Gamma Mixture Density Networks and their application to modelling insurance claim amounts

Łukasz Delong

SGH Warsaw School of Economics
Institute of Econometrics, Division of Probabilistic Methods

Joint work with Mathias Lindholm, Stockholm University, Department of Mathematics, and Mario V. Wüthrich, ETH Zurich, Department of Mathematics

Motivation

- Modelling of insurance claim amounts is challenging since simple loss distributions usually lead to a poor fit,
- The reasons for this are: unobserved heterogeneity, multi-modality, heavy tailedness of claim amounts and different tail behavior of small and large claim amounts,
- **One solution:** We can try to fit mixtures of distributions,
- Mixtures of Erlang distributions (Gamma distributions with integer shape parameters) with a common rate parameter play an important role here since such mixtures are dense in the space of positive continuous distributions in the weak convergence sense.

Motivation

- The additional complexity in loss distribution modelling comes from the fact that the parameters of the loss distribution vary across individuals and their values can be related to fixed effects explained by observable features of the individuals,
- In actuarial pricing, in case of claim severities, the most common approach is to fit a Generalized Linear Model (GLM) or Generalized Additive Model (GAM) with Gamma responses to differentiate the expected claim severity across policyholders based on their features,
- The classical fit of Gamma GLM/GAM to claim amounts might be unsatisfactory, again due to the characteristics of the insurance claim amounts mentioned above,
- **One solution:** We can try to fit mixtures of GLMs/GAMs/experts to model the parameters of the distributions as functions of the features.

Goal of the study

- We investigate mixtures of Gamma distributions with all parameters related to features which characterize the individuals,
- The goal is to fit the density:

$$f(y|\boldsymbol{x}) = \sum_{k=1}^{K} p_k(\boldsymbol{x}) \frac{\beta(\boldsymbol{x})^{\alpha_k(\boldsymbol{x})}}{\Gamma(\alpha_k(\boldsymbol{x}))} y^{\alpha_k(\boldsymbol{x})-1} e^{-\beta(\boldsymbol{x})y}, \quad y > 0, \tag{1}$$

  where the mixing probabilities $(p_k)_{k=1}^K$, the shape parameters $(\alpha_k)_{k=1}^K$ and the rate parameter β depend on a d-dimensional vector of features $\boldsymbol{x} = (x_1, \ldots, x_d)$,
- **The novel part:** We use neural networks with two architectures to model the parameters $\left((p_k)_{k=1}^K, (\alpha_k)_{k=1}^K, \beta\right)$ in the mixture of Gamma distributions as functions of features $\boldsymbol{x}$,
- We call (1) by Gamma Mixture Density Network (Gamma MDN).

Contribution to the literature

- The Gamma MDN with mixing probabilities, shape and rate parameters all depending on features is a next natural step in modelling claim amounts,
- Compared to mixtures of experts and mixtures of GLMs, by applying neural networks we gain more flexibility in finding non-linear relations between features and responses, including interactions between features, and we can model dispersion coefficients with features,
- By applying early stopping and regularization, we control over-fitting of our network regression model, from which the full mixture of experts suffers,
- We test calibration strategies for Gamma MDNs and we develop two versions of the Expectation-Maximization algorithm for fitting Gamma MDNs, which we call the EM network boosting algorithm and the EM forward network algorithm,
- We illustrate an actuarial application on a real data set and we conclude that Gamma MDNs can improve the fit of the regression model and the predictions of the claim severities compared to classical actuarial technique.

Two architectures of Gamma MDN

- We choose a neural network with $M \in \mathbb{N}$ hidden layers and $q_m \in \mathbb{N}$ hidden neurons in each hidden layer $m = 1, \ldots, M$,
- We define the network layers as follows

$$\begin{aligned} \boldsymbol{x} \in \mathbb{R}^{q_{m-1}} &\mapsto \theta^m(\boldsymbol{x}) &&= \left(\theta_1^m(\boldsymbol{x}), \ldots, \theta_{q_m}^m(\boldsymbol{x})\right)' \in \mathbb{R}^{q_m}, \quad m = 1, \ldots, M, \\ \boldsymbol{x} \in \mathbb{R}^{q_{m-1}} &\mapsto \theta_r^m(\boldsymbol{x}) &&= \varphi\left(b_r^m + \langle \boldsymbol{w}_r^m, \boldsymbol{x} \rangle\right), \quad r = 1, \ldots, q_m, \end{aligned}$$

where $\varphi : \mathbb{R} \to \mathbb{R}$ denotes an activation function, $\boldsymbol{w}_r^m \in \mathbb{R}^{q_{m-1}}$ denotes network weights, $b_r^m \in \mathbb{R}$ denotes a bias term, and $\langle \cdot, \cdot \rangle$ denotes a scalar product in $\mathbb{R}^{q_{m-1}}$,

- The mapping:

$$\boldsymbol{x} \in \mathbb{R}^{q_0} \mapsto \Theta^{M+1}(\boldsymbol{x}) = \left(\Theta_1^{M+1}(\boldsymbol{x}), \ldots, \Theta_{q_{M+1}}^{M+1}(\boldsymbol{x})\right)' \in \mathbb{R}^{q_{M+1}},$$

with a composition of the network layers $\theta^1, \ldots, \theta^M$, and the components

$$\boldsymbol{x} \mapsto \Theta_r^{M+1}(\boldsymbol{x}) = b_r^{M+1} + \left\langle \boldsymbol{w}_r^{M+1}, \left(\theta^M \circ \cdots \circ \theta^1\right)(\boldsymbol{x}) \right\rangle, \quad r = 1, \ldots, q_{M+1},$$

gives us the prediction based on $\boldsymbol{x}$ in the output layer $M+1$ of dimension q_{M+1}, on the canonical scale of the parameter.

