Humboldt-Universität zu Berlin
School of Business and Economics
Ladislaus von Bortkiewicz Chair of Statistics

Deep Generalized Additive Regression Models for Location, Scale and Shape using TensorFlow

Master's Thesis submitted

to

Prof. Dr. Nadja Klein
Dr. David Rügamer

by

Kang Yang
(592371)

in partial fulfillment of the requirements for the degree of
Master of Science in Statistics

Berlin, March 02, 2020

Abstract

Deep Distributional Learning model, or Deep Generalized Additive Regression Models for Locations, Scale and Shape (DGAMLSS), is a flexible framework designed by Rügamer et al[1], where both structured additive and deep neural network parts are included into the additive predictor for distributional regression. In this paper, we employ a series of datasets separately simulated by different distributions, examine performance of deep GAMLSS, and compare it with other three state-of-the-art designs for GAMLSS, namely GAMLSS with maximum likelihood[2], BAMLSS[3] and gamboostLSS[4]. Such comparison is demonstrated from different aspects. Estimations on both linear and non-linear terms are studied and compared through different models to address their strength and weakness. DGAMLSS generally outperforms other models in respect of prediction accuracy, convergence speed and robustness under noises especially for complex distributional assumption, but fails to distinguish structure linear, non-linear estimator, and unstructured estimator.

Keywords: GAMLSS; DFNN; Variational Inference; Deep learning; Tensorflow probability.

Contents

1 Introduction **1**

2 GAMLSS and State-of-the-art Methods **4**

2.1 Generalized Additive Model and Structured Additive Regression Model 4

2.2 GAMLSS based on Penalized Maximum Likelihood 4

2.3 gamboostLSS 6

2.4 BAMLSS 6

3 DGAMLSS **7**

4 Non-parametric Estimator **11**

4.1 Splines & Smoothing Splines 11

4.2 B-splines & P-splines 12

4.3 Deep Neural Network 13

4.3.1 Deep Forward Neural Network 13

4.3.2 Other Types of ANN in Practice 15

5 Experimental Result **18**

5.1 Classical Distributions 18

5.1.1 Data Generating 19

5.1.2 Model Specification 24

5.1.3 Estimation Results 27

5.2 Mixture Distribution: Zero-inflated Poisson 43

6 Conclusion & Discussion **44**

7 Appendix **45**

List of Abbreviations

ANN: Artificial Neural Network

BAMLSS: Bayesian Additive Model for Location, Scale and Shape

CNN: Convolutional Neural Network

CUDA: Compute Unified Device Architecture

DFNN: Deep Forward Neural Network

DGAMLSS: Deep Generalized Additive Model for Location, Scale and Shape

gamboostLSS: Generalized Additive Model for Location, Scale and Shape based on boosting

GAMLSS: Generalized Additive Model for Location, Scale and Shape

GAM: Generalized Additive Model

GCN: Graph Convolutional Network

GLM: Generalized Linear Model

MISE: Mean Integrated Squared Error

MSE: Mean Squared Error

Relu: Rectified linear unit

RNN: Reccurent Neural Network

STAR: Structured Additive Regression Model

ZIP: Zero-inflated Poisson (distribution)

List of Figures

4.1 B-spline of degree one 12

4.2 B-spline of degree three 12

4.3 An example for multiple-layer Deep Forward Neural Network with four hidden layer[5] . 14

4.4 Rectifier (Relu) in comparison with Softplus activation function<$f(x) = ln(1 + e^x)$ >:[6] 14

4.5 Multi-layer neural networks and back-propagation[7]: Part a shows how a three-layer neural network with two neurons in both input and hidden layer simulate non-linear boundary for two-fold classification issue; Part b is the chain rule in calculus used of composite of two functions; Part c depicts the information pipeline within a neural network; Part d gives details in back-propagation algorithm based on chain rule 15

4.6 An Example on CNN: LeNet-5[8] (from left to right: the input image, first convolutional layer, first pooling layer, second convolutional layer, second pooling layer, two full-connection layers and output layer) 16

4.7 The Stretch of A Recurrent Neural Network[7](left: elements of inputs x are sequentially processed by neuron s with status vector W from last run; right: such procedure could be viewed as a sequence of neurons lined up, and each neuron x_t process both input x_t and status vector from the previous neuron 17

5.1 Orthogonalization on $f_{1,x}(x)$ (from left to right: the original function, subtraction of linear trend, subtraction of linear trend of the intercept) 20

5.2 Orthogonalization on $f_{1,z}(z)$(from left to right: the original function, subtraction of linear trend, subtraction of linear trend of the intercept) 20

5.3 Orthogonalization on $f_{1,v}(v)$(from left to right: the original function, subtraction of linear trend, subtraction of linear trend of the intercept) 21

5.4 Orthogonalization on $f_{2,x}(x)$(from left to right: the original function, subtraction of linear trend, subtraction of linear trend of the intercept) 21

5.5 Orthogonalization on $f_{2,z}(z)$(from left to right: the original function, subtraction of linear trend, subtraction of linear trend of the intercept) 22

5.6 Orthogonalization on $f_{2,v}(v)$(from left to right: the original function, subtraction of linear trend, subtraction of linear trend of the intercept) 22

5.7 True values of parameters depending on x and z 27

5.8 Estimation of parameters based on x an z by different models 28

5.9 Linear Estimation (Poisson distribution, two-dimensional case) by each model in 100 iteration (figure on the left: intercept in parameter 1; figure on the right: linear coefficient in parameter 1; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression) 29
5.10 Averaged non-linear estimation by each model (Poisson distribution, two-dimensional case) 30
5.11 Variational inference on estimation of non-linear function by Deep Distributional Learning model (Poisson distribution, two-dimensional case) 30
5.12 Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i) - f(x_i)]^2)$> of non-linear estimation by each model (Poisson distribution, two-dimensional case) 31
5.13 MSE of estimated parameters on test dataset (Poisson distribution, two-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS) 32
5.14 Linear Estimation (with noise, Poisson distribution, two-dimensional case) by each model in 100 iteration (figure on the left: intercept in parameter 1; figure on the right: linear coefficient in parameter 1; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression) 33
5.15 Averaged non-linear estimation by each model (with noises, Poisson distribution, two-dimensional case) 33
5.16 Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i) - f(x_i)]^2)$> of non-linear estimation by each model (with noise, Poisson distribution, two-dimensional case) 34
5.17 Variational inference on estimation of non-linear function by Deep Distributional Learning model (with noises, Poisson distribution, two-dimensional case) 34
5.18 MSE of estimated parameters on test dataset (with noises, Poisson distribution, two-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS) 35
5.19 Variational inference on estimation of non-linear function by Deep Distributional Learning model (Poisson distribution, three-dimensional case) 35
5.20 MSE of estimated parameters on test dataset (Poisson distribution, three-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS) 36
5.21 Linear Estimation (Poisson distribution, three-dimensional case) by each model in 100 iteration (figure on the left: intercept in parameter 1; figure on the right: linear coefficient in parameter 1; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression) 37
5.22 Averaged non-linear estimation by each model (Poisson distribution, three-dimensional case) 37

5.23 Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i)-f(x_i)]^2)$> of non-linear estimation by each model (Poisson distribution, three-dimensional case) 38
5.24 MSE of estimated parameters on test dataset (Normal distribution, two-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS) 39
5.25 Linear Estimation (Normal distribution, two-dimensional case) by each model in 100 iteration (figure on the top-left: intercept in parameter 1; figure on the top-right: linear coefficient in parameter 1; figure on the bottom-left: intercept in parameter 2; figure on the bottom-right: linear coefficient in parameter 2; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression) 40
5.26 Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i)-f(x_i)]^2)$> of non-linear estimation by each model (Normal distribution, two-dimensional case) 41
5.27 Averaged non-linear estimation by each model (Normal distribution, two-dimensional case) 42
7.1 MSE of estimated parameters on test dataset (with noises, Normal distribution, two-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS) 45
7.2 Linear Estimation (with noises, Normal distribution, two-dimensional case) by each model in 100 iteration (figure on the top-left: intercept in parameter 1; figure on the top-right: linear coefficient in parameter 1; figure on the bottom-left: intercept in parameter 2; figure on the bottom-right: linear coefficient in parameter 2; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression) 46
7.3 Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i)-f(x_i)]^2)$> of non-linear estimation by each model (with noises, Normal distribution, two-dimensional case) 47
7.4 Averaged non-linear estimation by each model (with noises, Normal distribution, two-dimensional case) 48
7.5 MSE of estimated parameters on test dataset (Normal distribution, three-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS) 49
7.6 Linear Estimation (Normal distribution, three-dimensional case) by each model in 100 iteration (figure on the top-left: intercept in parameter 1; figure on the top-right: linear coefficient in parameter 1; figure on the bottom-left: intercept in parameter 2; figure on the bottom-right: linear coefficient in parameter 2; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression) 50
7.7 Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i)-f(x_i)]^2)$> of non-linear estimation by each model (Normal distribution, three-dimensional case) 51

7.8 Averaged non-linear estimation by each model (Normal distribution, three-dimensional case) . . . 52
7.9 MSE of estimated parameters on test dataset (ZIP distribution, two-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS) . . . 53
7.10 Linear Estimation (ZIP distribution, two-dimensional case) by each model in 100 iteration (figure on the top-left: intercept in parameter 1; figure on the top-right: linear coefficient in parameter 1; figure on the bottom-left: intercept in parameter 2; figure on the bottom-right: linear coefficient in parameter 2; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression) . . . 54
7.11 Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i) - f(x_i)]^2)$> of non-linear estimation by each model (ZIP distribution, two-dimensional case) . . . 55
7.12 Averaged non-linear estimation by each model (ZIP distribution, two-dimensional case) . 56
7.13 MSE of estimated parameters on test dataset (ZIP distribution, three-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS) . . . 57
7.14 Linear Estimation (ZIP distribution, three-dimensional case) by each model in 100 iteration (figure on the top-left: intercept in parameter 1; figure on the top-right: linear coefficient in parameter 1; figure on the bottom-left: intercept in parameter 2; figure on the bottom-right: linear coefficient in parameter 2; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression) . . . 58
7.15 Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i) - f(x_i)]^2)$> of non-linear estimation by each model (ZIP distribution, three-dimensional case) . . . 59
7.16 Averaged non-linear estimation by each model (ZIP distribution, three-dimensional case) 60

List of Tables

5.1 List of non-linear functions in data-generating process . . . 19
5.2 Distribution functions and links for normal, lognormal, and logistic distributions . . . 19
5.3 MSE(log) of estimated linear coefficients by each model (Poisson distribution, two-dimensional case) . . . 29
5.4 MSE(log) of estimated linear coefficients by each model (with noises, Poisson distribution, two-dimensional case) . . . 32
5.5 MSE(log) of estimated linear coefficients by each model (Poisson distribution, three-dimensional case) . . . 36
5.6 MSE(log) of estimated linear coefficients by each model (Normal distribution, two-dimensional case) . . . 39

1 Introduction

Generalized Additive Model (GAM), firstly introduced by Trevor Hastie and Robert Tibshiani(1984) [9], is one of the most remarkable model in the scope of statistical regression model. It blends the properties of Generalized Linear Models [10] with additive functions. In GAM, statisticians take the assumption that response variable y follows exponential family distribution with two parameters mean μ and dispersion ϕ, while skewness (or third moment) and kurtosis (or fourth moment) are implicitly modeled and dependent on μ and ϕ.

Rigby and Stasinopoulos (2005) [11] have developed Generalized additive models for location, scale, and shape (GAMLSS) as a modern distribution-based approach in semi-parametric statistics, extending the scope of GAM to more flexible distributional cases. In contrast to GAM, a more general distribution family is used in GAMLSS as the alternative to the exponential family assumption. In this model all parameters (usually up to four parameters) of the assumed conditional distribution, involving not only high order moments but other parameters such as zero inflation probability for zero-inflated model [12] or degree of freedom of the Chi-square distribution, could be modeled as linear or additive nonlinear functions of explanatory variables and/or random effect terms with associated link functions. Thereby, a tremendous amount of distributional assumptions, including truncated, censored or mixture distributions, is available to be deployed as conditional probability of data where exponential family such as binomial, normal and Poisson distributional assumptions may fail.

Several algorithmic procedures have been proposed to fit a GAMLSS model in last decades. Based on the Newton-Raphson or Fisher scoring algorithm, Rigby and Stasinopoulos (2007)[2] have invented the RS and CG algorithms, respectively rooted from Newton-Raphson and Fisher Scoring algorithms, for the (penalized) Maximum Likelihood Estimation. These two methods are implemented in R-package "gamlss" and explicitly explained in their original paper.

GAMLSS has developed into a powerful statistical model for regression after being published, been exploited by researches around the world, and expanded to various applications. According to the description on the website[1], the model has been employed by several international organizations, namely the World Health Organization (WHO), the International Monetary Fund (IMF), the European Bank and the Bank of England, and individuals for researches in various fields including marine research, medicine, meteorology, rainfalls, vaccines, food consumption, biology, biosciences, energy economic, genomics, finance, fisheries, growth curves estimation, actuarial science.

However, such algorithm is approved to be less efficient in high-dimensional case, and inadequate for the aim of variable selection to avoid overfitting as well. Regarding to this issue, Mayr Andreas et al(2012)[4] proposed a boosting-based method named "gamboostLSS" in form of the compound of

[1] https://www.gamlss.com

GAMLSS and gradient boosting algorithm[13], where the component-wise gradient descent algorithm[14] is used for regression and estimation with intrinsic variable selection mechanism. Hence, this algorithm is a powerful tool to tackle high-dimensional datasets in which effects to be estimated outnumber observations, and to provide a sparse estimation in regards to all GAMLSS parameters. By designed early stopping strategy, gamboostLSS contains a built-in technic for regularization on the model. Another essential characteristic of this model is the flexibility in selection of the base learners such as linear effects, non-linear effects, spatial effects and random effects. Lastly, gamboostLSS can be utilized for more complex distributions, as the number of distribution parameters is allowed to be more than four.

BayesX[15] is the independent software written in C++ with a R-interface **R2BayesX**[16] supporting analyze on majority of Structured Additive Regression (STAR) model including Generalized Additive Model (GAM). Subsequently, Nadja, Klein et al[17] developed the Bayesian Generalized Additive Model for Location, Scale and Shape (BAMLSS) and R-package "BAMLSS"[16] constructed upon standalone statistical software packages including BayesX and JAGS. The "BAMLSS" package allows for estimation of not only classical regression type including GAM but flexible regression model as GAMLSS. A proposal method called iteratively weighted least square (IWLS)[18] approximation could be implemented in MCMC sampler for Bayesian inference in case of complex structure of the likelihoods. Literally studies reveal that the BAMLSS model could outperform the GAMLSS in many respect (unbiasedness, smaller MSE and fast coverage rate), and yield reliable credible intervals when the asymptotic likelihood theory fails.

From the beginning of the 21st century, the rise of Artificial Neural Network (ANN)[19] is a bombshell to the field of Artificial Intelligence and statistical analysis. Since then, the analysis of large-scale dataset is inclined to be remarkably approachable and efficient. The flexible architecture of Deep Neural Network (multiple-layer ANN)[7], or so-called Deep Learning, allows more sophisticated designs in neurons and connections aimed on special tasks. Deep learning provides more general models for analysis on unstructured dataset, high-dimensional cases or sequential datasets. However, the estimation process with deep neural network is regarded as a Black-Box[20] for researchers and less interpretable than other statistical models. Academicians and developers around the world have been dedicated to facilitate the interpretability and robustness of the Deep Neural Network.

Therefore, the last model presented in this paper is called "Deep Generalized Additive Regression Models for Location, Scale and Shape" (DGAMLSS), or Deep Distributional Learning model and "deepregression" in this paper (Rügamer et al[1]). The basic idea is to combine generalized additive model with deep neural network. DGAMLSS incorporates smoothness penalties for separation of structured and deep neural network predictor, and offers different options such as mean-filed variational inference to assess estimation uncertainty.

With all of these techniques developed for estimation of GAMLSS, researchers are able to choose from various options when tackling complex situations for regression and estimation. However, there is a lack of comparison within those models regarding to different aspects, videlicet, the estimation accuracy under different assumption, the efficiency in computation, and likewise trade-off between variance and bias. Analysis in DGAMLSS is needed for further development in respect to its weakness.

The rest of this article is organized as followed: In Chapter 2.1 we briefly describe GAM and STAR. Then we present first three models deployed in our simulation work with necessary mathematical details, namely GAMLSS with (penalized) maximum likelihood, gamboostLSS, and BAMLSS. Thereupon we elaborate a comprehensive introduction of DGAMLSS in Chapter 3, and description on non-parametric estimators deployed in our model in Chapter 4, including P-splines and Deep Neural Network. Chapter 5 develops our simulation work constructed on various datasets under different distributions and comparison of estimations. The Final Chapter will summarize findings and comments on future development.

2 GAMLSS and State-of-the-art Methods

2.1 Generalized Additive Model and Structured Additive Regression Model

Initially, an introduction to Generalized Linear Model (GLM) or Generalized Additive Model (GAM) would be demonstrated in brief. In GLM or GAM, the distribution of response variable y given covariates x and unknown parameters $\nu = (\theta, \phi)$ is in form of exponential family:

$$p(y|x, \nu) = exp(\frac{y\theta - b(\theta)}{\phi})c(y, \theta).$$

The mean $\mu = E(y|x, \nu)$ is linked to a linear predictor η via link function g by $\mu = g^{-1}(\eta)$, and η is modeled by either a linear predictor $\eta = x^T\gamma$ or a semi-parametric additive predictor $\eta = f_1(x_1) + ... + f_p(x_p) + x^{'}\beta$, where $x^{'}\beta$ represents the strictly parametric part of the predictor and $f_i(x_i)$ is smooth function of continuous variables such as splines.

