Schnatter, S.: Mixture of experts models. In: Frühwirth-Schnatter, S., Celeux, G., Robert, C.P. (eds.) Handbook of Mixture Analysis, pp. 271–307. Chapman & Hall/CRC Handbooks of Modern Statistical Methods, Chapman and Hall/CRC (2019)

Grün, B., Leisch, F.: Fitting finite mixtures of generalized linear regressions in R. Comput. Stat. Data Anal. **51**(11), 5247–5252 (2007)

Grün, B., Scharl, T., Leisch, F.: Modelling time course gene expression data with finite mixtures of linear additive models. Bioinformatics **28**(2), 222–228 (2011)

Hinton, G., Srivastava, N., Swersky, K.: Neural networks for machine learning. Coursera, video lectures **264**(1) (2012)

Hubert, L., Arabie, P.: Comparing partitions. J. Classif. **2**(1), 193–218 (1985)

Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014). https://doi.org/10.48550/arXiv.1412.6980

Kneib, T.: Beyond mean regression. Stat. Model. **13**(4), 275–303 (2013)

Leisch, F.: FlexMix: A general framework for finite mixture models and latent class regression in R. J. Stat. Softw. **11**(8), 1–18 (2004). https://doi.org/10.18637/jss.v011.i08

Luan, Y., Li, H.: Clustering of time-course gene expression data using a mixed-effects model with B-splines. Bioinformatics **19**(4), 474–482 (2003)

Magdon-Ismail, M., Atiya, A.: Neural networks for density estimation. In: Advances in Neural Information Processing Systems (1998)

McLachlan, G.J., Peel, D.: Finite Mixture Models. John Wiley & Sons, London (2004)
McLachlan, G.J., Lee, S.X., Rathnayake, S.I.: Finite mixture models. Annu. Rev. Stat. Appl. **6**(1), 355–378 (2019)
Quandt, R.E.: The estimation of the parameters of a linear regression system obeying two separate regimes. J. Am. Stat. Assoc. **53**(284), 873–880 (1958)
R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2022). https://www.R-project.org/
Rigby, R.A., Stasinopoulos, M.D.: GAMLSS: a distributional regression approach. J. R. Stat. Soc.: Ser. C (Appl. Stat.) **54**(3), 507–554 (2005)
Rügamer, D., Kolb, C., Fritz, C., et al.: deepregression: A flexible neural network framework for semi-structured deep distributional regression. J. Stat. Softw. **105**(1), 1–31 (2023a)
Rügamer, D., Kolb, C., Klein, N.: Semi-structured distributional regression. Am. Stat. 1–12 (2023b)
Schalk, D., Bischl, B., Rügamer, D.: Accelerated componentwise gradient boosting using efficient data representation and momentum-based optimization. J. Comput. Graph. Stat. 1–11 (2022)
Späth, H.: Algorithm 39 clusterwise linear regression. Computing **22**(4), 367–373 (1979)
Spellman, P.T., Sherlock, G., Zhang, M.Q., et al.: Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. Mol. Biol. Cell **9**(12), 3273–3297 (1998)
Stasinopoulos, D.M., Rigby, R.A.: Generalized additive models for location scale and shape (GAMLSS) in R. J. Stat. Softw. **23**(7), 1–46 (2007)
Stasinopoulos, M., Rigby, B.: gamlss.mx: Fitting Mixture Distributions with GAMLSS. R package version 4.3-5 (2016)
Stasinopoulos, M.D., Rigby, R.A., Bastiani, F.D.: GAMLSS: a distributional regression approach. Stat. Model. **18**(3–4), 248–273 (2018)
Tibshirani, R.: Regression shrinkage and selection via the lasso. J. R. Stat. Soc.: Ser. B (Methodol.) **58**(1), 267–288 (1996)
Ushey, K., Allaire, J., Tang, Y.: reticulate: Interface to 'Python'. https://CRAN.R-project.org/package=reticulate, r package version 1.26 (2022)
Van den Oord, A., Schrauwen, B.: Factoring variations in natural images with deep Gaussian mixture models. In: Advances in Neural Information Processing Systems, pp. 3518–3526 (2014)
Van Rossum, G., Drake Jr, F.L.: Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam (1995)
Viroli, C., McLachlan, G.J.: Deep Gaussian mixture models. Stat. Comput. **29**(1), 43–51 (2019)
Wedel, M., DeSarbo, W.S.: A mixture likelihood approach for generalized linear models. J. Classif. **12**(1), 21–55 (1995)
Wood, S.N.: Thin plate regression splines. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **65**(1), 95–114 (2003)
Wood, S.N.: Generalized Additive Models: An Introduction with R. CRC Press (2017)
Wood, S.N., Fasiolo, M.: A generalized Fellner–Schall method for smoothing parameter optimization with application to Tweedie location, scale and shape models. Biometrics **73**(4), 1071–1081 (2017). https://doi.org/10.1111/biom.12666
Wright, L.: New deep learning optimizer, ranger: Synergistic combination of radam lookahead for the best of... Medium (2019)
Zeiler, M.D.: ADADELTA: An adaptive learning rate method (2012). https://doi.org/10.48550/arXiv.1212.5701
Zong, B., Song, Q., Min, M.R., et al.: Deep autoencoding Gaussian mixture model for unsupervised anomaly detection. In: International Conference on Learning Representations (2018)