Two architectures of Gamma MDN

Architecture 1: Three separate neural networks for $\left((p_k)_{k=1}^K, (\alpha_k)_{k=1}^K, \beta\right)$:

- The parameters for an individual case i are determined by

$$\begin{aligned} \beta_i &= \beta(\boldsymbol{x}_i) = e^{g_{\beta,1}(\boldsymbol{x}_i)}, \\ g_{\beta,1}(\boldsymbol{x}_i) &= c_\beta \log\left(\hat{\beta}^{\text{init}}(\boldsymbol{x}_i)\right) + b_\beta^{M+1} + \left\langle \boldsymbol{w}_\beta^{M+1}, \left(\theta_\beta^M \circ \cdots \circ \theta_\beta^1\right)(\boldsymbol{x}_i)\right\rangle, \end{aligned}$$

$$\begin{aligned} \alpha_{i,k} &= \alpha_k(\boldsymbol{x}_i) = e^{g_{\alpha,k}(\boldsymbol{x}_i)}, \\ g_{\alpha,k}(\boldsymbol{x}_i) &= c_{\alpha,k} \log\left(\hat{\alpha}_k^{\text{init}}(\boldsymbol{x}_i)\right) + b_{\alpha,k}^{M+1} + \left\langle \boldsymbol{w}_{\alpha,k}^{M+1}, \left(\theta_\alpha^M \circ \cdots \circ \theta_\alpha^1\right)(\boldsymbol{x}_i)\right\rangle, \end{aligned}$$

$$\begin{aligned} p_{i,k} &= p_k(\boldsymbol{x}_i) = \frac{e^{g_{p,k}(\boldsymbol{x}_i)}}{\sum_{u=1}^K e^{g_{p,u}(\boldsymbol{x}_i)}}, \\ g_{p,k}(\boldsymbol{x}_i) &= c_{p,k} \log\left(\hat{p}_k^{\text{init}}(\boldsymbol{x}_i)\right) + b_{p,k}^{M+1} + \left\langle \boldsymbol{w}_{p,k}^{M+1}, \left(\theta_p^M \circ \cdots \circ \theta_p^1\right)(\boldsymbol{x}_i)\right\rangle, \end{aligned}$$

where $\hat{\beta}^{\text{init}}(\boldsymbol{x}_i)$, $(\hat{\alpha}_k^{\text{init}}(\boldsymbol{x}_i))_{k=1,\ldots,K}$ and $(\hat{p}_k^{\text{init}}(\boldsymbol{x}_i))_{k=1,\ldots,K}$ denote initial estimates of the parameters for individual i, and c_β, $(c_{\alpha,k})_{k=1,\ldots,K}$ and $(c_{p,k})_{k=1,\ldots,K}$ denote trainable weights related to the initial estimates.

Two architectures of Gamma MDN

Architecture 1: Three separate neural networks for $\left((p_k)_{k=1}^K, (\alpha_k)_{k=1}^K, \beta\right)$:

- After the three networks are jointly calibrated, the parameters are estimated with the transformations:

$$
\begin{aligned}
\hat{\beta}_i &= \left(\hat{\beta}^{\text{init}}(\boldsymbol{x}_i)\right)^{\hat{c}_\beta} e^{\hat{\Theta}_{\beta,1}^{M+1}(\boldsymbol{x}_i)}, \\
\hat{\alpha}_{i,k} &= \left(\hat{\alpha}_k^{\text{init}}(\boldsymbol{x}_i)\right)^{\hat{c}_{\alpha,k}} e^{\hat{\Theta}_{\alpha,k}^{M+1}(\boldsymbol{x}_i)}, \quad k = 1, \ldots, K, \\
\hat{p}_{i,k} &= \frac{\left(\hat{p}_{i,k}^{\text{init}}(\boldsymbol{x}_i)\right)^{\hat{c}_{p,k}} e^{\hat{\Theta}_{p,k}^{M+1}(\boldsymbol{x}_i)}}{\sum_{u=1}^K \left(\hat{p}_{i,u}^{\text{init}}(\boldsymbol{x}_i)\right)^{\hat{c}_{p,u}} e^{\hat{\Theta}_{p,u}^{M+1}(\boldsymbol{x}_i)}}, \quad k = 1, \ldots, K,
\end{aligned}
$$

- The weights and the bias terms depend on the network we fit for the parameter, this is highlighted by lower subscripts in c and Θ^{M+1}.

Two architectures of Gamma MDN

Architecture 2: One neural network for $\left((p_k)_{k=1}^K, (\alpha_k)_{k=1}^K, \beta\right)$:

- We use only one neural network with output dimension $q_{M+1} = 2K + 1$ in layer $M + 1$, to which offsets with corresponding initial estimates of the parameters are added and exponential and softmax activation functions are applied at the last step,
- After the network is calibrated, the parameters are estimated with the transformations:

$$
\begin{aligned}
\hat{\beta}_i &= \left(\hat{\beta}^{\text{init}}(\boldsymbol{x}_i)\right)^{\hat{c}_k} e^{\hat{\Theta}_k^{M+1}(\boldsymbol{x}_i)}, \quad k = 1, \\
\hat{\alpha}_{i,k} &= \left(\hat{\alpha}_k^{\text{init}}(\boldsymbol{x}_i)\right)^{\hat{c}_{1+k}} e^{\hat{\Theta}_{1+k}^{M+1}(\boldsymbol{x}_i)}, \quad k = 1, \ldots, K, \\
\hat{p}_{i,k} &= \frac{\left(\hat{p}_k^{\text{init}}(\boldsymbol{x}_i)\right)^{\hat{c}_{1+K+k}} e^{\hat{\Theta}_{1+K+k}^{M+1}(\boldsymbol{x}_i)}}{\sum_{u=K+2}^{2K+1} \left(\hat{p}_u^{\text{init}}(\boldsymbol{x}_i)\right)^{\hat{c}_u} e^{\hat{\Theta}_u^{M+1}(\boldsymbol{x}_i)}}, \quad k = 1, \ldots, K,
\end{aligned}
$$

- All parameters share the weights and the bias terms of one neural network. There is no lower subscript related to (p, α, β) in $\hat{\Theta}^{M+1}$.