A group of statisticians have extended this model into a more generalized form, Structured Additive Regression Model (STAR)[18][21]. It is a group of model classes consisting of *"generalized additive model, generalized mixed model, varying coefficient models and geographically weighted regression"*(Nikolaus Umlauf et al 2012, p.6)[16]. The non-parametric smoothing function f_i could be in form of various effects depending on the model design, including nonlinear functions depending on continuous covariates, two-dimensional functions, spatially correlated effects, and (spatial) varying coefficients.

There are numbers of techniques used to dispose of overfitting caused by large number for non-parametric smoothers, including squared r-th order differences in frequentist approach and r-th order random walks with Gaussian errors in Bayesian approach.

2.2 GAMLSS based on Penalized Maximum Likelihood

The limit of GAM or STAR results from the inflexible distributional assumption, and an alternative to them is the Generalized Additive Model for Location, Scale and Shape. Generalized Additive Model for Location, Scale and Shape is the extension of Generalized Linear/Additive Model, which holds the assumption on exponential family distribution, to more flexible model under general distribution assumption. All distribution parameters of the conditional distribution of data, including degree of freedom or zero-inflation probability, are able to be modeled as parametric or additive smooth function of regressors with a link function for each parameter. More generally, the model allows various random effect terms to be contained in the function including *"cubic splines or smoothness priors, random-walk terms and many random-effects terms (including terms for simple overdispersion, longitudinal random effects, random-coefficient models, multilevel hierarchical models and crossed and spatial random effects"*(R. A. Rigby and D. M. Stasinopoulos 2005, p.2)[11].

Assuming that we have totally p parameters $\theta^T = (\theta_1, \theta_2, ..., \theta_p)$ for the conditional distribution function of y: $f(y|\theta)$. All of those parameters are hypothetically modeled by additive functions through monotonic link functions $g_k()$ where $k \in \{1, ..., p\}$:

$$g_k(\theta_k) = \eta_k = \mathcal{X}_k \, \beta_k + \sum_{j=1}^{J^k} \mathcal{Z}_{jk} \, \gamma_{jk}$$

where $\mathcal{X}$ and $\mathcal{Z}$ are designed matrices respectively for fixed effects and random effects, β is parameter vector, and γ_{jk} is random variable under Gaussian distribution $\mathcal{N}(0, \mathcal{G}_{jk}^{-1}(\lambda_{jk}))$. λ_{jk} is a hyper-parameter in model. In some cases, $_{jk}$ could be modeled in form of $f_{jk}(x_j)$ as non-parametric function of explanatory variables x_j.

Usually, the first parameter θ_1 is characterized as mean μ and the second one θ_2 is called scale noted as σ, where others are named shape parameters. Generally, the number of parameters to be estimated for an arbitrary distribution function could no more than four and that would cover a large majority of issues encountered in statistical analysis.

There are basically two major parts in the model: one is parametric term and the other is additive term. The parametric part contains linear and interaction terms for explanatory variables and factors, polynomials, fractional polynomials, and piecewise polynomials with fixed knots. The additive terms are non-parametric smoothing functions involving cubic smoothing splines, parameter-driven time series terms and smoothness priors, penalized splines, varying-coefficient terms, and spatial (covariate) random-effect terms.

Two algorithms are implemented in combination in R-package "gamlss", namely the RS algorithm and the CG algorithm. The RS algorithm is a generalization of the algorithm used by Rigby and Stasinopoulos (1996)[22][23] for fitting Mean and Dispersion Additive Models, and the CG algorithm is a generalization of the Cole and Green (1992)[24] algorithm. The later employs the first, the second and cross derivatives of the likelihood function with respect to the distribution parameters θ. When the parameters are information orthogonal and therefore the expectations of the cross derivatives of the likelihood function are zero, the simpler RS algorithm is the optimal option to speed up convergence of the model. A combination of these two algorithms is adopted in the R-package "GAMLSS" and they are proved to be stable and efficient with simple starting values of parameters.

Lastly, they proposed to use a more generalized version of the AIC, defined as:

$$GAIC(a) = -2 \sum_{i=1}^{n} log\{f_{dens}(y_i|\hat{\theta}_i)\} + a \, df$$

to penalize on overfitting, where df denotes the total effective degrees of freedom used in the model. A standard choice of a is between two and four. When $a = 2$, the criterion minimizes the Kullback-Leibler discrepancy to access to the optimal model (Andreas Mayr, 2012, chapter 2.2)[4] .

2.3 gamboostLSS

The gamboostLSS model is developed by Mayr Andreas et al[4] to address the issue on variable selection by GAMLLS. Statisticians have found out that, the generalized Akaike Information Criterion used by GAMLSS for variable selection would lead to inherited problems associated to classical AIC especially in case of a large number of covariates. Such method suffers from instability in estimation[25]. Besides, a number of non-informative variables may be added into the final result[22]. Lastly, such GAMLSS fitting produces inacurate result in distinguishing linear and non-linear effects[26]. In contrast, the new algorithm based on the boosting technique for estimation and variable selection would mitigate those problems. Component-wise gradient boosting approach could likewise be integrated into gamboostLSS for sparsity. Hence, gamboostLSS model is more efficient than GAMLSS with penalized maximum likelihood in high-dimensional cases.

Boosting could be used as a technique for fitting generalized additive model since it can be interpreted as a gradient descent algorithm in functional space and therefore relates to forward stage-wise additive modeling. In practice, the aim is to minimize the experimental risk:

$$arg\,min_{\,\eta} \sum_{i=1}^{n} \frac{\rho(y_i,\eta(x_i))}{n}$$

with a specified loss function $\rho(.)$ by a stepwise descent of the loss function's gradient. The boosting algorithm iteratively fits the negative partial derivatives of the empirical loss function with base learner function $f(.)$, which is typically in form of trees, linear models or penalized regression splines. In every step, the current version of η is updated additively by a step length *sl* to approach to a global minimum.

Furthermore, to model the distribution parameters of the conditional density $f_{dens}(y|\mu,\sigma,\nu,\tau)$ where (μ,σ,ν,τ) is $(\theta_1,\theta_2,\theta_3,\theta_4)$, we attempt to minimize the empirical risk with arbitrary loss function ρ:

$$arg\,min_{\,\eta} \sum_{i=1}^{n} \frac{\rho(y_i,\eta_\mu(x_i),\eta_\sigma(x_i),\eta_\nu(x_i),\eta_\tau(x_i))}{n}$$

Then we iteratively fit the negative partial derivatives of the empirical loss function

$$\mathbf{u}_k^{[m-1]} = (-\frac{\partial}{\partial\eta_{\theta_k}}\rho(y_i,\eta_i))_{i=1,\ldots,n},$$

where $\eta_i = (\eta_\mu^{[m-1]}(x_i), \eta_\sigma^{[m-1]}(x_i), \eta_\nu^{[m-1]}(x_i), \eta_\tau^{[m-1]}(x_i))$, with base-learner $f_{j,k^*}^{[m-1]}(x_i)$ for one parameter η_{θ_k} at a time in scale of step length *sl* until $m > m_{stop,k}$.

2.4 BAMLSS

BAMLSS is a Bayesian approach developed by Nadja Klein et al[27][17] to fit GAMLSS, where Bayesian inference is obtained through MCMC based on iteratively weight least squares proposal (IWLS). Assuming that, for each predictor we have a structured additive semi-parametric estimation:

$$\eta_k = \beta_0^k + x\beta^k + f_1(x) + f_2(x) + ... + f_{J^k}(x),$$

where $\eta = g(\theta)$ is the estimator for parameter θ via link function $g(.)$ and x is the observation. Fundamentally, $f_j(x))$ is approximated through the structured additive regression function in terms of basis as:

$$f_j(x_i) = \sum_{d=1}^{D_j} \gamma_{j,d} \cdot f_{j,d}(x).$$

The BAMLSS assumes the prior probability to each coefficient vector $\gamma_j = (\gamma_{j,1}, \gamma_{j,2}, ..., \gamma_{j,D_j})^{'}$ be in form of multivariate Gaussian prior distribution

$$p(\gamma_j|\tau_j^2) \propto (\frac{1}{\tau_j^2})^{\frac{rk(K_j)}{2}} exp(-\frac{1}{2\tau_j^2}\gamma_j^{'} K_j \gamma_j),$$

and the smoothing variance τ_j^2 under inverse gamma distribution $IG(a_j, b_j)$ with small values for the hyper-parameters a_j and b_j.

For the iteratively weight least squares proposal approach, let $l(\eta)$ be the log-likelihood which leads to

$$log(p(\gamma_j|.) \propto l(\eta) - \frac{1}{2\tau_j^2}\gamma_j^{'} K_j \gamma_j.$$

From Taylor expansion we have

$$\frac{\partial l^{(t)}}{\partial \eta_i} - \frac{\partial^2 l^{(t)}}{\partial \eta_i^2} \cdot (\eta_i^{(t+1)} - \eta_i^{(t)}) = 0$$

with iteration index t. This implies to the working model:

$$z^{(t)} \sim N(\eta^{(t)}, (W^{(t)})^{-1}).$$

in which $z = \eta + W^{-1}v$ is the working observations with expectation η.

Note that, $v = \partial l/\partial \eta$ is the score vector and W is the diagonal working weight matrix based on a Fisher-scoring approximation with $w_i = E(-\partial^2 l/\partial \eta_i^2)$. By the multivariate Gaussian priors for γ_j, we obtain the Gaussian proposal densities $\gamma_j \sim N(\mu_j, P_j^{-1})$ with $\mu_j = P_j^{-1} Z_j^{'} W(z - \eta_{-j})$ and $P_j = Z_j^{'} W Z_j + \frac{1}{\tau_j^2} K_j$ where $\eta_{-j} = \eta - Z_j \gamma_j$. The smoothing parameter τ_j^2 could be updated through a Gibbs sampler with $a_j^{'} = \frac{rk(K)}{2} + a_j$ and $b_j^{'} = \frac{1}{2}\gamma_j^{'} K \gamma_j + b_j$.

3 DGAMLSS

In the field of machine learning, methods such as bagging and boosting where uniform base leaners are combined into an aggregated estimator are called homogeneous ensemble. In contrast, another widely used ensemble method called heterogenous ensemble is model stacking. In this case, as demonstrated by Deep Distributional Learning model, outputs of various trained base leaners are taken as input and

artfully commixed to obtain an enhanced prediction. Usually a rather sophisticated approach named k-fold cross validation is preferred by training the stacking model. Analogously, Cheng Heng-Tze et al proposed a model called "Wide & Deep Learning"[28] where wide linear model and deep neural network are combined through weight matrices for each part, jointly trained and simultaneously optimized.

Deep Distributional Learning model is the project on "statistical deep learning" combining generalized linear and/or additive model for location, scale and shape (GAMLSS) with multi-layer neural network via the architecture of an orthogonalization machanism. Assuming that we are interested in both structured and unstructured effects by covariates, ideally statistical model GAMLSS is efficient for analysis on structured effects, and deep neural network is helpful to capture unstructured effects or noises. For the statistical part, various smoothers in R-package "mgcv"[29] could be adopted by model including cubic splines, B-splines, P-splines. On top of that, the deep neural network is based on powerful open source platform tensorflow[30] and built-on library tensorflow-probability[31]. Tensorflow-probability allows users to combine deep learning with probabilistic models, build so-called "deep probabilistic models"[32] with probabilistic layers and assess uncertainty through Mento Carlo, MCMC, variational inference and stochastic optimization methods.

Literally, the structured additive predictor and deep neural network are independently trained. Afterwards, both structured linear and non-linear input types are modeled as a single unit hidden layer with linear activation functions under different regularization. The structured model part and the unstructured deep neural network (DNN) predictor are combined as the direct sum before transmitted to distribution layers.

$$\eta_k = f_0(.) + f_1(.) + ... + f_{J^k}(.) + d_1(.) + ... + d_{g^k}(.),$$

Where $f_0(x)$ is the linear function, $f_1, ..., f_{J^k}(.)$ are non-linear functions, and $d_1(.), ..., d_{g^k}(.)$ are neural network estimators. In respect to the issue of identification when inputs overlap in both structured and unstructured part, a linear orthogonalization operation yielding centered non-linear functions is deployed. In case of overlap between linear effects and deep learning effects:

$$\tilde{Z} = (\mathbb{I}_n - \mathbb{P}_X)\hat{Z},$$

where $\hat{Z}$ is the output of unstructured part, X is features of structured linear part and $\mathbb{P}_X = X(X^TX)^{-1}X^T$. When structured linear and structured non-linear parts are both present in the issue of identifiability, this orthogonalization operation is firstly used to ensure identifiability between the linear and non-linear structured parts before combination, and thereafter the same operation for separation of the whole structured from the unstructured deep learning predictor. Lastly, both adjusted predictors are summed up as parameters of distributional layer. To access the uncertainty of estimation, one choice is to build a variational inference layer at the end and account for the epistemic uncertainty by staying close to a *prior*.

Besides its flexibility and efficiency, DGAMLSS provided several advantages by data analysis:

- In high-dimensional setting, both lasso and ridge regression could be applied into the model. In addition with P-splines regularized on the number of knots used in the structured additive part, GAMLSS provides multiple choices on trade-off between bias and variance by estimation.
- With an inbuilt model selection mechanism, simple linear and non-linear effects of features are technically separated from the part of the neural network, where the later is usually lack of interpretability and redeemed as the "black-box"[20]. Therefore, Deep Distributional Learning model may shed light onto the approach to the *interpretable machine learning*[33].
- In Tensorflow-probability, large number of distributions are available as distribution layer including classical univariate, multivariate and mixture distributions. Distribution layers such as mixtures of same distribution families and General Mixture Distributions release assumptions on single distribution function and enable analysis on flexible model definition such as zero-inflated data.
- Tensorflow provides instant solutions on large-scale dataset by hardware acceleration, massive parallelization and distributed computation by GPU (supported by CUDA of $NVIDIA^{TM}$).

Lastly, variational inference[34] (or variational Bayes) is one of most powerful characteristics introduced by tensorflow-probability into the model. Compared with Monte Carlo and MCMC, variational Bayes could be more efficient in high-dimensional casse or the number of observations is not reasonable large. For notation, θ is the unobserved (or so-called latent) variables related to observations y. Through approximating the *posterior*

$$p(\theta|y) \propto p(\theta, y) \cdot p(\theta) \equiv h(\theta)$$

with a tractable $q_\lambda(\theta)$, one can find the most proper calibrate or hyper-parameters λ that minimizing the Kullback-Leibler divergence:

$$KLD(q||p) = \int q_\lambda(\theta) \cdot log(\tfrac{q_\lambda(\theta)}{p(\theta|y)}) d\theta$$

or equivalently maximizing the variational lower bound:

$$\mathscr{L}(\lambda) = E_q[log(h(\theta)) - log(q_\lambda(\theta))].$$

One possible option for the variational density is the mean filed variational Bayes. For a number of m unknown variables $\theta = (\theta_1, ..., \theta_m)$, the variational family is assumed to factorize:

$$p(\theta|y) \approx q(\theta) = q(\theta_1, ..., \theta_m) = \prod_{j=1}^{m} q_j(\theta_j).$$

From mathematical deduction we have:

$$q_j(\theta_j) = \frac{e^{E_{m|m\neq j}[log^{p(\theta,X)}]}}{Z_j},$$

where Z_j is a normalization constant. In most cases the form of equation above matches a familiar distribution (e.g. Normal, Gamma etc.) after plug-in and Z_j turns out to be unimportant.

4 Non-parametric Estimator

Firstly we need to introduce two non-linear estimators used in our simulation work, namely, P-splines and Deep Forward Neural Network (DFNN)[5] .

To start with, a brief description on splines will be demonstrated as the corner-stone to structured additive terms. Based on that, we forward to the demonstration on smoothing splines and B-splines. Lastly, we come to the description on P-splines which combines ideas of B-splines and smoothing splines.

The other estimator used in our model is the deep neural network, where multiple layers of nodes are connected and outputs of one layer are transmitted as inputs of next layer. Here we call it Deep feedforward neural networks (DFNN), which is the first and simplest design for ANN since the information moves one-way from input layer through hidden layers to output layer. Other types of network could be as well deployed in Deep Generalized Additive Model for Location, Scale and Shape. Numerical variants for DFNN have been designed for various kinds of tasks and achieved remarkable performance in contests. For example, Convolutional Neural Network achieved excellent results in image classification up to human benchmark[35].

4.1 Splines & Smoothing Splines

A k-th order spline in statistics is defined as a continuous piecewise polynomial function of degree k with at least k-1 orders continuous derivatives at its knot points. Formally, function $f : \mathcal{R} \Rightarrow \mathcal{R}$ is a kth-order spline with $t_1 < t_2 < ... < t_m$ knot points with following properties:

- f is a polynomial of degree k at each single interval separated by knots $\{t_i\}_{i=1}^m$;
- the j-th derivative of f is continuous at each knot.