EM algorithm

- Mixtures of distributions/GLMs/experts are usually estimated with a version of the Expectation-Maximization algorithm,
- Let $D = (y_i, \boldsymbol{x}_i)_{i=1}^n$ denote the data set at our disposal, and let $L\Big(p, \alpha, \beta \Big| D\Big)$ denote the corresponding log-likelihood:

$$L\Big(p, \alpha, \beta \Big| (y_i, \boldsymbol{x}_i)_{i=1}^n\Big) = \sum_{i=1}^n \log\Big(\sum_{k=1}^K p_k(\boldsymbol{x}_i) \frac{\beta(\boldsymbol{x}_i)^{\alpha_k(\boldsymbol{x}_i)}}{\Gamma(\alpha_k(\boldsymbol{x}_i))} y_i^{\alpha_k(\boldsymbol{x}_i)-1} e^{-\beta(\boldsymbol{x}_i) y_i}\Big).$$

- To any observation $(y_i, \boldsymbol{x}_i)$ we associate a vector of latent variables $\boldsymbol{z}_i = (z_{i,1}, \ldots, z_{i,K}) \in \{0,1\}^K$, which illustrates the one-hot encoding of which mixture component a particular observation y_i has been chosen from,
- We define the expected value of the complete-data log-likelihood for $(y_i, \boldsymbol{x}_i, \boldsymbol{z}_i)_{i=1}^n$ under the posterior distribution of the latent variables:

$$\mathcal{Q}\Big(p, \alpha, \beta \Big| (y_i, \boldsymbol{x}_i)_{i=1}^n\Big) = \sum_{i=1}^n \sum_{k=1}^K \hat{z}_{i,k} \log\Big(p_k(\boldsymbol{x}_i) \frac{\beta(\boldsymbol{x}_i)^{\alpha_k(\boldsymbol{x}_i)}}{\Gamma(\alpha_k(\boldsymbol{x}_i))} y_i^{\alpha_k(\boldsymbol{x}_i)-1} e^{-\beta(\boldsymbol{x}_i) y_i}\Big),$$

where $(\hat{z}_{i,k})_{k=1,\ldots,K}$ denote the posterior probabilities of the latent variables w.r.t. $(y_i, \boldsymbol{x}_i)$ for cases $i = 1, \ldots, n$.

EM algorithm

- We iterate the following computations for $\ell = 0, 1, \ldots, \ell^*$:

  Step 1: Assume the estimates of $\left((\hat{p}_{i,k}^{\ell})_{k=1}^{K}, (\hat{\alpha}_{i,k}^{\ell})_{k=1}^{K}, \hat{\beta}_i^{\ell}\right)$ are given for cases $i = 1, \ldots, n$;

  Step 2: [Expectation step] The unobserved variables $(\boldsymbol{z}_i)_{i=1,\ldots,n}$ are estimated with the posterior probabilities:

$$\hat{z}_{i,k}^{\ell} = \frac{\hat{p}_{i,k}^{\ell} \frac{\left(\hat{\beta}_i^{\ell}\right)^{\hat{\alpha}_{i,k}^{\ell}}}{\Gamma(\hat{\alpha}_{i,k}^{\ell})} y_i^{\hat{\alpha}_{i,k}^{\ell}-1} e^{-\hat{\beta}_i^{\ell} y_i}}{\sum_{k=1}^{K} \hat{p}_{i,k}^{\ell} \frac{\left(\hat{\beta}_i^{\ell}\right)^{\hat{\alpha}_{i,k}^{\ell}}}{\Gamma(\hat{\alpha}_{i,k}^{\ell})} y_i^{\hat{\alpha}_{i,k}^{\ell}-1} e^{-\hat{\beta}_i^{\ell} y_i}}, \qquad \text{for } k = 1, \ldots, K \text{ and } i = 1, \ldots, n;$$

  Step 3: [Maximization step] The function $\mathcal{Q}(p, \alpha, \beta | D)$ with $\hat{z}_{i,k} = \hat{z}_{i,k}^{\ell}$ is optimized with respect to the parameters of the regression functions for $\left((p_k)_{k=1}^{K}, (\alpha_k)_{k=1}^{K}, \beta\right)$;

  Step 4: The new estimates of $\left((\hat{p}_{i,k}^{\ell+1})_{k=1}^{K}, (\hat{\alpha}_{i,k}^{\ell+1})_{k=1}^{K}, \hat{\beta}_i^{\ell+1}\right)$ are derived from the regression functions fitted in Step 3,

- The log-likelihood $L\left(\hat{p}^{\ell}, \hat{\alpha}^{\ell}, \hat{\beta}^{\ell} \middle| D\right)$ increases in consecutive iterations $\ell = 0, 1, \ldots,$
- The EM algorithm converges to a local maximum if the likelihood is bounded.

EM algorithm

- The likelihood for our mixture of Gamma distributions is bounded,
- The estimation of neural networks of Gamma MDN in each EM loop is a not-trivial optimization procedure,
- If the networks are poorly initialized at the next iteration, then the log-likelihood $L(\hat{p}^{\ell}, \hat{\alpha}^{\ell}, \hat{\beta}^{\ell} | D)$ may decrease - this is an important difference compared to the EM algorithm for fitting mixtures of GLMs/experts,
- We should discuss how the information about the parameters should be passed effectively between the iterations of the EM algorithm and how to initialize the neural networks (weights and biases) in consecutive iterations of the EM algorithm,
- The EM network boosting algorithm: We can pass the present estimates of the parameters $\left((p_k)_{k=1}^K, (\alpha_k)_{k=1}^K, \beta\right)$ given by $\left((\hat{p}_{i,k}^{\ell})_{k=1}^K, (\hat{\alpha}_{i,k}^{\ell})_{k=1}^K, \hat{\beta}_i^{\ell}\right)_{i=1}^n$ and include them as the offsets $\left((\hat{p}_{i,k}^{\text{init}})_{k=1}^K, (\hat{\alpha}_{i,k}^{\text{init}})_{k=1}^K, \hat{\beta}_i^{\text{init}}\right)_{i=1}^n$ in the M-step of the next loop $\ell + 1$. In addition, we have to initialize the weights and bias terms (see the paper for details on four initialization procedures),
- The EM forward network algorithm: We can pass forward from one to the next loop all the fitted network parameters (the weights and biases) and drop the offsets. We initialize the weights and biases in all layers of the network trained at step $\ell + 1$ by setting the values of $\left((\boldsymbol{w}_r^m)_{r=1}^{q_m}, (b_r^m)_{r=1}^{q_m}\right)_{m=1}^{M+1}$ which are recovered from the last network trained at step ℓ.

EM algorithm

- Initializer 1: Initialize the weights using the same uniform distributions which are used to initialize the weights at the first iteration of the EM algorithm and set the biases to zero - Xavier initializer where the range of the uniform distribution depends on the number of input and output connections from the layer,
- Initializer 2: Use the same initialization strategy as above but expand the ranges of the uniform distributions,
- Initializer 3: Use the same initialization strategy as above but update the ranges of the uniform distributions so that the range of the uniform distribution used for initialization in a layer at the next iteration is not smaller than the range of the final weights recovered from the last iteration and the range produced by the Xavier initializer for the layer,
- Initializer 4: Use the same weights and biases which we were obtained in the previous iteration of the EM algorithm. Such an initialization would guarantee that the weights in the output layer would be updated in the first epoch in the same direction as the weights at the last epoch of the gradient descent algorithm at the previous EM iteration.

EM algorithm

- For the EM network boosting algorithm, the neural networks fitted in the last iteration ℓ^* depend on the parameter estimates from the previous iteration $\ell^* - 1$, since we recursively use the parameter estimates of EM loop ℓ as initial values (offsets) for EM loop $\ell + 1$,
- In order to derive the final regression models for the parameters $\left((p_k)_{k=1}^K, (\alpha_k)_{k=1}^K, \beta\right)$, we fit a so-called meta model,
- The meta model is a new Gamma MDN which reproduces the estimates $(\hat{p}_k^{\ell^*+1})_{k=1}^K, (\hat{\alpha}_k^{\ell^*+1})_{k=1}^K, \hat{\beta}^{\ell^*+1})$,
- To calibrate the meta model, we minimize the Kullback-Leibler divergence between the mixtures of Gamma distributions of the meta model and the model from the last iteration of the EM algorithm (see the paper for details),
- We use analogous architectures (architecture 1 and 2) for the meta model as for the networks fitted in the EM iterations.

Simulation study

- Data set 1: We generate 100,000 observations from a mixture of three Gamma distributions with probabilities, shape and rate parameters depending on a three-dimensional variable $\boldsymbol{x} = (x_1, x_2, x_3) \in [-1, 1]^3$,
- Data set 2: We generate 100,000 observations from a mixture of three Log-normal distributions. The mixing probabilities are the same as in data set 1. The expected values and the variances of the three individual Log-normal distributions are chosen so that they are equal to the expected values and the variances of the three individual Gamma distributions in data set 1,
- For both data sets, we performed optimization of hyperparameters by considering a range of possible values for hyperparameters.

Simulation study - data set 1

Architecture	1					
Algorithm	EM_NB_3		EM_NB_4		EM_FN	
Epochs	25	100	25	100	25	100
Log-likelihood	-0.1591	-0.1589	-0.1581	-0.1580	-0.1573	-0.1578
Architecture	2					
Algorithm	EM_NB_3		EM_NB_4		EM_FN	
Epochs	25	100	25	100	25	100
Log-likelihood	-0.1588	-0.1586	-0.1578	-0.1585	-0.1573	-0.1575

Table: The maximum log-likelihood achieved on the validation set.

- We prefer the forward network algorithm, Architecture 2 and 25 epochs,
- We need ca. 13 minutes to achieve the highest log-likelihood on the validation set (100 iterations of the EM algorithm).