A commonly used case is cubic splines – continuous piecewise cubic functions with continuous first, and second derivatives. The continues of derivatives results into the smoothness of the function that makes the knots undiscovered. To parametrize the set of splines with knots $\{t_i\}_{i=1}^m$, a natural way is called truncated power basis with $g_1, g_2, ..., g_{m+k+1}$,:

$$g_1(x) = 1, \quad g_2(x) = x, \quad g_3(x) = x^2, \quad ... \quad , g_{k+1}(x) = x^k, g_{k+1+j}(x) = (f_{k+1+j}(x))^k,$$
$$\text{where } f_{k+1+j}(x) = max\{0, x - t_j\}, \quad j = 1, ..., m$$

For estimation, regression on splines could be performed by minimizing squared errors:

$$\sum_{i=1}^n (y_i - \sum_{j=1}^m \beta_j g_j(x_i))^2$$

And then the regression splines is obtained through $\hat{r}(x) = \sum_{j=1}^{m+k+1} \hat{\beta}_j g_j(x)$

An alternative way to define splines is to add constrain on degree to the left of the leftmost knot, and to the right of the rightmost knots, which are intervals $(-\infty, t_1]$ and $[t_m, \infty)$. Therefore we have *natural spines* only defined by odd orders k, normally three, with additional properties:

$$f \text{ is a polynomial of degree } (k-1)/2 \text{ on intervals } (-\infty, t_1] \text{ and } [t_m, \infty).$$

To circumvent the issue of knot selection and overfitting as well, scientists invented the smoothing splines by adding regularization or shrinkage on coefficients to the natural splines. Instead, the criterion is to minimize the squared errors under second order difference penalty:

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{m}\beta_j g_j(x_i))^2 + \lambda \sum_{i=1}^{n}\sum_{j=1}^{m}\beta_i\beta_j \int g_i^{''}(t)g_j^{''}(t)\,dt$$

The exact form of the penalty depends on situations and could be any order derivatives. The λ is the hyper-parameter for trade-off between bias and variance.

4.2 B-splines & P-splines

B-splines are constructed from polynomial pieces and joined at the knots. A B-spine of degree q is constructed through q+1 q-order polynomial pieces joined at q inner knots. For example, one B-spline of degree 1 consists of two linear pieces joined at 1 inner knot. The B-spline is positive on a domain spanned by q + 2 knots and zero elsewhere and overlaps with 2q polynomial pieces of its neighbors except at the boundaries. Consequently, q+1 B-splines are nonzero at each given x. When we design $k-1$ intervals by points $\{t_i\}_{i=1}^{m}$ for B-splines of degree q, totally we have $k+2q$ knots for construction of B-splines and $k+q$ B-splines

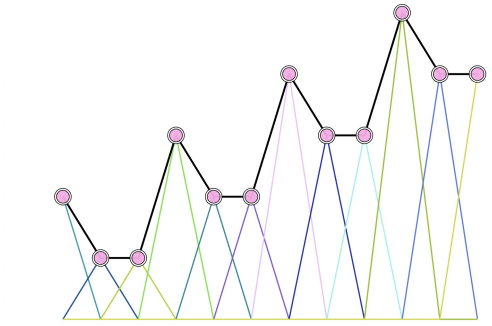


Figure 4.1: B-spline of degree one

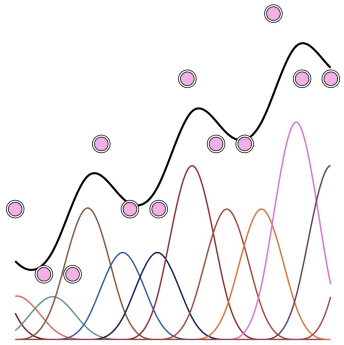


Figure 4.2: B-spline of degree three

A recursive definition of the B-splines is followed: firstly we have zero-degree B-splines defined on interval $[t_j, t_{j+1}], j = 1, ..., k-1$:

$$B_j^0(x) = \begin{cases} 1 & \text{x}\in [t_j, t_{j+1}], \\ 0 & otherwise. \end{cases}$$

For an increase in degree to $I+1$ with extra nodes t_{-I+m} and t_{m+I} :

$$B_j^{I+1}(x) = \frac{x-t_{j-I-1}}{x_j-t_{j-I-1}} B_{j-1}^I(x) + \frac{t_{j+1}-x}{t_{j+1}-t_{j-I}} B_{j-1}^I(x),\ j = 1, 2, ..., k+I-1$$

All B-splines of same degree are summed up to 1 at any value of x on its domain.

Then the idea of smoothing splines is introduced into B-splines and this resulted into penalized splines, P-splines[36]. O'Sullivan(1986[37], 1988[38]) formed an objective function by introducing a penalty on the second derivative of the fitted curve:

$$S = \sum_{i=1}^m \{y_i - \sum_{j=1}^n a_j B_j(x_j)\}^2 + \lambda \int_{x_{min}}^{x_{max}} \{\sum_{j=1}^n a_j B_j''(x_j)\}^2\, dx$$

With equidistant knots we have the following for the second derivative :

$$h^2 \sum_j a_j\, B_j''(x_j; q) = \sum_j \Delta^2 a_j\, B_j(x; q-2)$$

where $\Delta^2 a_j = a_j - 2a_{j-1} + a_{j-2}$ and h is the distance between two adjacent knots. Therefore, we can approximate the second derivative penalty by $\sum_{j=3}^d (\Delta^2 a_j)^2$, which is called second order difference penalty. Theoretically there is nothing special about the second derivative. Lower or higher orders might be used as well and the form could be changed accordingly.

4.3 Deep Neural Network

4.3.1 Deep Forward Neural Network

A deep forward neural network is a collection of multiple layers constructed by numerous neurons. The typical architecture of DFNN consists an input layer, several hidden layers and an output layer, as shown in figure 4.3 by Tran, M-N et al[5]. In each neuron, outputs from last layer (if exist) are received as input signals and aggregated. After activation layer in form of a linear transformation (such as logistic or sigmoid) or nonlinear transformation (such as Relu), processed signals are sent out as the input of next layer or the final output. According to universal approximation theorem, a DFNN can learn highly nonlinear relationships in data and is able to approximate any continuous function to arbitrary degrees of accuracy. If a DFNN is used for transforming the covariates, then the final output has the following form:

$$f_L(W_L, f_{L-1}(W_{L-1}, ..., f_1(W_1, X)...)),$$

where f_i is the aggregation and activation function of layer i, and W_i is the weight matrix.

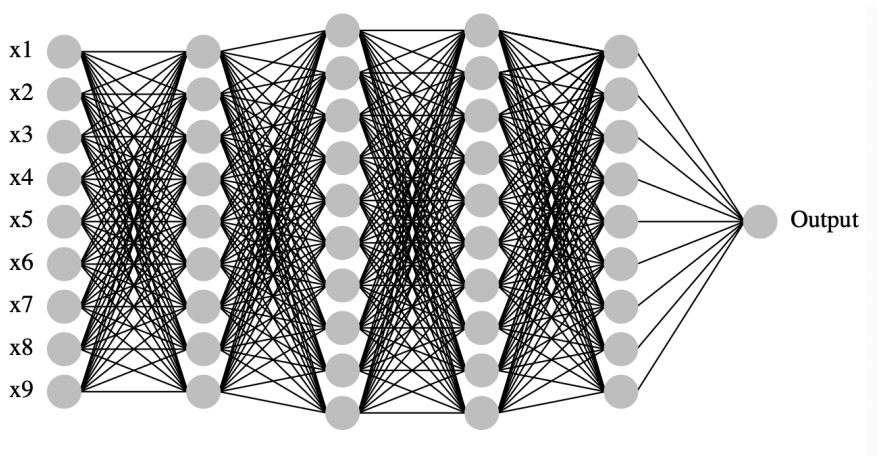


Figure 4.3: An example for multiple-layer Deep Forward Neural Network with four hidden layer[5]

A popular training strategy used in the neural network is call back-propagation[39][40]. Firstly, connection weights are randomly set and the error of the network using the current weights are measured by a specific loss function. Then the model passes this error back through the network and iteratively updates each weight by their contribution to the network error via the chain rule in calculus. One key to successfully train the model is a proper learning rate as the training meta-parameter. The figure 4.5 quoted from the paper by Yann LeCun et al[7] gives a vivid description on the back-propagation algorithm used in the neural network.

Various tools to improve performance and accelerate training speed have been designed. Among them, drop-out[41] and "Relu" activation function[6] shown in figure 4.4 are common choices for training large-scale network.

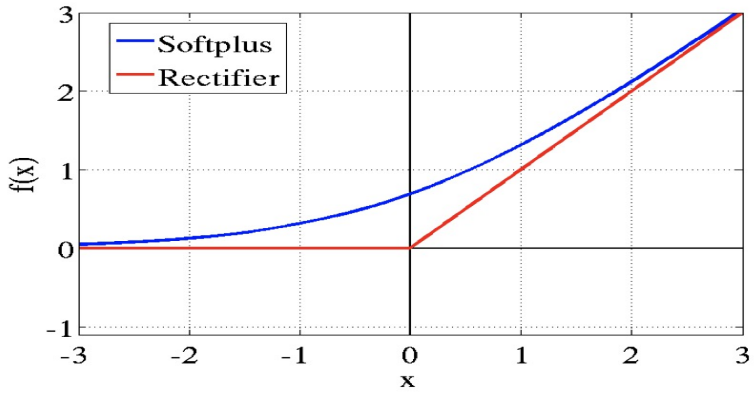


Figure 4.4: Rectifier (Relu) in comparison with Softplus activation function<$f(x) = ln(1 + e^x)$ >:[6]

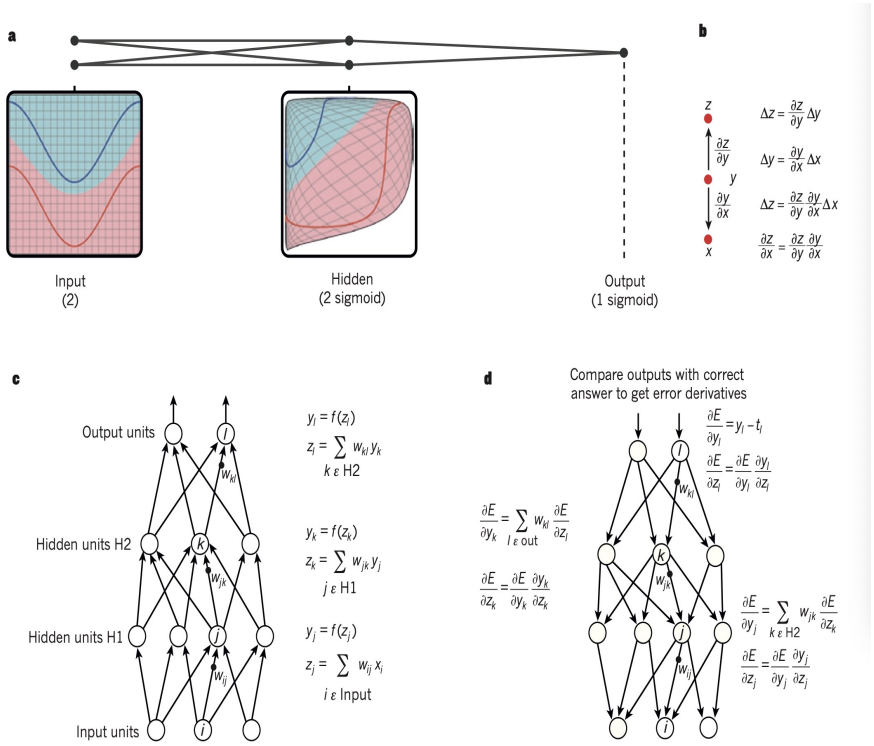


Figure 4.5: Multi-layer neural networks and back-propagation[7]: Part a shows how a three-layer neural network with two neurons in both input and hidden layer simulate non-linear boundary for two-fold classification issue; Part b is the chain rule in calculus used of composite of two functions; Part c depicts the information pipeline within a neural network; Part d gives details in back-propagation algorithm based on chain rule

4.3.2 Other Types of ANN in Practice

Other types of Artificial Neural Network could be similarly integrated into the Deep Generalized Additive Model for Location, Scale and Shape as the unstructured predictor. Here we mainly introduce three influential variants of neural network specified for uncontroversial inputs that could be potentially adopted in DGAMLSS, namely Convolutional Neural Network (CNN), Recurrent Neural networks (RNN), and Graph Convolutional Network (GCN).

Convolutional Neural Network (CNN)

Convolutional neural network is specially designed for 2-dimensional input such as images. In the last two decades, CNN has achieved impressive success in the field of image processing including detection, segmentation and recognition. Local connections, shared weights, pooling and multiple layers are four key ideas behind. The basic element to construct a multilayer convolutional neural network is con-

sisted of convolutional layer and pooling layer. Input of neurons in one layer is summed up through the filter bank, which is a set of weights or a discrete convolution for local patch normally in shape of 5×5 or 3×3. Pooling or subsampling layers transfer a number of local patch inputs into one single output value. Since a neighbor of input values is virtually highly correlated, images are compressed through such architecture to reduce computation and to gradually build up further spatial and configure invariance with minimum loss of information.

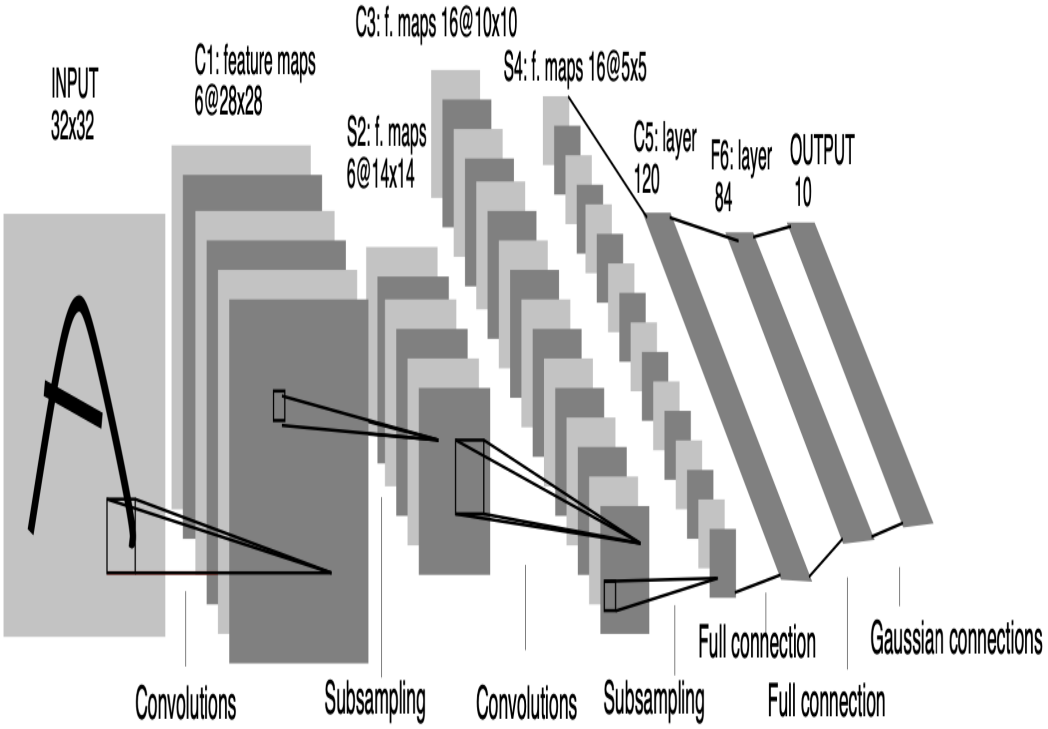


Figure 4.6: An Example on CNN: LeNet-5[8] (from left to right: the input image, first convolutional layer, first pooling layer, second convolutional layer, second pooling layer, two full-connection layers and output layer)

Recurrent Neural networks (RNN)

For tasks with sequential inputs, such as speech and text, Recurrent Neural Network (RNN) is one of ideal options. RNN is a powerful tool to predict the next word in a sentence or the next character for a word. It processes one element of the input at a time, providing not only element of sequential outputs but status vector containing information on all the past elements of the sequence as the accessorial inputs to the next iteration.

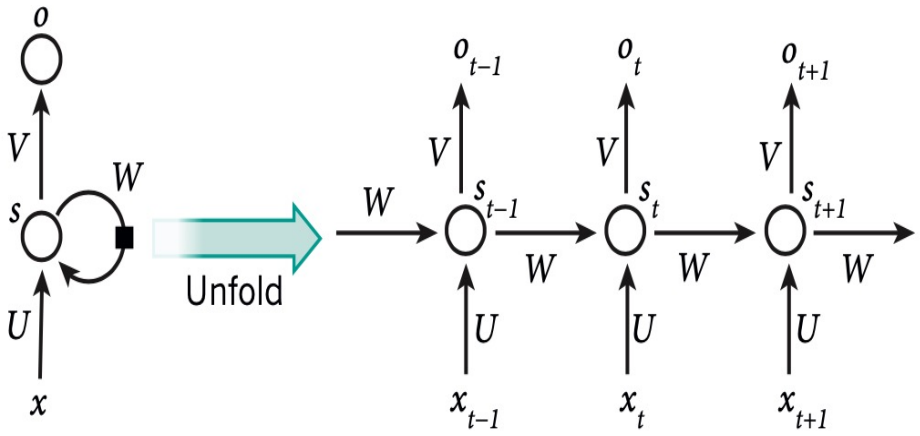


Figure 4.7: The Stretch of A Recurrent Neural Network[7](left: elements of inputs x are sequentially processed by neuron s with status vector W from last run; right: such procedure could be viewed as a sequence of neurons lined up, and each neuron x_t process both input x_t and status vector from the previous neuron

Graph Convolutional Network (GCN)

The last model introduced in this Chapter is called Graph Convolutional Network[42], which is a variant based on Graph Neural Network extending previous neural network methods to process datasets in graph domains. A graph $\mathbb{G}$ is a pair $(\mathbb{N}, \mathbb{E})$ representing the relationships and connections among nodes $\mathbb{E}$ in data and therefore is an essential factor to analyze. Graph Convolutional Network is the model using spectral convolutional method on undirected graph to tackle classification issue. A multi-layer GCN employ the following layer-wise propagation rule:

$$H^{(l+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}),$$

where $\widetilde{A} = A + I_N$ is the adjacency matrix with added self-connections by I_N, $\widetilde{D} = \Sigma_j \widetilde{A}_{ij}$, $W^{(l)}$ is the trainable weight matrix specified for layer l, and $\sigma(.)$ is the activation function. Technically, this propagation rule is motivated via a first-order approximation of localized spectral filters on graphs[43][44].