Simulation study - data set 1

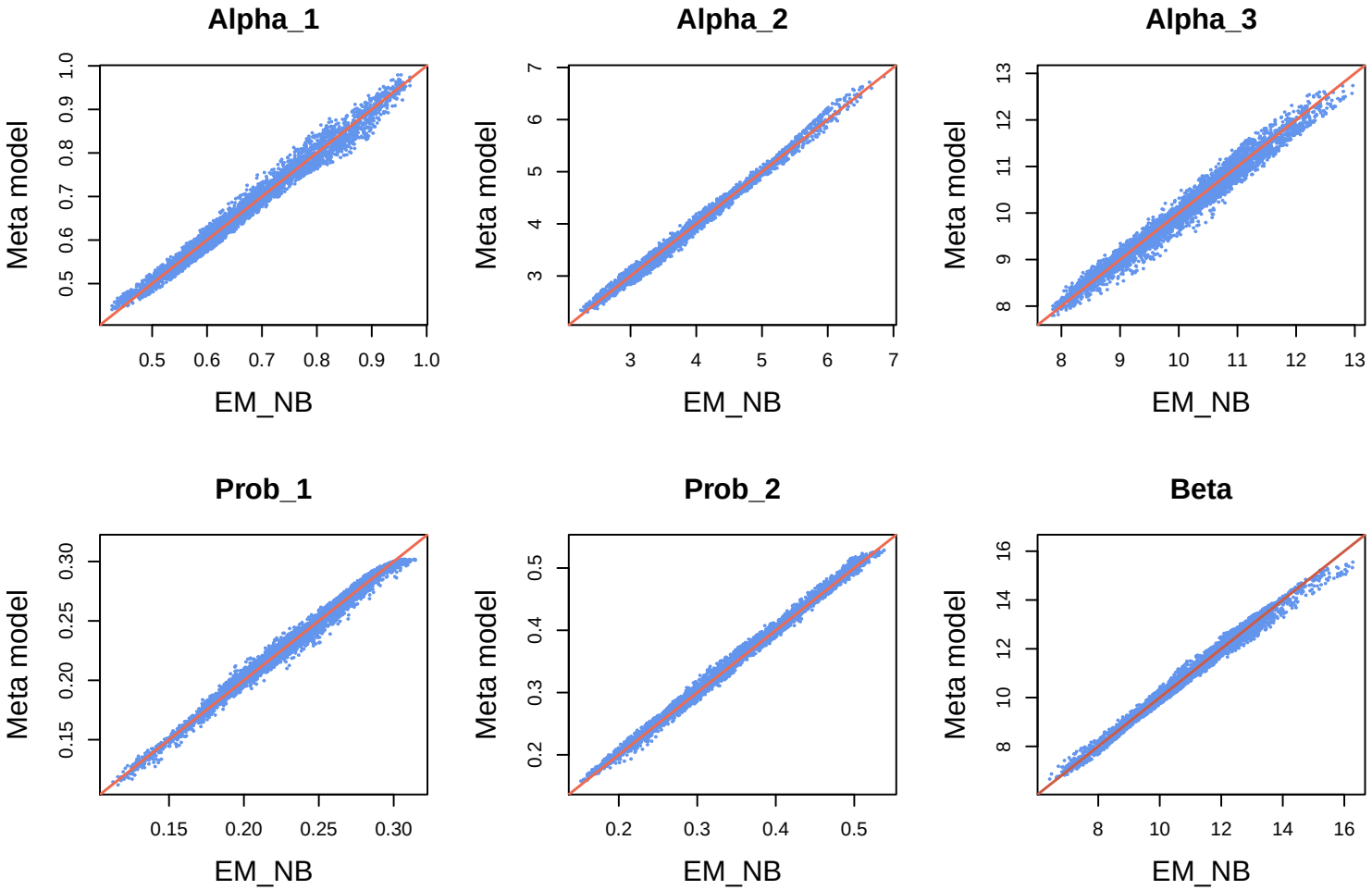

Figure: Validation of the estimates from the meta model vs. EM_NB_4.