For example, a two-layer GCN design demonstrated in the their paper for semi-supervised node classification has following structure:

$$Z = f(X, A) = softmax(\widetilde{A}\, ReLU(\widetilde{A}XW^{(0)})\, W^{(1)})$$

with cross entropy error $\mathscr{L} = -\sum_{l \in_L} \sum_{f=1}^{F} Y_{lf}\, ln(\, Z_{lf})$.

5 Experimental Result

5.1 Classical Distributions

A simulation work with several independent experiments has be completed for comparison of estimation by all four models mentioned above, namely GAMLSS, BAMLSS, gamboostLSS and DGAMLSS. All following issues are analyzed based on serial measurements:

- separability of structured and deep predictor;
- size and coverage of confidence intervals using variational Bayes;
- empirical uncertainty in comparison to state-of-the-art GAM(LSS) estimation;
- influence of different distribution assumptions.

We mainly focuse on two scenarios:

- one is for classical distribution functions including Poisson distribution, Gamma distribution, Negative Binomial distribution, Normal distribution, Lognormal distribution and Logistic distribution;
- the other aims on more complicated distributional function – Zero-inflated Poisson Distribution.

All models are estimated based on same dataset expect for Gamma and Negative Binomial distribution.

We start with Poisson distribution as discrete case with only one parameter to be estimated. Afterwards we move onto more challenging (discrete) distributions with two parameters, namely Gamma and Negative Binomial distribution. In both cases, datasets are separately generated for each model since different parameters are targeted. A brief mathematical demonstration will be presented in place. Then experiments on two-parametric distributions in continues cases are studied: namely normal distribution, lognormal distribution and logistic distribution. We carry out experiments on these distributions as they are most representative and widely-used in statistical analysis. Lastly, a two-parametric distribution – Zero-inflated Poisson distribution – is chosen as the case for complex mixture distribution assumption. In practice, the zero-inflated Poisson distribution is commonly used in various situations including applications of patent citation and claim of insurance. The number of zeros tremendously exceeds the expectation by Poisson distribution, which means a large amount of publications and patents is never cited, or insurances are not claimed during the policy period[17].

All calculations and simulations are conducted on the R software for statistical computing, and model implementation are proceed by add-on packages "bamlss", "gamlss", "gamboostLSS" and "DGAMLSS".

5.1.1 Data Generating

Considering of the wide range of distribution types we model in this paper, a brief description of data generation and model construction is necessary.

There are totally 3 explanatory variables in our simulation work. Among these three variables, x is assumed to follow a standard normal distribution, z is under uniform distribution between -3 and 3, and v is the uniform distributional variable on the interval $[1, 6]$. For parameter ν in distribution function $f(y|\nu)$, we assumed a monotonic link function $g(.)$ relating all explanatory variables to the parameter through structured additive regression:

$$g(\nu) = \eta^{\nu} = \beta_0 + f_1^{\nu}(.) + f_2^{\nu}(.) + ... + f_k^{\nu}(.)$$

In addition to two-order linear regression, various non-linear functions are applied onto explanatory variable as shown in the following list. All non-linear function are reconstructed through orthogonalization as sum-to-zero functions, shown in figure 5.1 to 5.6.

$f_{1,x}(x) = sin(3x)$	$f_{1,z}(z) = 0.1 \cdot z^3$	$f_{1,v}(v) = log(v)$
$f_{2,x}(x) = cos(0.5x)$	$f_{2,z}(z) = 2^{z-1}$	$f_{2,v}(v) = exp(0.5 \cdot v)$

Table 5.1: List of non-linear functions in data-generating process

Take Poisson distribution as an example, only one parameter λ would be estimated with two explanatory variables x & z under log link function:

$$log(\lambda) = 2 - x + f_{1,x}(x) + f_{1,z}(z)$$

And details of data generating process for two-dimensional normal, lognormal and logistic distribution are demonstrated by following list:

Name	Distributional Function	Link 1	Link 2
Normal	$f(y\|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(y-\mu)^2}{2\sigma^2}}$	$\eta_1 = \mu$	$\eta_2 = log(\sigma)$
Lognormal	$f(y\|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(log(y)-\mu)^2}{2\sigma^2}}$	$\eta_1 = log(\mu)$	$\eta_2 = log(\sigma)$
Logistic	$f(y\|\mu,\sigma) = \frac{exp(-\frac{y-u}{\sigma})}{(1+exp(-\frac{y-u}{\sigma}))^2}$	$\eta_1 = \mu$	$\eta_2 = log(\sigma)$

Table 5.2: Distribution functions and links for normal, lognormal, and logistic distributions

Both linear predictors η_1 and η_2 are explained by the additive regression functions of x and z:

$$\eta_1 = 2 - x + f_{1,x}(x) + f_{1,z}(z), \eta_2 = 1 + 0.5 \cdot z + f_{2,x}(x) + f_{2,z}(z)$$

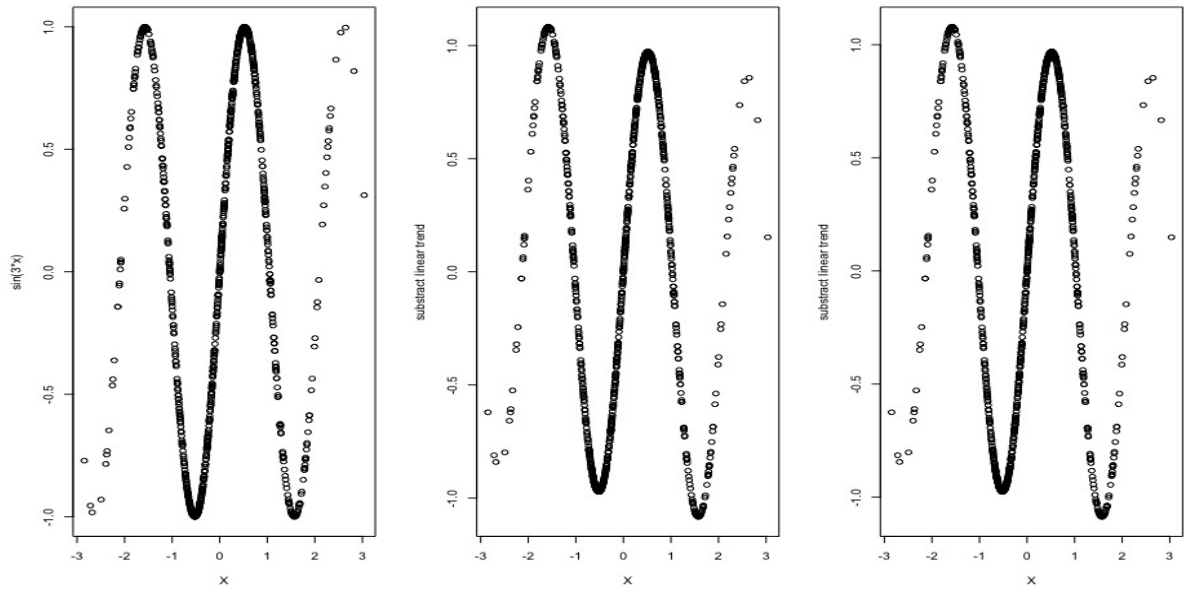


Figure 5.1: Orthogonalization on $f_{1,x}(x)$ (from left to right: the original function, subtraction of linear trend, subtraction of linear trend of the intercept)

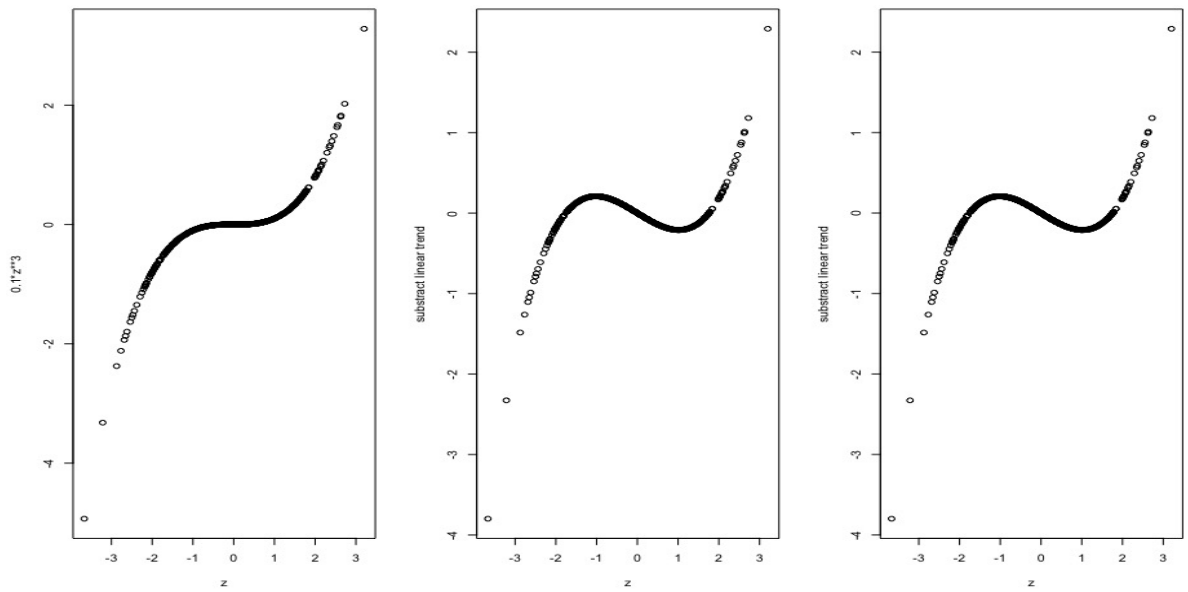


Figure 5.2: Orthogonalization on $f_{1,z}(z)$(from left to right: the original function, subtraction of linear trend, subtraction of linear trend of the intercept)

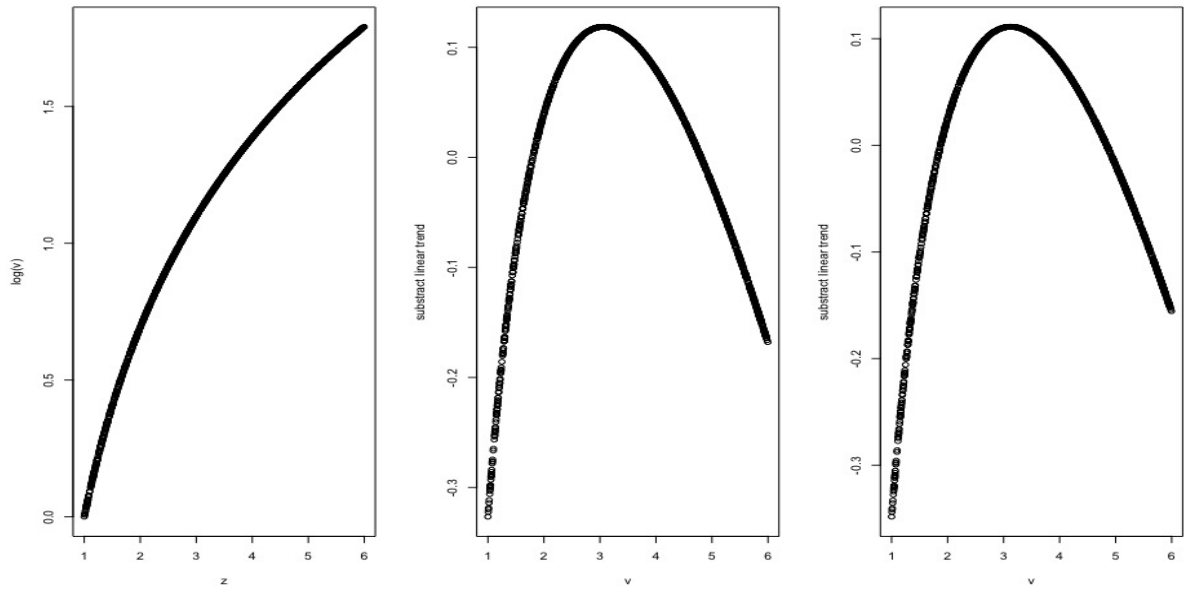


Figure 5.3: Orthogonalization on $f_{1,v}(v)$(from left to right: the original function, subtraction of linear trend, subtraction of linear trend of the intercept)

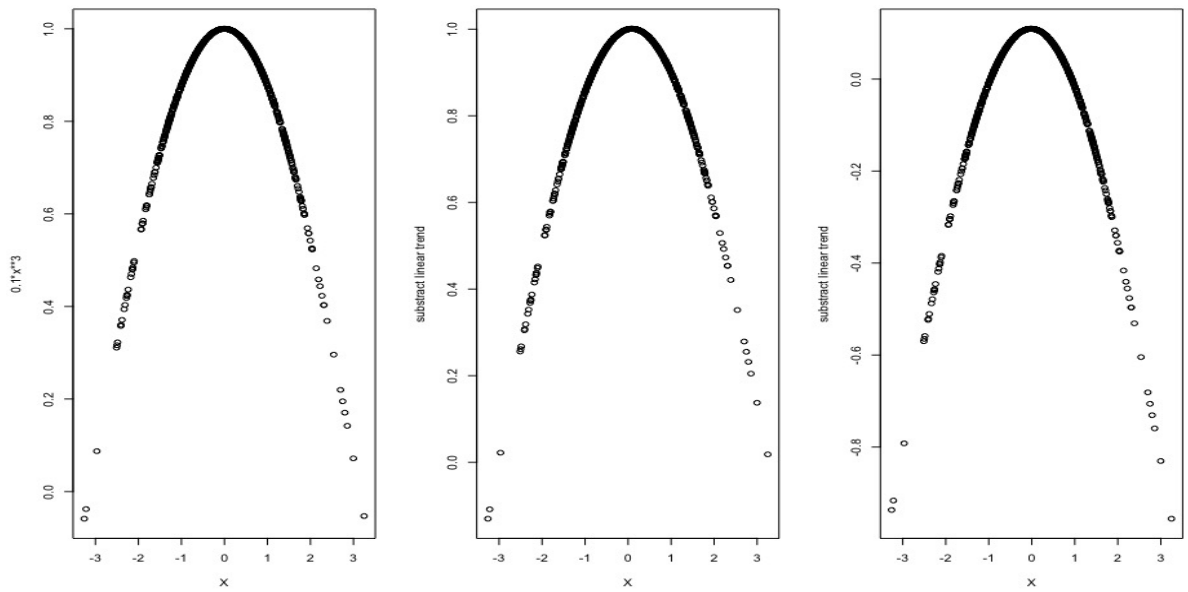


Figure 5.4: Orthogonalization on $f_{2,x}(x)$(from left to right: the original function, subtraction of linear trend, subtraction of linear trend of the intercept)

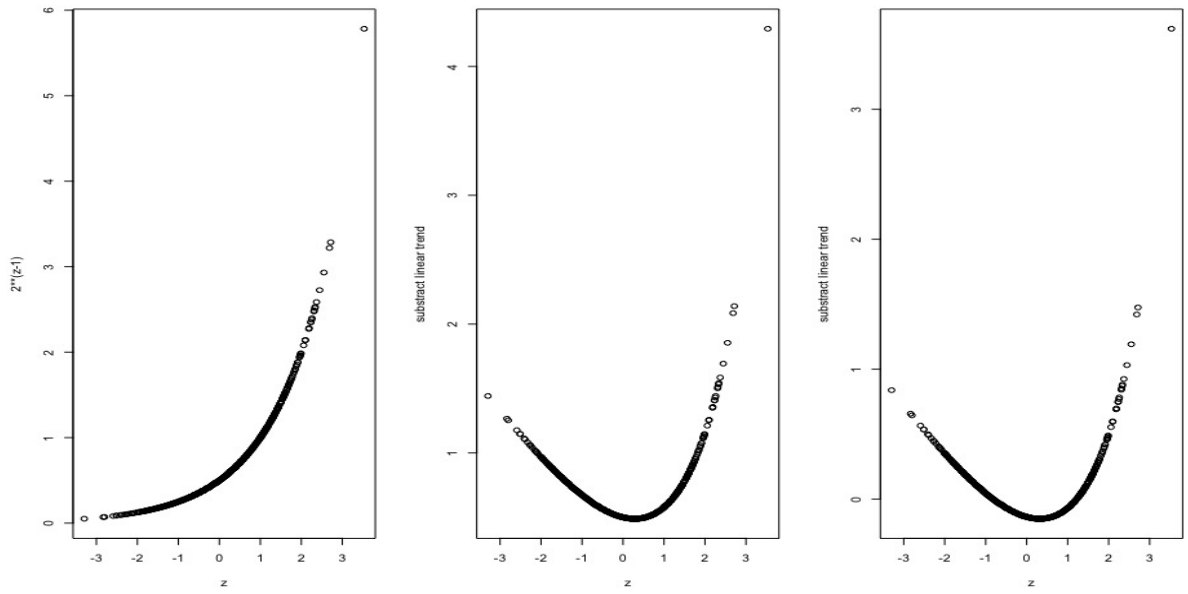


Figure 5.5: Orthogonalization on $f_{2,z}(z)$(from left to right: the original function, subtraction of linear trend, subtraction of linear trend of the intercept)

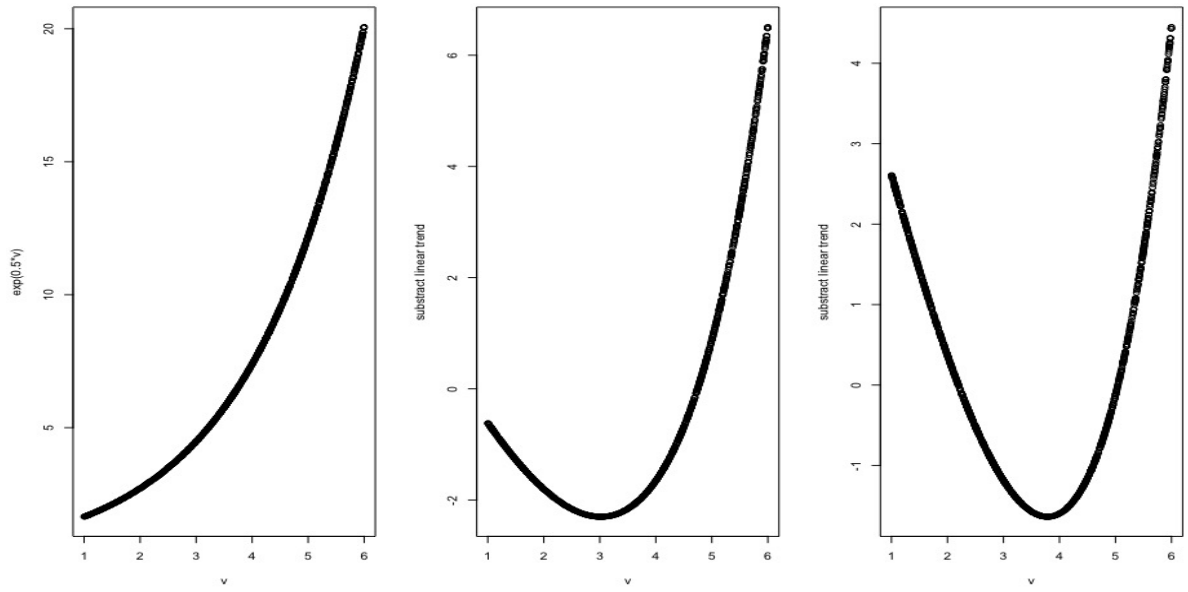


Figure 5.6: Orthogonalization on $f_{2,v}(v)$(from left to right: the original function, subtraction of linear trend, subtraction of linear trend of the intercept)

In contrast, the parameterization process for both Gamma and Negative Normal distributions is mildly more complicated since different parameters are estimated through each model. For the Gamma distribution, we imply two individual data generating processes, the first is implemented for the BAMLSS model with two parameters, μ_1 as the mean and σ_1 as the shape parameter. Both parameters are assigned with log link functions:

$$\eta_1 = log(\mu_1) = 2 - x + f_{1,x}(x) + f_{1,z}(z)$$
$$\eta_2 = log(\sigma_1) = 1 + 0.5 \cdot z + f_{2,x}(x) + f_{2,z}(z)$$

And the density distribution function conditioned on μ_1 and σ_1 is:

$$f(y|\mu_1, \sigma_1) = \frac{y^{\sigma_1 - 1} * e^{(-\frac{\sigma_1 * y}{\mu_1})}}{(\frac{\mu_1}{\sigma_1})^{\sigma_1} * \Gamma(\sigma_1)}$$

The other is for GAMLSS, gamboostLSS and DGAMLSS (with family="gammar"), where μ_2 stands for the mean but the σ_2 is instead the square root of the usual dispersion parameter for a GLM gamma[2]. This resulted to a new density distribution function:

$$f(y|\mu_2, \sigma_2) = \frac{y^{(\frac{1}{\sigma_2^2} - 1)} * e^{(-\frac{y}{\mu_2 \sigma_2^2})}}{(\mu_2 \sigma_2^2)^{\frac{1}{\sigma_2^2}} * \Gamma(\frac{1}{\sigma_2^2})}$$

And the relation between these two sets of parameters is: $variance = (\mu_1 * \sigma_1)^2 = \frac{(\mu_2)^2}{\sigma_2}$

Moreover, three datasets are generated by each iteration regarding to Negative Binomial distribution. The first two datasets are produced under type II negative binomial distribution for the BAMLSS and GAMLSS/ gamboostLSS respectively with the probability to a given number of failures until r-th successes with p as probability of each success. The BAMLSS targets on two parameters: the mean and the r as μ_1 and θ. The density function is

$$f(y|\mu_1, \sigma) = \frac{\Gamma(y + \frac{\mu_1}{\sigma})}{\Gamma(\frac{\mu_1}{\sigma}) * \Gamma(y+1)} * \frac{\sigma^y}{(1+\sigma)^{(y + \frac{\mu_1}{\sigma})}},$$

and GAMLSS/ gamboostLSS model parameters μ_2 and σ under probability distribution function:

$$f(y|\mu_2, \sigma) = \frac{\Gamma(y + \frac{\mu_2}{\sigma})}{\Gamma(\frac{\mu_2}{\sigma}) * \Gamma(y+1)} * \frac{\sigma^y}{(1+\sigma)^{(y + \frac{\mu_2}{\sigma})}}$$

Howbeit, the DGAMLSS follows the distributional function "tfd_negative_binomial" in R-package tensorflow-probability with two parameters total_count k and probability p.

$$\eta_1 = log(k) = 1 - 0.5 \cdot x + f_{1,x}(x) + f_{1,z}(z)$$
$$\eta_2 = logit(p) = 1 + 0.5 \cdot z + f_{2,x}(x) + f_{2,z}(z)$$

This function models the probability to a specific number of successes until kth failure with p as probability of each success with conditional probability density function:

$$f(y|k, p) = \frac{\Gamma(y+k)}{\Gamma(k) * \Gamma(y+1)} * p^y * (1-p)^k$$

All datasets are generated with 1500 samples by each iteration and totally 100 replications. For Poisson and Normal distributions, analysis on additional datasets with noises is implemented in two-dimensional cases in order to verify robustness of models.

```
link1=function(b0_mu,bx_mu,x,z)
{ return(b0_mu + bx_mu*x + f_x_1(x) + f_z_1(z))}

mu_gs <- link1(b0_mu,bx_mu,x,z)

link2=function(b0_sigma,bz_sigma,x,z)
{ return(exp(b0_sigma + bz_sigma*z + f_x_2(x)+f_z_2(z)))}

sigma_gs <- link2(b0_sigma,bz_sigma,x,z)

y_gs<-rnorm(n,mean = mu_gs,sd = sigma_gs)
data_gs <- data.frame(y = y_gs,x = x, z = z)
```

5.1.2 Model Specification

BAMLSS

We implement the BAMLSS model containing linear regressors and P-splines with 20 equidistant knots by each variables under second order difference penalty. The default backfitting algorithm is used in optimization phrase, and for the sampling phrase we use the default proposal function based on iteratively weighted least squares with the default proposal function "GMCMC_iwlsC_gp" assuming an inverse gamma prior. In addition, we use a boosting algorithm optimizer for logistic distribution by selecting the model term with the largest contribution to the log-likelihood, since result by back-fitting algorithm is slightly unstable. 12000 iterative samples are generated through MCMC by every iteration and the first 2000 are set as the burn-in phrase. Every tenth sample is stored for estimation.

In addition, we add the constrain on main effect to p-splines terms, since estimation of linear coefficients could be unstable when combined with non-linear effects. Theoretically, the linear effect and non-linear effect based on the same variable could be misidentified or even integrated by estimation if there is no constrain on the non-linear estimator. By introducing orthogonal reconstruction on true non-linear functions and constrain on non-linear estimation, the structured linear and structure non-linear are technically distinguishable.

```
bf_f<-list(y ~ x + tx(x,bs="ps",m=c(2),ctr="main")
           + tx(z,bs="ps",m=c(2),ctr="main"),
           sigma ~  z + tx(x,bs="ps",m=c(2),ctr="main")
           + tx(z,bs="ps",m=c(2),ctr="main"))
bf_gs <- bamlss(bf_f,data=data_gs,knots = list(x_knots,z_knots),
                family = "gaussian_bamlss",n.iter = 12000, burnin = 2000, thin = 10)
```

GAMLSS

We used the GAMLSS with both linear and structured nonlinear regressors in form of cubic P-splines with second order difference penalization. The degree of freedom of P-spline term is set to 1, which is the trace of the smoother matrix minus two (for constant and linear fit), so that the linear regressor wouldn't be automatically canceled out during estimation. Instead of using penalized likelihood function, a minimum of decrease of global dividend of 0.001 for each iteration and a maximum step of 20000 are used to avoid overfitting.

```
gf_logis=gamlss(formula = y~x
                + pb(x,nknots=20,control = pb.control( df=1,degree = 3, order = 2))
                + pb(z,nknots=20,control = pb.control( df=1, degree = 3, order = 2)),
                sigma.formula = ~ z
                + pb(x,nknots=20,control = pb.control(  df=1,degree = 3, order = 2))
                + pb(z,nknots=20,control = pb.control( df=1, degree = 3, order = 2)),
                 data=data_logis,  family = LO(),
                control = gamlss.control(c.crit = 0.001,n.cyc = 1000))
```

gamboostLSS

The setting up for gamboostLSS is mostly identical to GAMLSS but additionally with the constrain on non-linear functions to be centered. Such constrain worked analogously to main effect constrain on BAMLSS model to remove linear trend, and to assure a zero-integration structured non-linear estimator. Such constrain could theoretically improve estimation accuracy on linear estimator as well. The cyclic fitting method is adopted and we used 5-fold cross-validation to choose an early stop for estimation.

```
gm_logis=gamboostLSS(formula = list(mu = y ~ bols(x, df = 1)
                                    + bbs(x, knots = 20, df = 1, center = TRUE)
                                    + bbs(z, knots = 20, df = 1, center = TRUE),
                                    sigma = y ~ bols(z, df = 1)
                                    + bbs(x, knots = 20, df = 1, center = TRUE)
                                    + bbs(z, knots = 20, df = 1, center = TRUE)),
                     data = data_logis,families = as.families("LO"),method = "cyclic",
                     control = boost_control(mstop = 20000))
cvr <- cvrisk.mboostLSS(gm_logis, folds = mboost::cv(model.weights(gm_logis), B = 5, type = "
  kfold"))
gm_logis=gamboostLSS(formula = list(mu = y ~ bols(x, df = 1)
                                    + bbs(x, knots = 20, df = 1, center = TRUE)
                                    + bbs(z, knots = 20, df = 1, center = TRUE),
                                    sigma = y ~ bols(z, df = 1)
                                    + bbs(x, knots = 20, df = 1, center = TRUE)
                                    + bbs(z, knots = 20, df = 1, center = TRUE)),
                     data = data_logis,families = as.families("LO"),method = "cyclic",
                     control = boost_control(mstop =mstop(cvr)))
```

DGAMLSS

Firstly, two 5-layer deep neural networks with activation "Relu" and drop-out rate at 0.2 is defined as the unstructured predictor, including a linear activation for the final layer. Ideally, each network is specified for one parameter of the distributional function. For additive estimators of parameters, all of linear regressor, P-splines under second order difference penalty and network outputs are combined to access to a more precise estimation.

```
gr_gs <- deepregression(
  y =  as.numeric(y_gs),
  data = data,
  list_of_formulae = list(loc = ~ 1 + x + s(x, bs="ps",m=c(2))
                          + s(z, bs="ps",m=c(2))
                          + networka(x,z),
                          scale=~ 1 + z + s(x, bs="ps",m=c(2))
                          + s(z, bs="ps",m=c(2))
                          + networkb(x,z)),
  list_of_deep_models = list(networka = deep_model, networkb =deep_model),
  family ="normal",
  variational = TRUE,
  #validation_split = NULL,
  df=10)
cvres <- gr_gs %>% cv(cv_folds = 5, epochs=300)
bestiter <- stop_iter_cv_result(cvres)
gr_gs <- deepregression(
  y =  as.numeric(y_gs),
  data = data,
  list_of_formulae = list(loc = ~ 1 + x + s(x, bs="ps",m=c(2))  + s(z, bs="ps",m=c(2))  +
    networka(x,z),
                          scale= ~ 1 + z + s(x, bs="ps",m=c(2))  + s(z, bs="ps",m=c(2))  +
                            networkb(x,z)),
  list_of_deep_models = list(networka = deep_model, networkb =deep_model),
  family ="normal",
  validation_split = NULL,
  variational = TRUE,
  df=10)
gr_gs %>% fit(epochs=bestiter, verbose = TRUE, view_metrics = TRUE,
                  save_weights = TRUE)
```

Instead of splitting all samples into training and testing datasets, the 5-fold cross-validation method with 300 epochs in each phrase determines the number of epoch in training phrase, in which case the global minimum total loss of estimations is insured as well.

In addition, the estimated function with default number of degree of freedom appeared to be seriously fluctuated and therefore overfitted. We manually find a proper value for degree of freedom of the structured non-linear estimation for trade-off between bias and variance.

5.1.3 Estimation Results

In this simulation work, we mainly focus on the issue of variance decomposition, in which case some covariates are shared in linear, non-linear and unstructured functions, and we manage to provide precise estimation on each part. Therefore, the true non-linear functions are assumed to be sum-to-zero without visual linear trend. Application for such assumption is common in time series analysis or sequential data, where long-term trend and short-term fluctuation are combined in analysis. As stated previously, we add constrains onto models in place to achieve such decomposition and use the neural network part as an accessorial estimator accounting for noises or unexpected effects. Therefore the neural network included in our work is relatively plain while complicated neural networks are computation-consuming along with problem in overfitting.

One-parametric distribution: Poisson Distribution

The estimation result of Poisson distributional dataset provides a direct insight into our simulation work. In two-dimensional case with explanatory variables x and z, we firstly simulate the true value of parameter λ at each pair of parameter values, in comparison to estimated values by all of four models. Figure 5.7 shows the true λ in three-dimensional shape. Variable x dominates the scale of true values since both linear and non-linear parts are dependent on x.

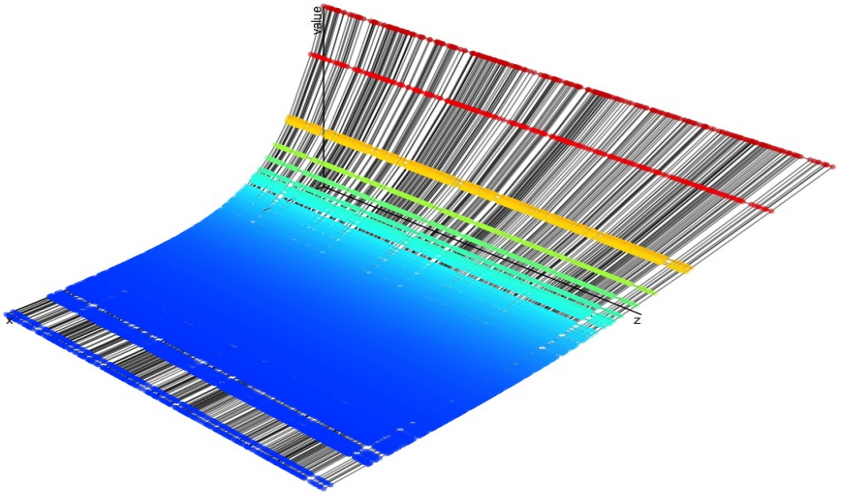
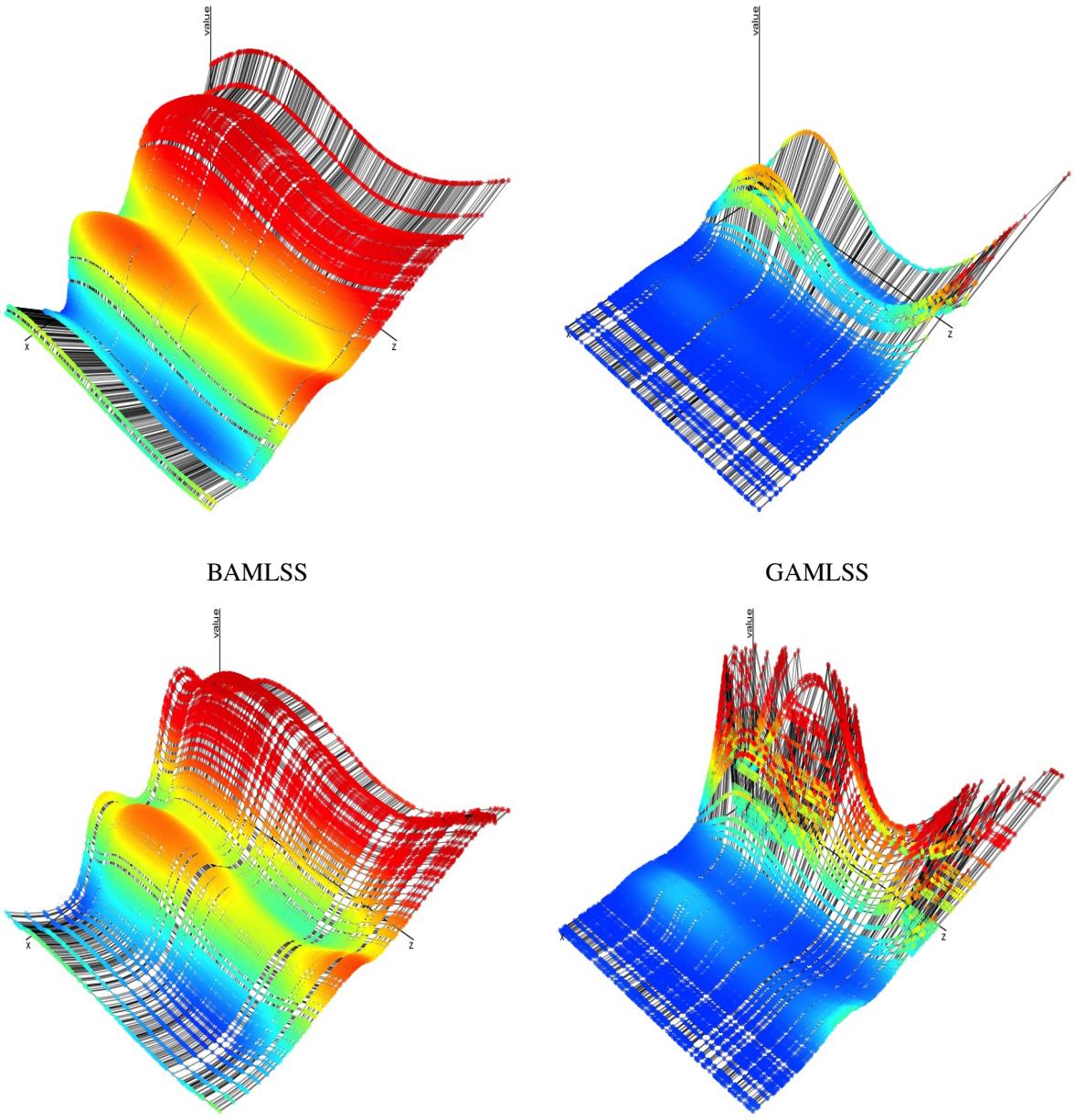


Figure 5.7: True values of parameters depending on x and z

The following figure presents estimated value by four models for a first-impression. Estimation of DGAMLSS is fluctuating in the red part since extreme values are removed in the plotting. At low level of true values, both GAMLSS and DGAMLSS provide consistent estimations.