Simulation study - data set 1

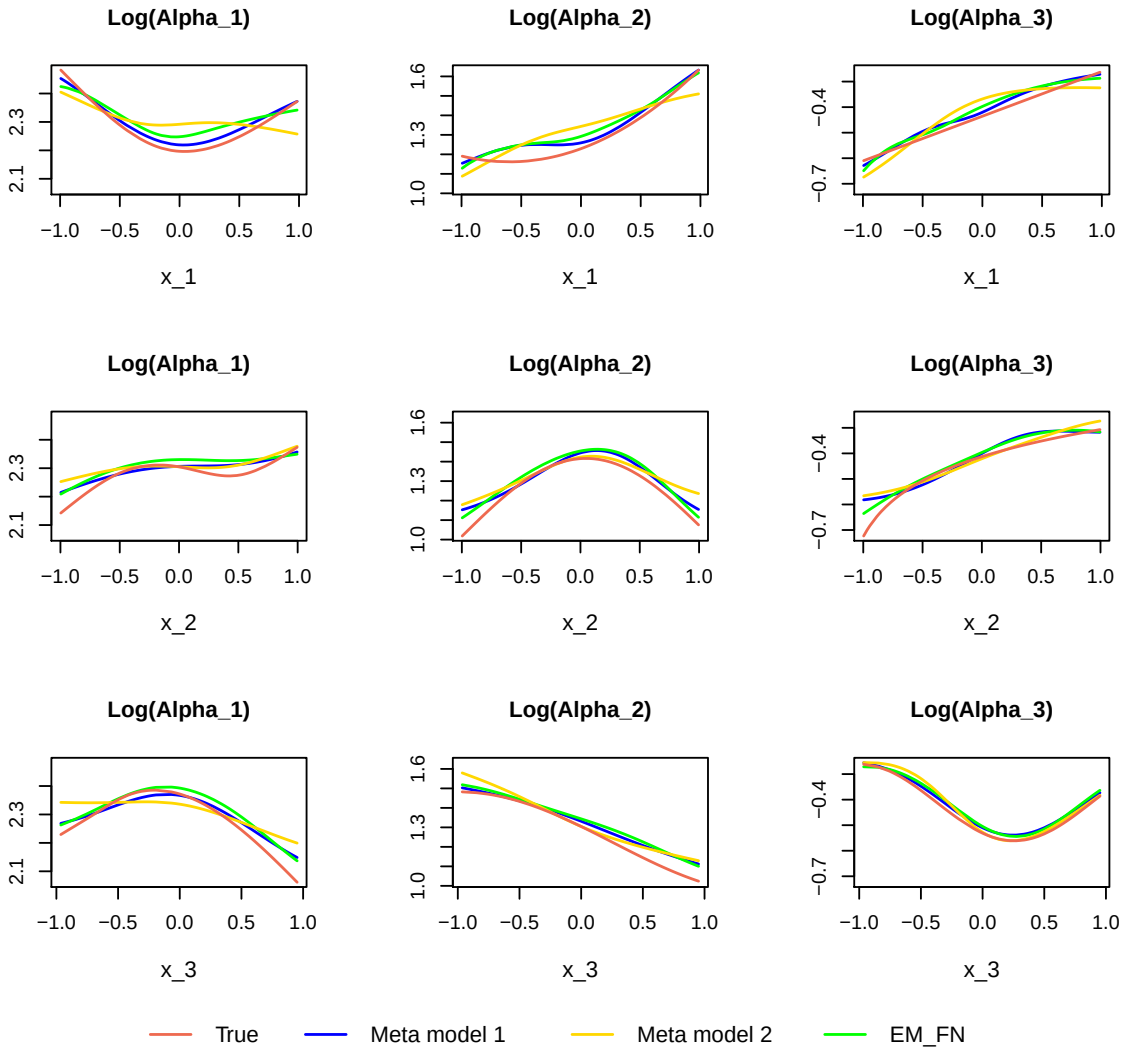

Figure: Results of the calibrations.

Simulation study - data set 1

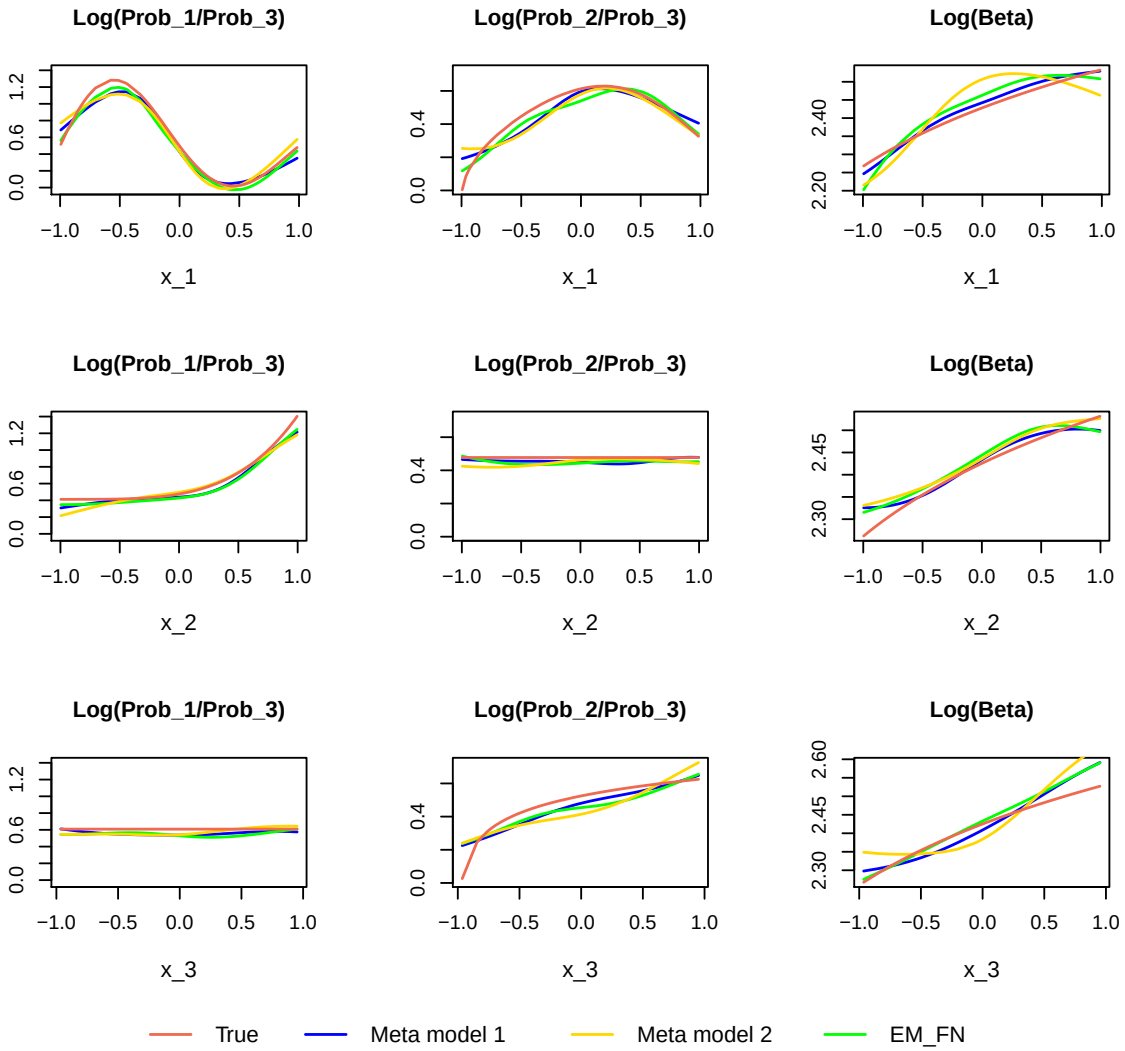

Figure: Results of the calibrations.