BAMLSS GAMLSS

gamboostLSS Deep Distributional Learning model

Figure 5.8: Estimation of parameters based on x an z by different models

From each model, we extract the estimation of linear and non-linear functions, and analyze the result from different respects. For the linear function we draw box-plots both for estimated constant and linear coefficient by 100 iterations[5.9].

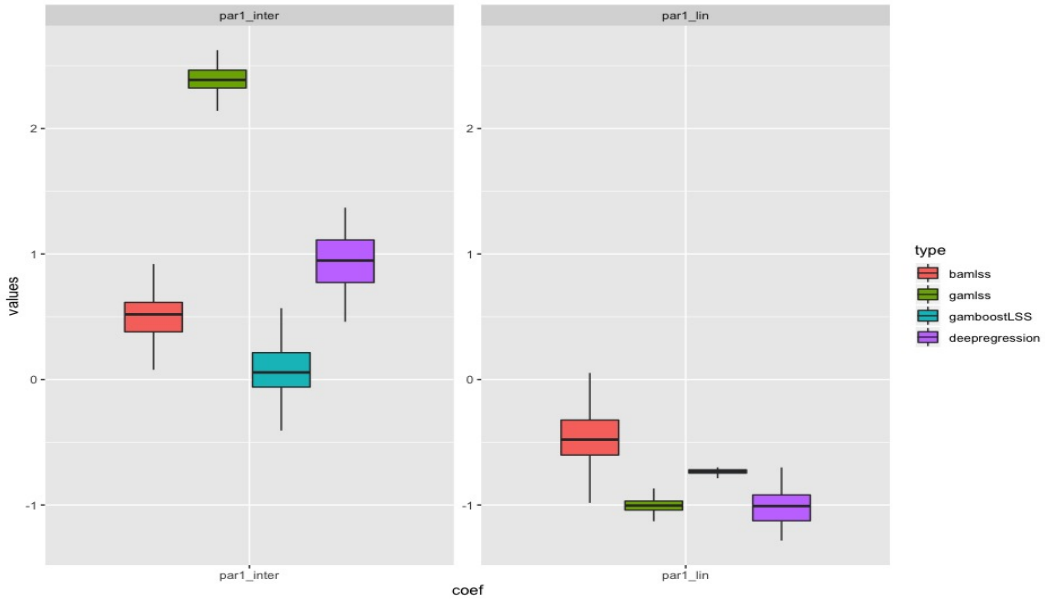


Figure 5.9: Linear Estimation (Poisson distribution, two-dimensional case) by each model in 100 iteration (figure on the left: intercept in parameter 1; figure on the right: linear coefficient in parameter 1; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression)

Recall that true values are respectively two and minus one. Both GAMLSS and Deep Distributional Learning model provide acceptable results in linear estimation. Howbeit, the estimation by Deep Distributional Learning model is relatively unstable as introducing the deep neural network into the model.

Then we calculate the mean squared error (MSE) of all estimations by following form:

$$MSE(\lambda) = E_\lambda[(\hat{\lambda} - \lambda)^2] = \frac{1}{n}\sum_{i=1}^{n}(\lambda_i - \hat{\lambda}_i)^2$$

Notice that the mean squared error could be interpreted as

$$MSE(\lambda) = Var_\lambda(\hat{\lambda}) + Bias_\lambda(\hat{\lambda}, \lambda)^2,$$

a subtle balance on the bias and variance is desired and the estimator is expected to be consistent (both accurate and stable).

coefficients	par1_inter	par1_lin
bamlss	-1.889149	-3.850300
gamlss	-4.523495	-8.535910
gamboostLSS	-1.392018	-5.340982
deep-regression	-2.534780	-6.901801

Table 5.3: MSE(log) of estimated linear coefficients by each model (Poisson distribution, two-dimensional case)

Generally, the GAMlSS provides most reliable estimation of linear model, and Deep Distributional Learning model works relatively better than other two models.

For the structured non-linear function we employ two criterion. Firstly, we plot the averaged non-linear estimator in accompany with the true function as shown in figure 5.10. No linear shift appears in estimation by all models. Estimated non-linear function by gamboostLSS on the left fails on the boundary, since the model uses linear extension for variables out of boundary.

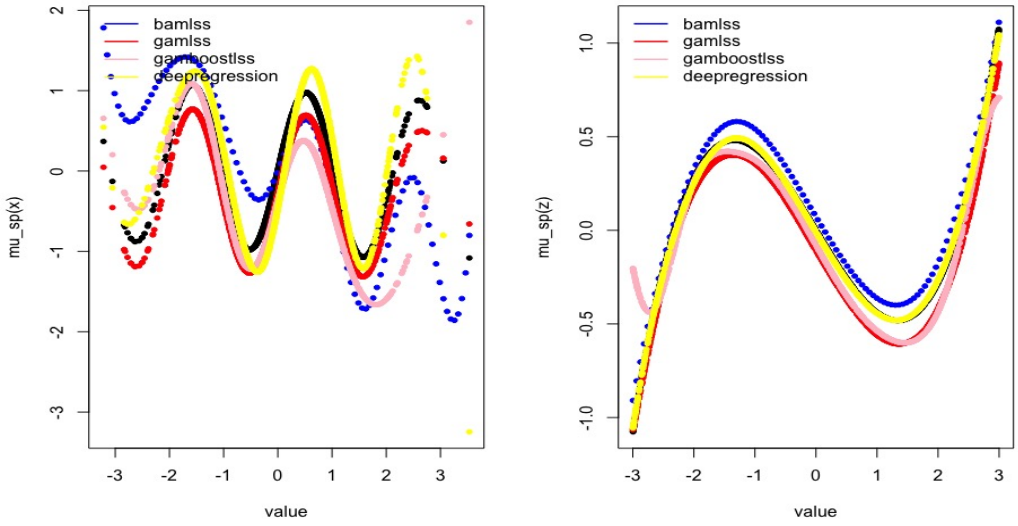


Figure 5.10: Averaged non-linear estimation by each model (Poisson distribution, two-dimensional case)

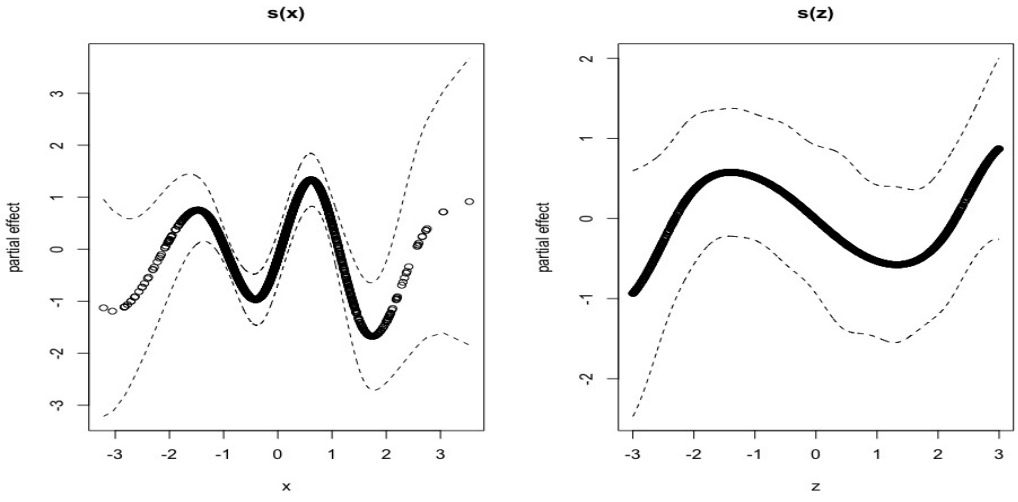


Figure 5.11: Variational inference on estimation of non-linear function by Deep Distributional Learning model (Poisson distribution, two-dimensional case)

Figure 5.11 shows results on variational inference on non-linear estimation given by Deep Distributional Learning model and evidence to the convergence of DGAMLSS estimation.

The the Mean Integrated Squared Error (MISE) is calculated as:

$$E||f_n - f||_2^2 = E\int(f_n(x) - f(x))^2 dx$$

Here we instead use the L-2 norm distance of each single estimation to the true function

$$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i) - f(x_i)]^2)$$

as the adjusted Mean Squared Error. Finally we draw boxplots in figure 5.12 for the adjusted squared error for comparison among models. It's obvious that the Deep Distributional Learning model outperforms other models in respect to non-linear estimation based on both two covariates.

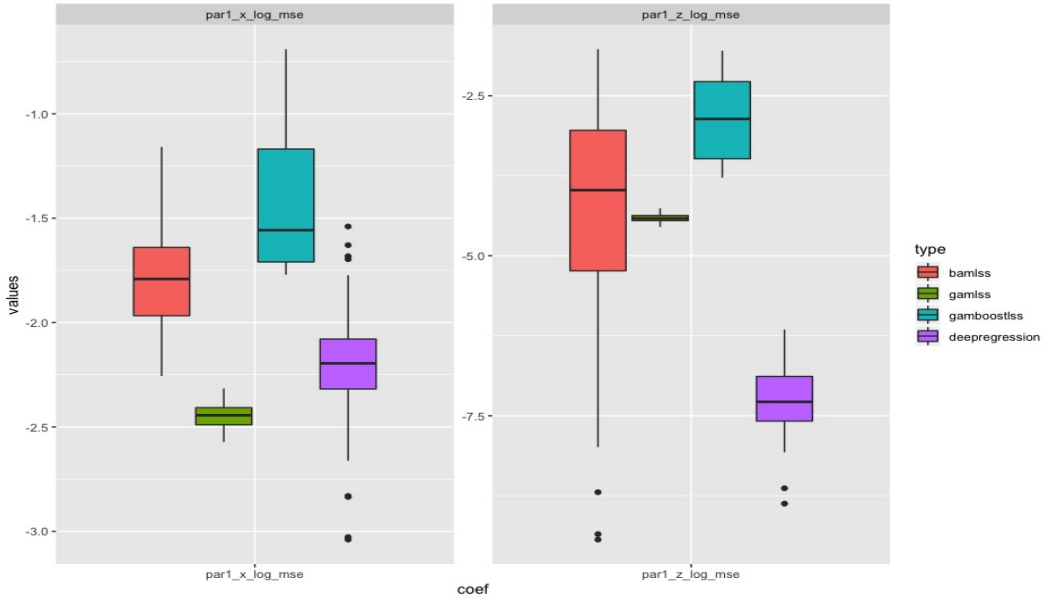


Figure 5.12: Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i) - f(x_i)]^2)$> of non-linear estimation by each model (Poisson distribution, two-dimensional case)

Nonetheless, the mean squared error of the estimated parameter is calculated based on a test dataset containing 200×200 pairs of new-generated x and z. The DGAMLSS model is proven to have the minimal value among all of four models, which means DGAMLSS achieves the best accuracy in estimating parameters but fails to distinguish the effect in respect to covariates and terms.

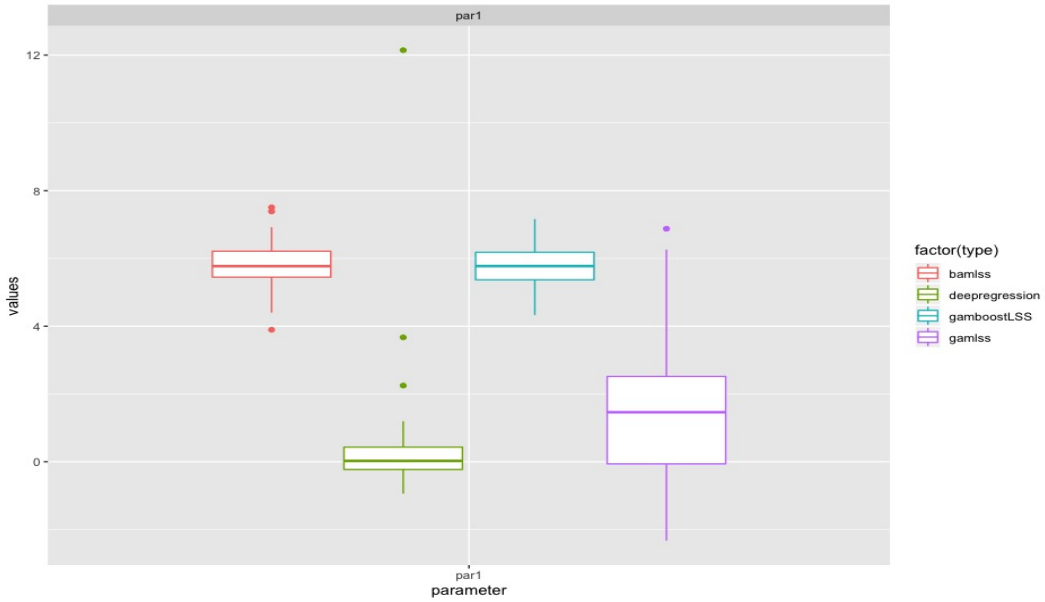


Figure 5.13: MSE of estimated parameters on test dataset (Poisson distribution, two-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS)

Furthermore, we look into the case where noises are included into the true data. In practice, the measured values could be slightly inaccurate due to both endogenic factors in data generating process or exogenous factors in data collecting process. Thereafter, statistical models are supposed to be robust against those noises. For Poisson distributional data, we add noise variables under $Binomial(2, 0.5)$ distribution into the generated samples y and work through the analysis iteratively on the new dataset.

coefficients	par1_inter	par1_lin
bamlss	-2.097615	-3.413322
gamlss	-5.046586	-7.000056
gamboostLSS	-2.161485	-4.599167
deepregression	-3.223227	-5.989382

Table 5.4: MSE(log) of estimated linear coefficients by each model (with noises, Poisson distribution, two-dimensional case)

For the linear estimation, results by the Deep Distributional Learning model is relatively worse that those by GAMLSS as being biased with large variance.

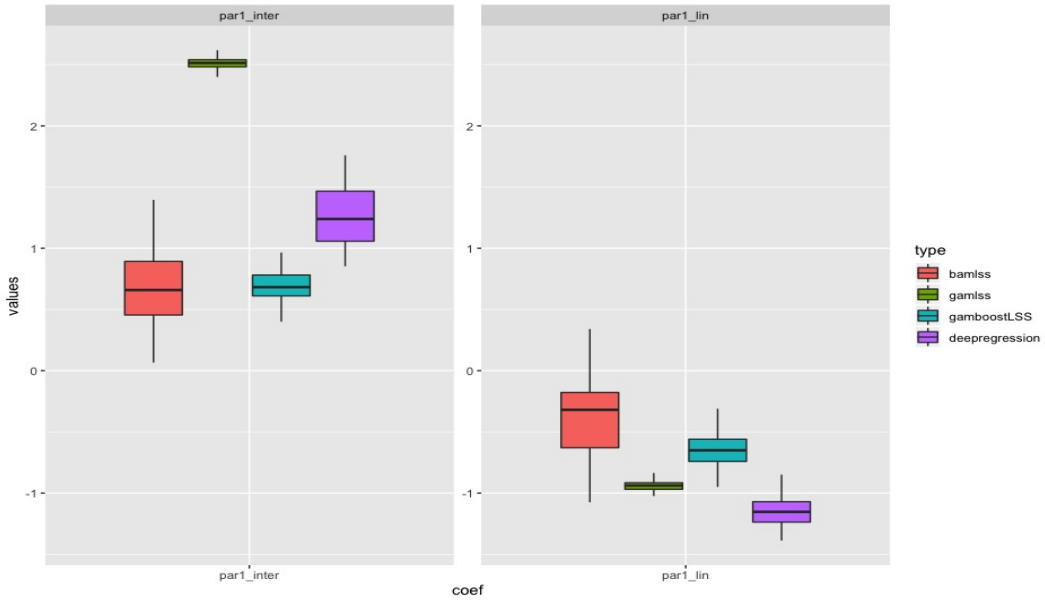


Figure 5.14: Linear Estimation (with noise, Poisson distribution, two-dimensional case) by each model in 100 iteration (figure on the left: intercept in parameter 1; figure on the right: linear coefficient in parameter 1; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression)

Then we present results in non-linear estimation. In figure 5.15, estimation by all models are inclined to be unstable on the right boundary. However, the Deep Distributional Learning model is still the best predictor especially for the non-linear function of x.

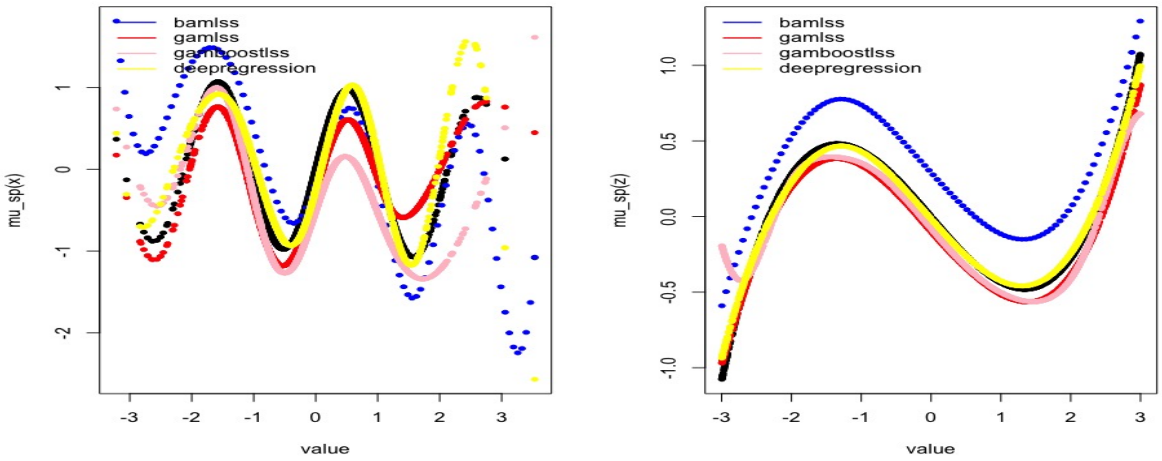


Figure 5.15: Averaged non-linear estimation by each model (with noises, Poisson distribution, two-dimensional case)

The MSE of non-linear estimation by Deep Distributional Learning model is at the lowest level among all four models and apparently outperforms the GAMLSS model under this situation. And variance of estimation by BAMLSS is larger than in two-dimensional case while others remain unchanged.

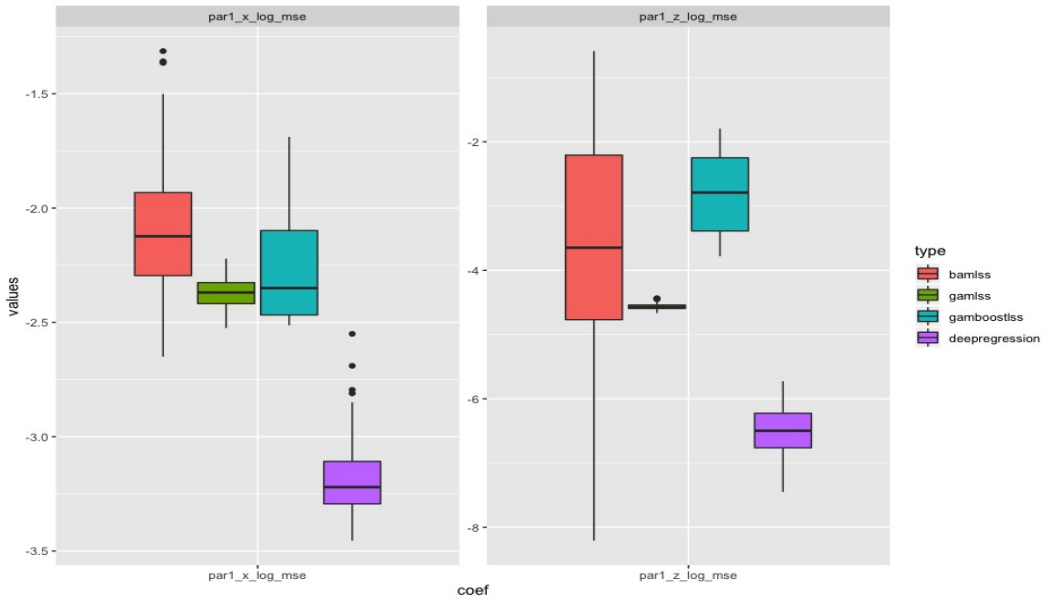


Figure 5.16: Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i) - f(x_i)]^2)$> of non-linear estimation by each model (with noise, Poisson distribution, two-dimensional case)

Deep-regression model converges regardless of effects of noises and provides reliable credible intervals through variance inference as shown below 5.17:

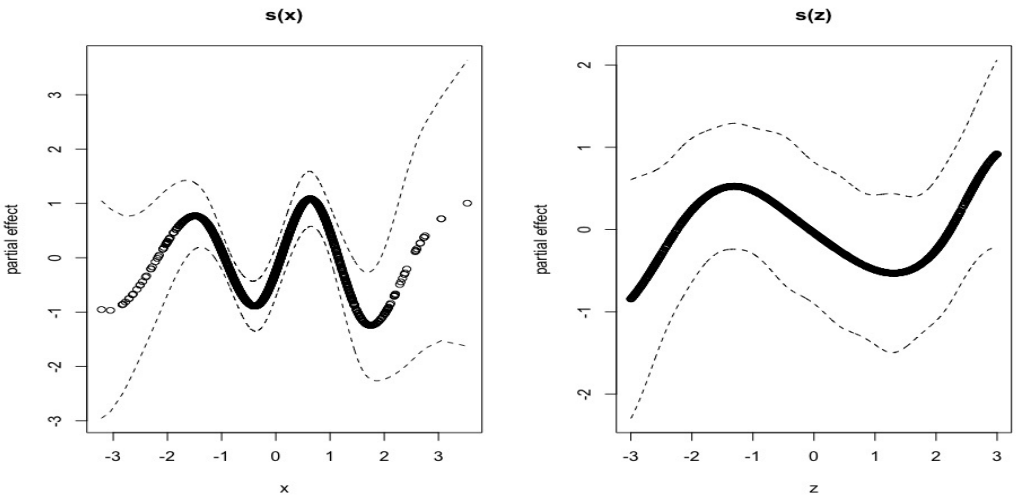


Figure 5.17: Variational inference on estimation of non-linear function by Deep Distributional Learning model (with noises, Poisson distribution, two-dimensional case)

And results in figure 5.18 leads to the conclusion that, DGAMLSS may outperform in estimation based on data with noises and therefore have stronger robustness than other models.

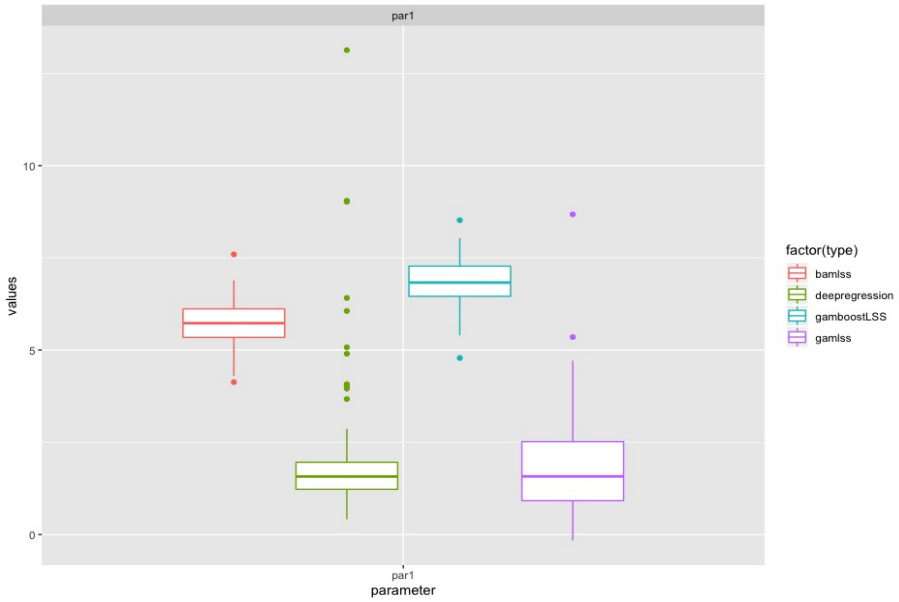


Figure 5.18: MSE of estimated parameters on test dataset (with noises, Poisson distribution, two-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS)

Lastly, we move onto 3-dimensional case in which non-linear function of variable v is included into parameterization function and follow the same procedure as in 2-dimensional case.

$$log(\lambda) = 2 - x + f_{1,x}(x) + f_{1,z}(z) + f_{1,v}(v)$$

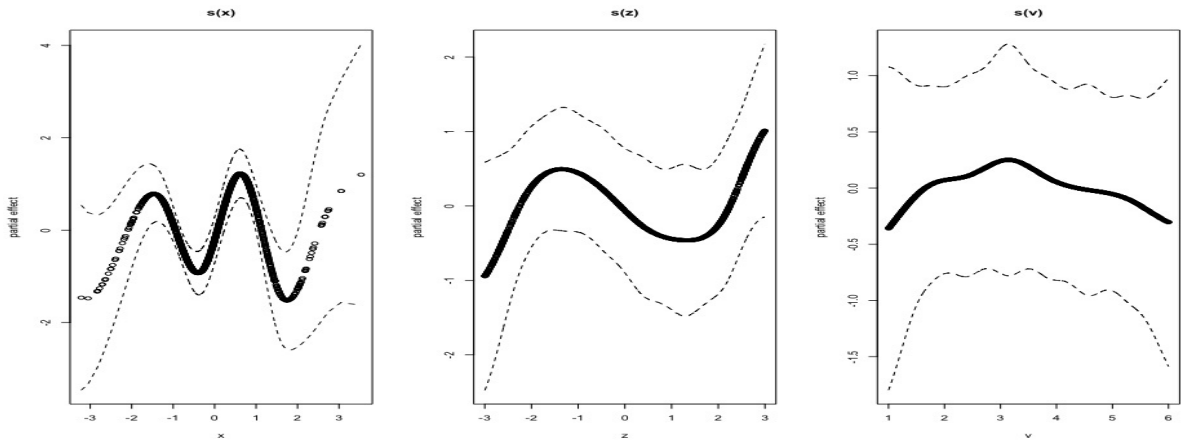


Figure 5.19: Variational inference on estimation of non-linear function by Deep Distributional Learning model (Poisson distribution, three-dimensional case)

As shown in figure 5.20, the GAMLSS and DGAMLSS provide best estimation of parameter on test data , where results by the other tow models are at the same level.

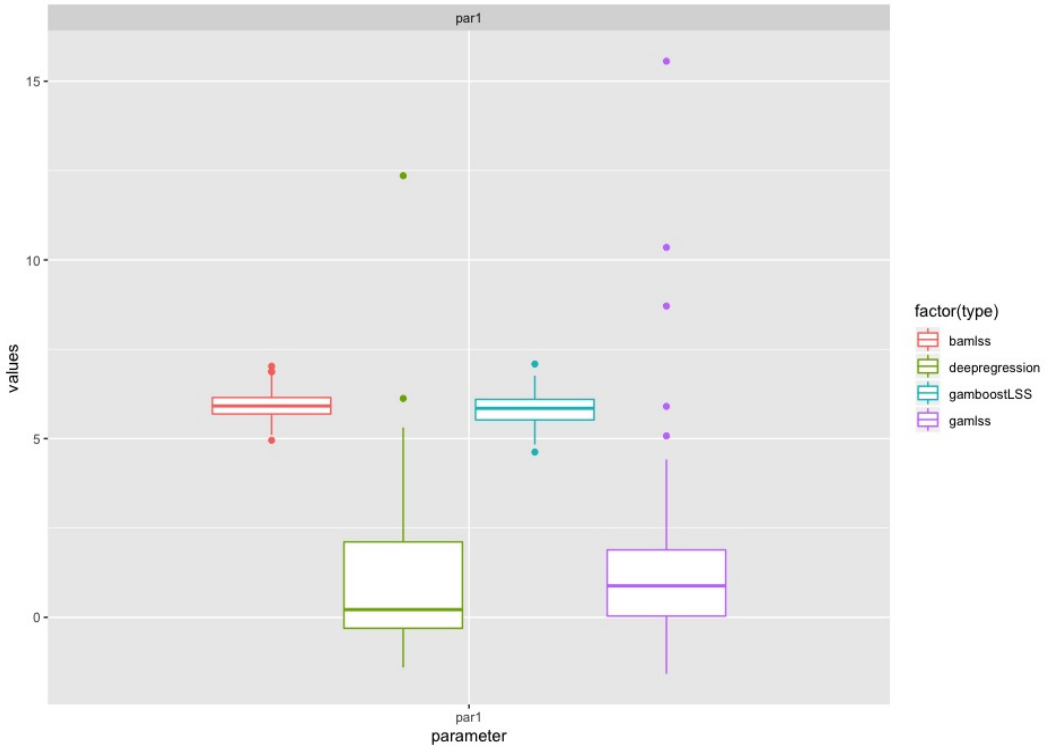


Figure 5.20: MSE of estimated parameters on test dataset (Poisson distribution, three-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS)

The GAMLSS model remains to work well for linear estimation while the estimation of Deep Distributional Learning model is acceptable. And the pattern of non-linear estimation is similar to two-dimensional case. Apparently estimation of nonlinear function of v by Deep Distributional Learning model is not ideally smooth while the other two estimations are more satisfying.

coefficients	par1_inter	par1_lin
bamlss	-1.3210794	-2.664920
gamlss	-4.4362885	-11.945086
gamboostLSS	-0.9237846	-5.321727
deepregression	-1.7090666	-6.444573

Table 5.5: MSE(log) of estimated linear coefficients by each model (Poisson distribution, three-dimensional case)

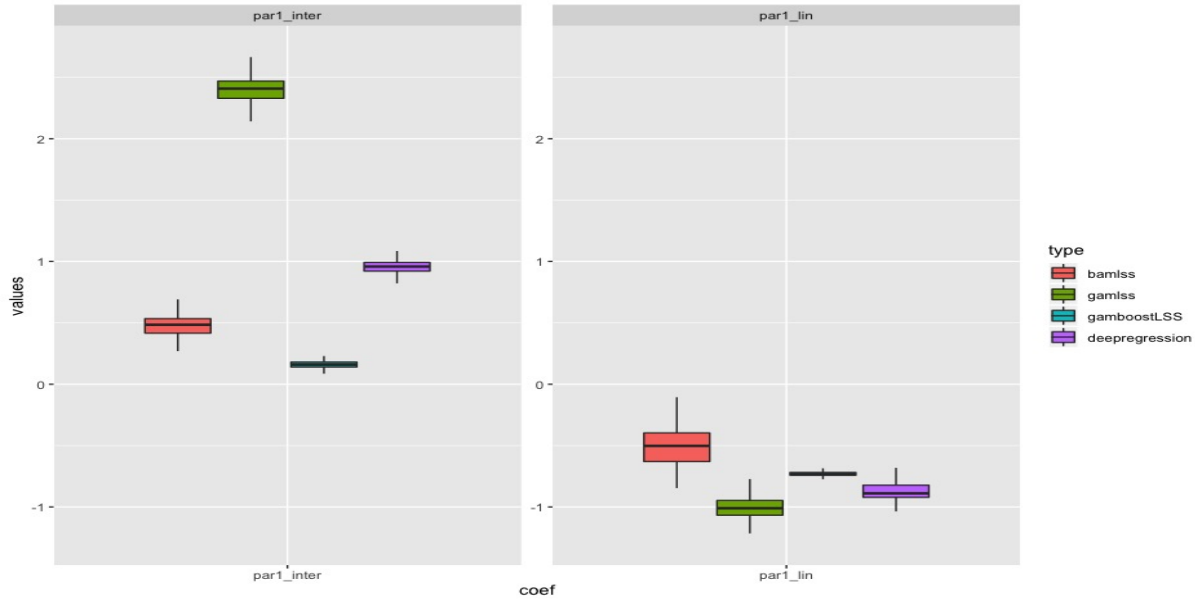


Figure 5.21: Linear Estimation (Poisson distribution, three-dimensional case) by each model in 100 iteration (figure on the left: intercept in parameter 1; figure on the right: linear coefficient in parameter 1; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression)

Figure 5.22 shows insignificant difference between results of estimated non-linear functions in average in three-dimensional case and one in two-dimensional case.