Simulation study - data set 2

- We investigate the robustness of our algorithms with respect to the tail of the underlying distribution,
- We expect that mixtures of Gamma distributions can approximate any response of any distribution arbitrarily well in the weak convergence sense if we allow for sufficient complexity in the mixture,
- We fit mixtures of Gamma distributions for $K = 1, 2, 3, \ldots$ with the EM forward network algorithm and investigate the log-likelihood on a validation set for different numbers of Gamma components.

No of mixtures	3	4	5	6	7	8	9
Log-likelihood	-0.1815	-0.1757	-0.1733	-0.1708	-0.1698	-0.1700	-0.1706

Table: The maximum log-likelihood achieved on the validation set.

- Remark: In a similar analysis for data set 1 we choose 3 mixtures of Gamma, but the results are very close also for 4 and 5 mixtures.

Simulation study - data set 2

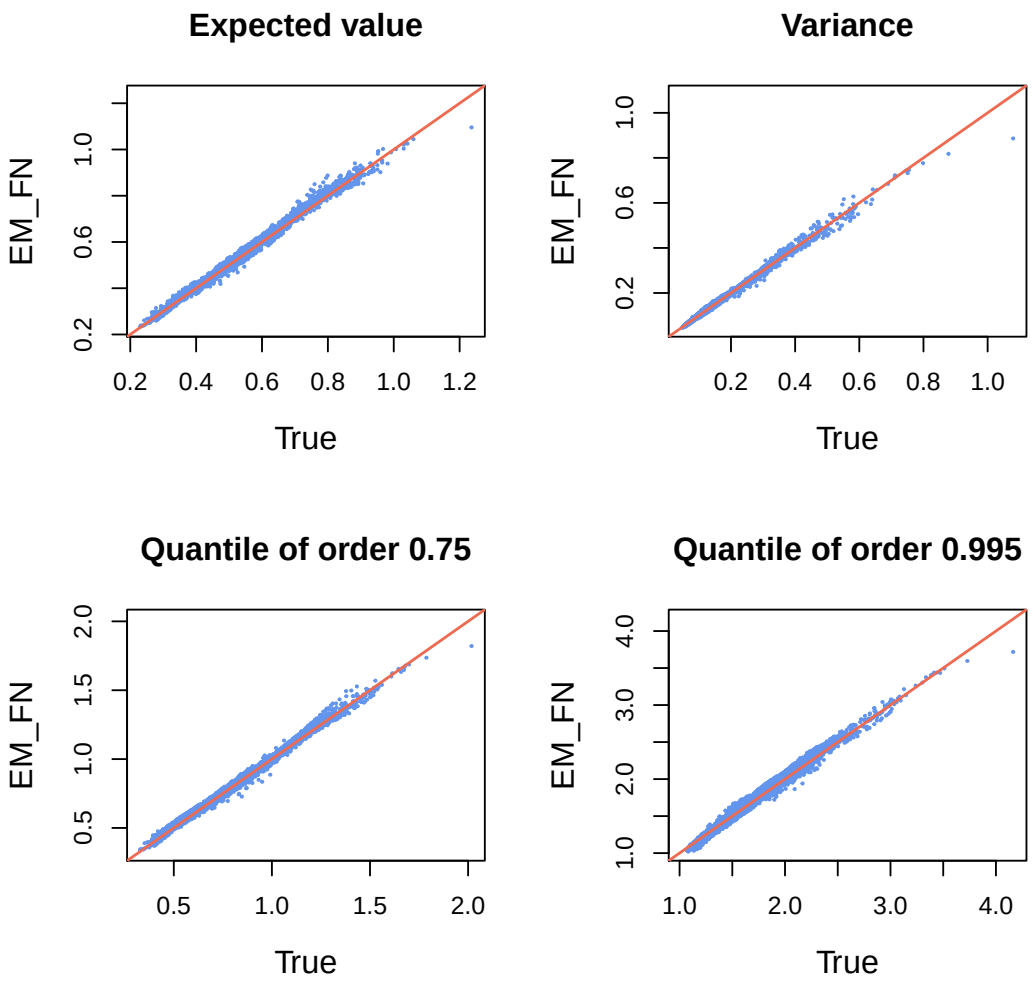

Figure: Validation of the true moments and the quantiles of the mixture of three Log-normal distributions vs. their predictions with the mixture of seven Gamma distributions.

Actuarial application - pricing with real data set

- We apply Gamma MDNs to model the claim amounts available from the data set freMPL1 contained in the R package CASdatasets,
- This data set includes 3,265 non-zero (multi-modal and heavy-tailed) claims from a motor personal line from a French insurer,
- For each observation in the data set, we have the claim amount and variables which characterize the observation. We have three continuous features, four binary features and eleven categorical features,
- Some categorical features are ordered categorical features and they can be implemented as categorical variables or continuous variables in the neural networks after ordinal encoding. We test both approaches. For categorical features, we apply the entity embedding technique,
- We run our EM network algorithms with hyperparameter optimization. We choose the mixture of seven Gamma distributions.

Actuarial application - pricing with real data set

Ordered categorical as categorical features					
Algorithm	EM_FN	EM_NB_1	EM_NB_2	EM_NB_3	EM_NB_4
Log-likelihood	-8.5386	-8.5481	-8.5498	-8.5017	-8.5927
Ordered categorical as continuous features					
Algorithm	EM_FN	EM_NB_1	EM_NB_2	EM_NB_3	EM_NB_4
Log-likelihood	-8.5291	-8.5475	-8.5422	-8.4989	-8.5922

Table: The maximum log-likelihood achieved on the validation set.

- We prefer the network boosting algorithm with Initializer 3, we only tested Architecture 2 and 25 epochs,
- We need ca. 2 minutes to achieve the highest log-likelihood on the validation set (50 iterations of the EM algorithm),
- The approach where we model the ordered categorical features as continuous features is slightly better than the approach where the variables are treated as pure categorical, but the results are very similar and additional runs are needed to confirm this hypothesis.

Actuarial application - pricing with real data set

- As the benchmark model, we fit a GAM with a Gamma response and the logarithmic link function. In this model the continuous (normalized) features are modelled with regression splines and the categorical (also ordered) features are modelled using dummy variables. No interactions are included,
- The first advantage of our Gamma MDN: the predictive power of the regression model, measured with the log-likelihood on the validation set, is increased from -8.7744 for the GAM to -8.5280 for the Gamma MDN (the meta model fitted to the last model from the EM NB),
- The second advantage of our Gamma MDN: The fit of the regression model is significantly improved. The fit of the GAM is poor, both in terms of the deviance function and the QQ-plot. The fit of our mixture of seven Gamma distributions is very good. The deviance function is almost perfect. The QQ-plot shows that the Gamma MDN is fitted accurately for the quantiles ranging from 5% up to 99.5%, and only low and extremely high quantiles are mis-estimated.

Actuarial application - pricing with real data set

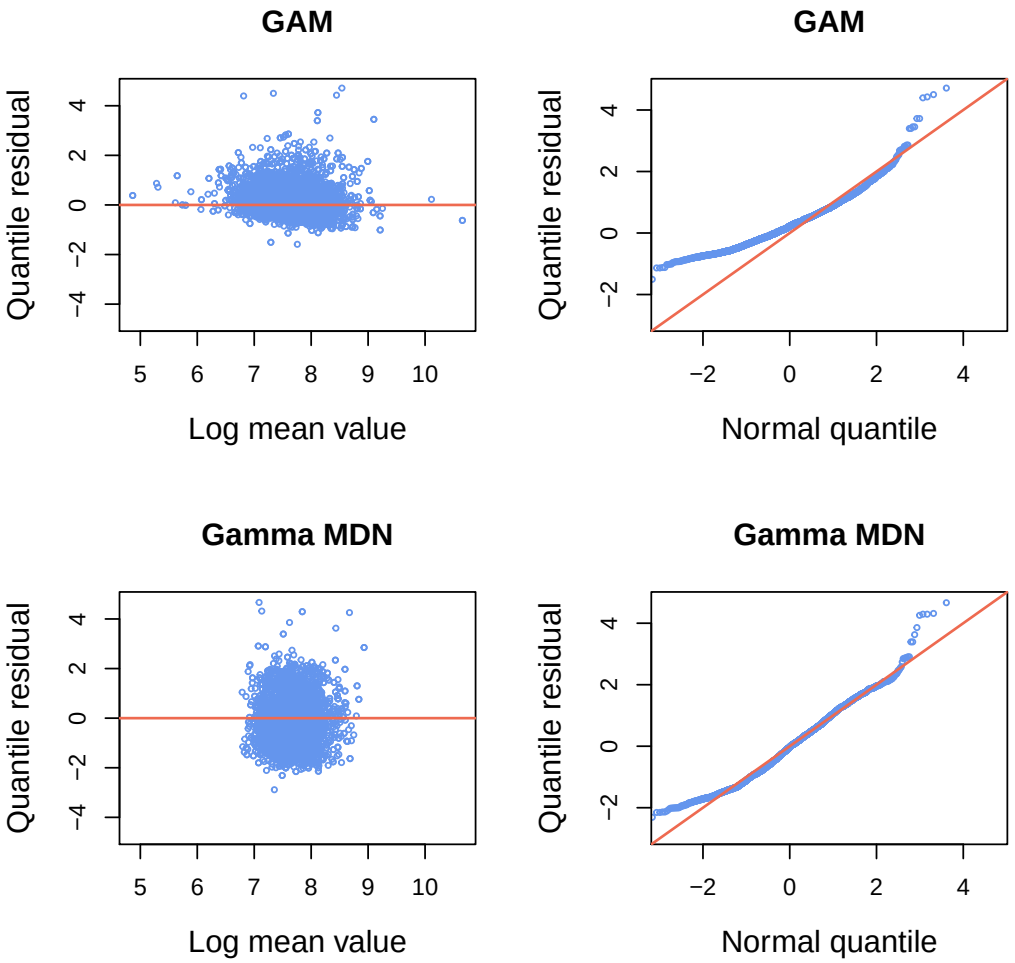

Figure: Results of the calibrations with the GAM and the meta model of Gamma MDN.

Gamma MDN - brute force vs EM iterations

What if we directly optimize the log-likelihood:

- The loss on the validation set is similar - data set 1,

BUT

- The loss on the validation set is significantly larger - data set 2 and freMPL1,
- The algorithm does not converge - the claims reserving problem.

- The paper is available: Ł. Delong, M.M. Lindholm, M.V. Wüthrich, 2021, Gamma Mixture Density Networks and their application to modelling insurance claim amounts, Insurance: Mathematics and Economics 101B, 2021, 240-261,
- The R codes are available at GitHub: https://github.com/LukaszDelong/GammaMDN.

Thank you very much.

Łukasz Delong
SGH Warsaw School of Economics
E-mail: lukasz.delong@sgh.waw.pl
Homepage: www.lukaszdelong.pl