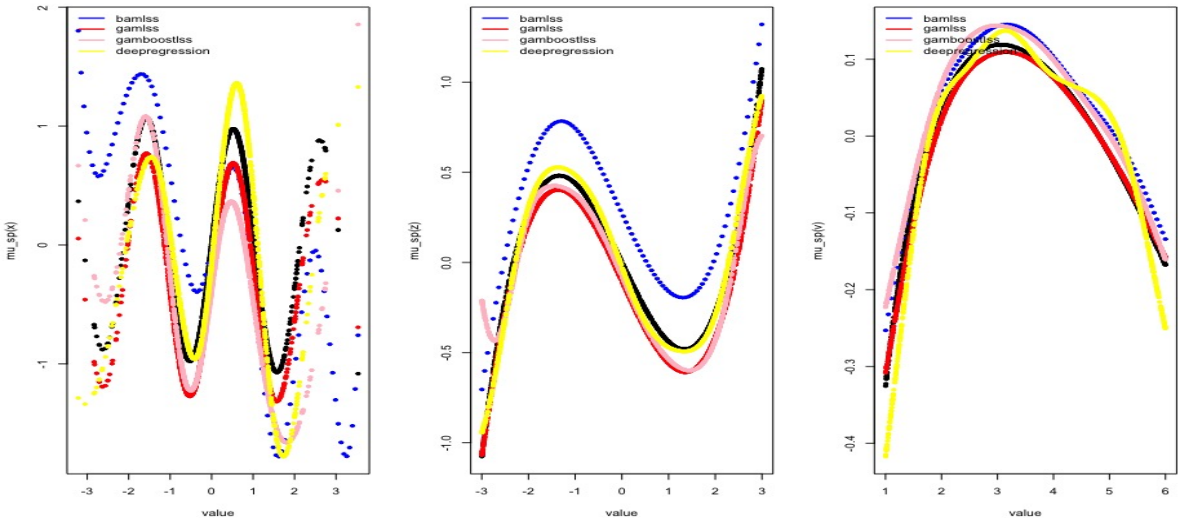


Figure 5.22: Averaged non-linear estimation by each model (Poisson distribution, three-dimensional case)

However, in figure 5.23, the MSE of estimation on function of variable v from Deep Distributional Learning model is inferior to that from gamlss, which means Deep Distributional Learning model may underperform in three dimensional case.

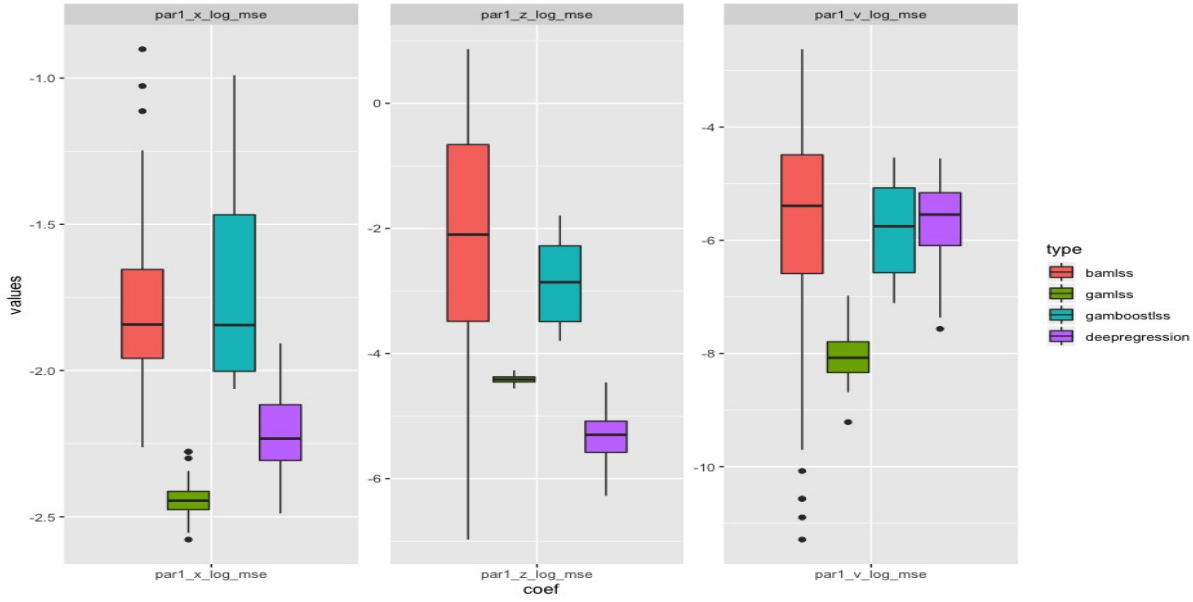


Figure 5.23: Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i) - f(x_i)]^2)$> of non-linear estimation by each model (Poisson distribution, three-dimensional case)

Two-parametric distribution: Normal Distribution

For two-parametric distributions, we mainly focus on the estimation for normal distributional dataset in this chapter, and this article does not have room to address other results, including lognormal, logistic, Gamma and negative binomial distributions, where patterns of estimation results are nearly similar to those presented here.

From figure 5.6, MSE of estimated parameters by all models are at the same level, where DGAMLSS comparatively shows advantage here. Additionally, the variance estimation of the parameter *scale* by Deep Distributional Learning is larger.

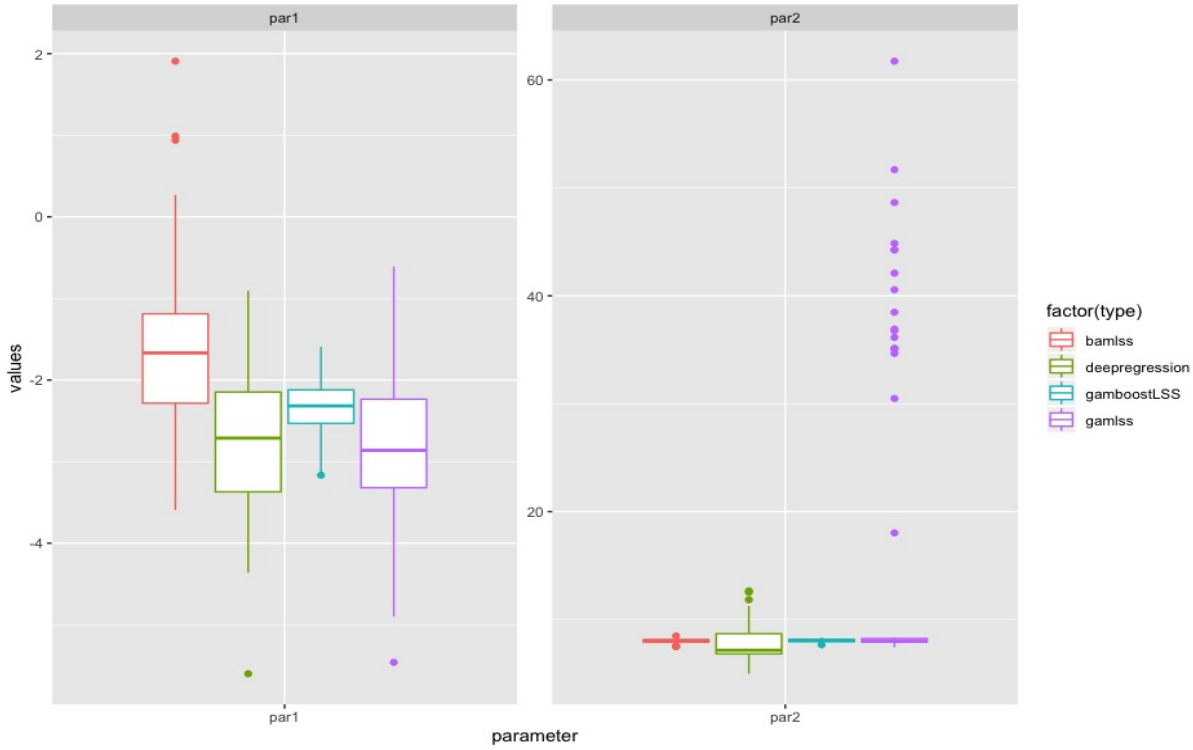


Figure 5.24: MSE of estimated parameters on test dataset (Normal distribution, two-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS)

As observed in figure 5.25 and table 5.6, the estimation of linear coefficients differs significantly among models, and the true values are 2(top-left), -1(top-right), -1(bottom-left) and 1(bottom-right). GAMLSS provided nearly perfect estimation on all of four coefficients. Estimation by Deep distribution Learning model is significantly biased for some parameters.

coefficients	par1_inter	par1_lin	par2_inter	par2_lin
bamlss	-1.321966	-2.716530	-4.480418	-4.105345
gamlss	-6.360862	-9.880125	-3.224452	-5.611579
gamboostLSS	-1.304477	-5.881143	-4.468784	-7.678759
deepregression	-1.489669	-3.266698	-5.463922	-3.559429

Table 5.6: MSE(log) of estimated linear coefficients by each model (Normal distribution, two-dimensional case)

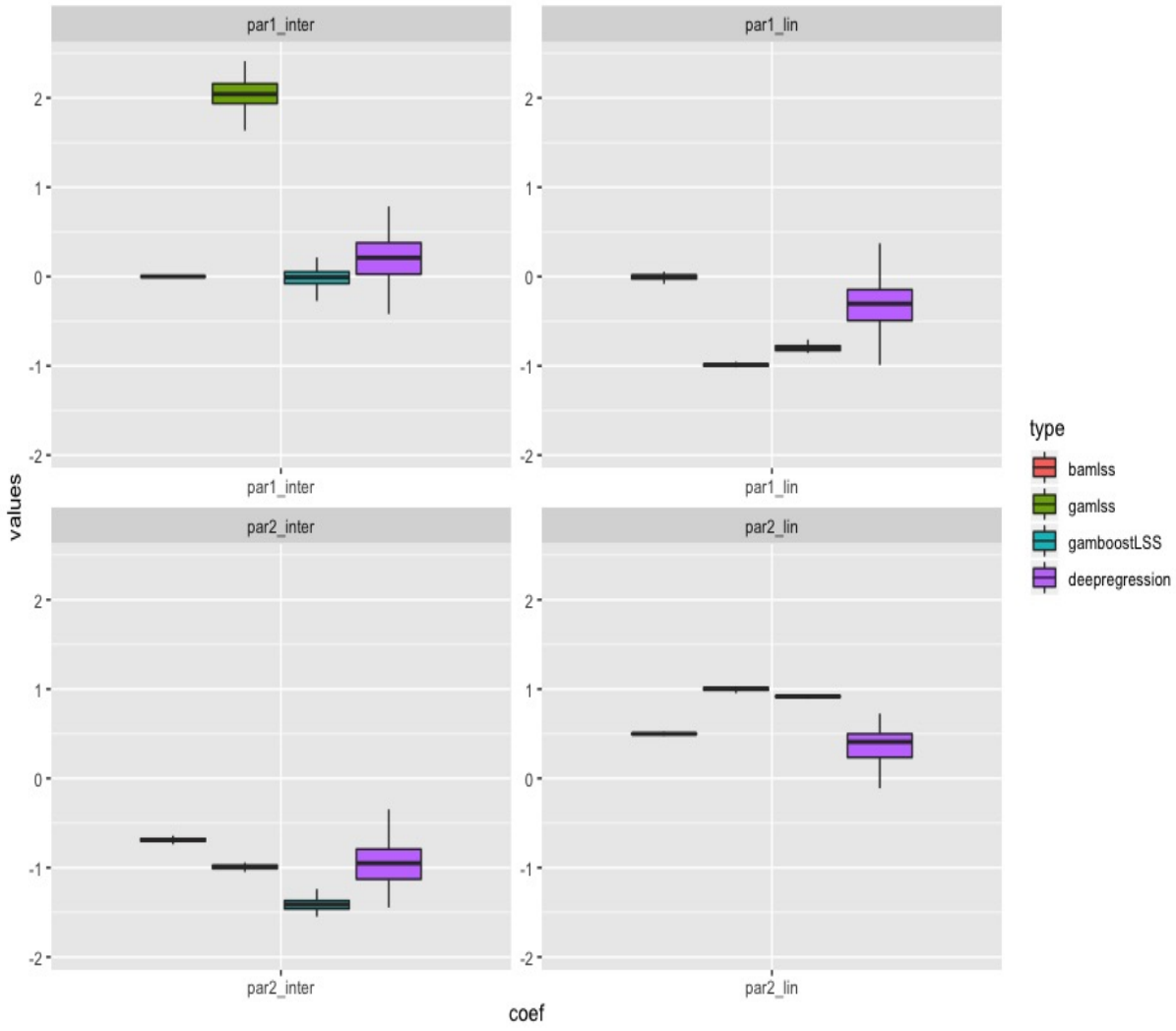


Figure 5.25: Linear Estimation (Normal distribution, two-dimensional case) by each model in 100 iteration (figure on the top-left: intercept in parameter 1; figure on the top-right: linear coefficient in parameter 1; figure on the bottom-left: intercept in parameter 2; figure on the bottom-right: linear coefficient in parameter 2; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression)

And the pattern of MSE of estimated non-linear function shown in figure 5.26 and 5.27 is slightly different from one in Poisson distribution. All of four model perform almost similar in non-linear estimation regarding to accuracy. Estimation of the scale parameter by GAMLSS doesn't appear to be stable as previous. DGAMLSS doesn't show any advantage here. Results of estimation on two-dimensional dataset with noises and three-dimensional dataset are presented in Appendix, from figure 7.1 to 7.4 and from figure 7.5 to 7.8 respectively.

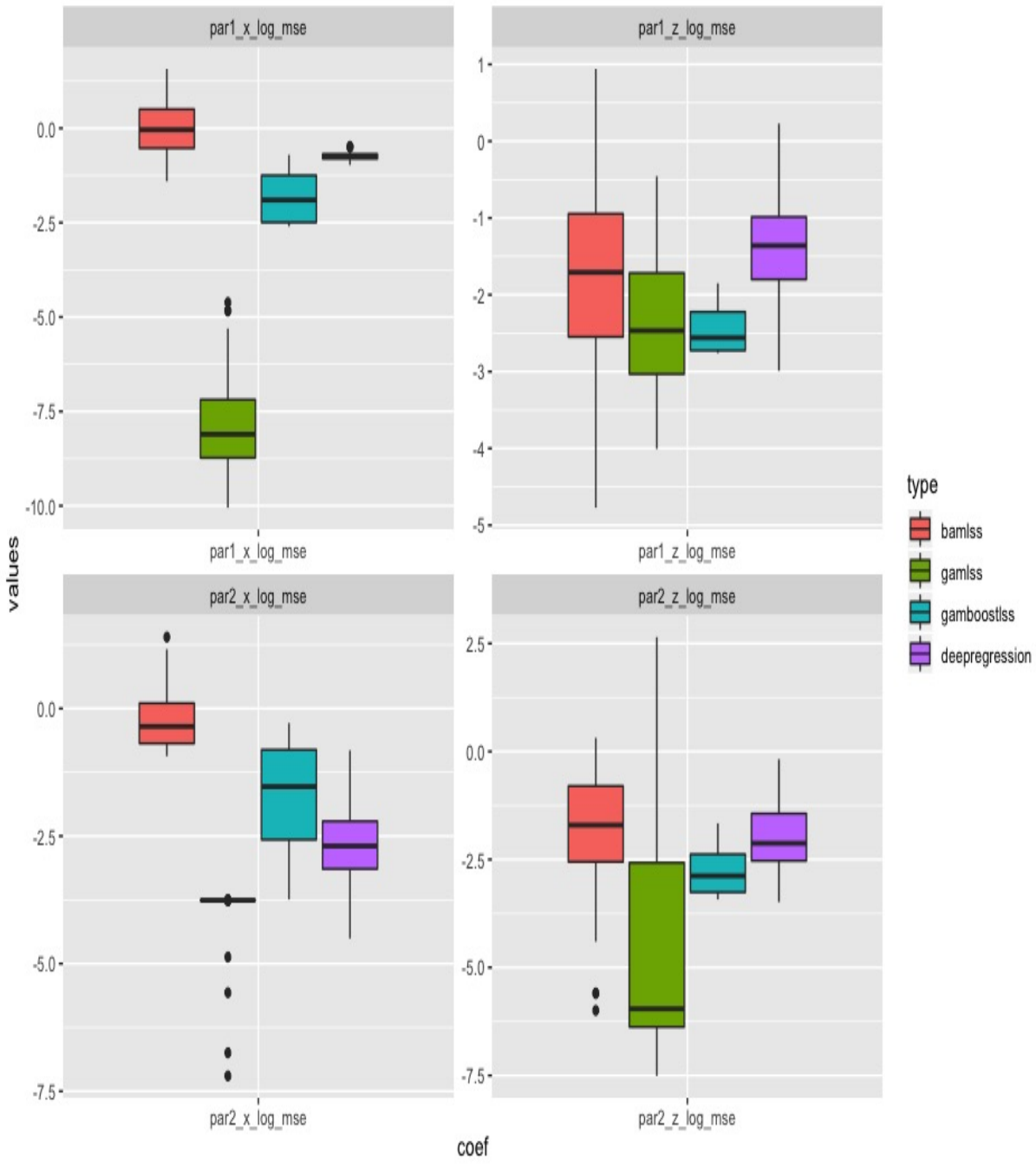


Figure 5.26: Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i) - f(x_i)]^2)$> of non-linear estimation by each model (Normal distribution, two-dimensional case)

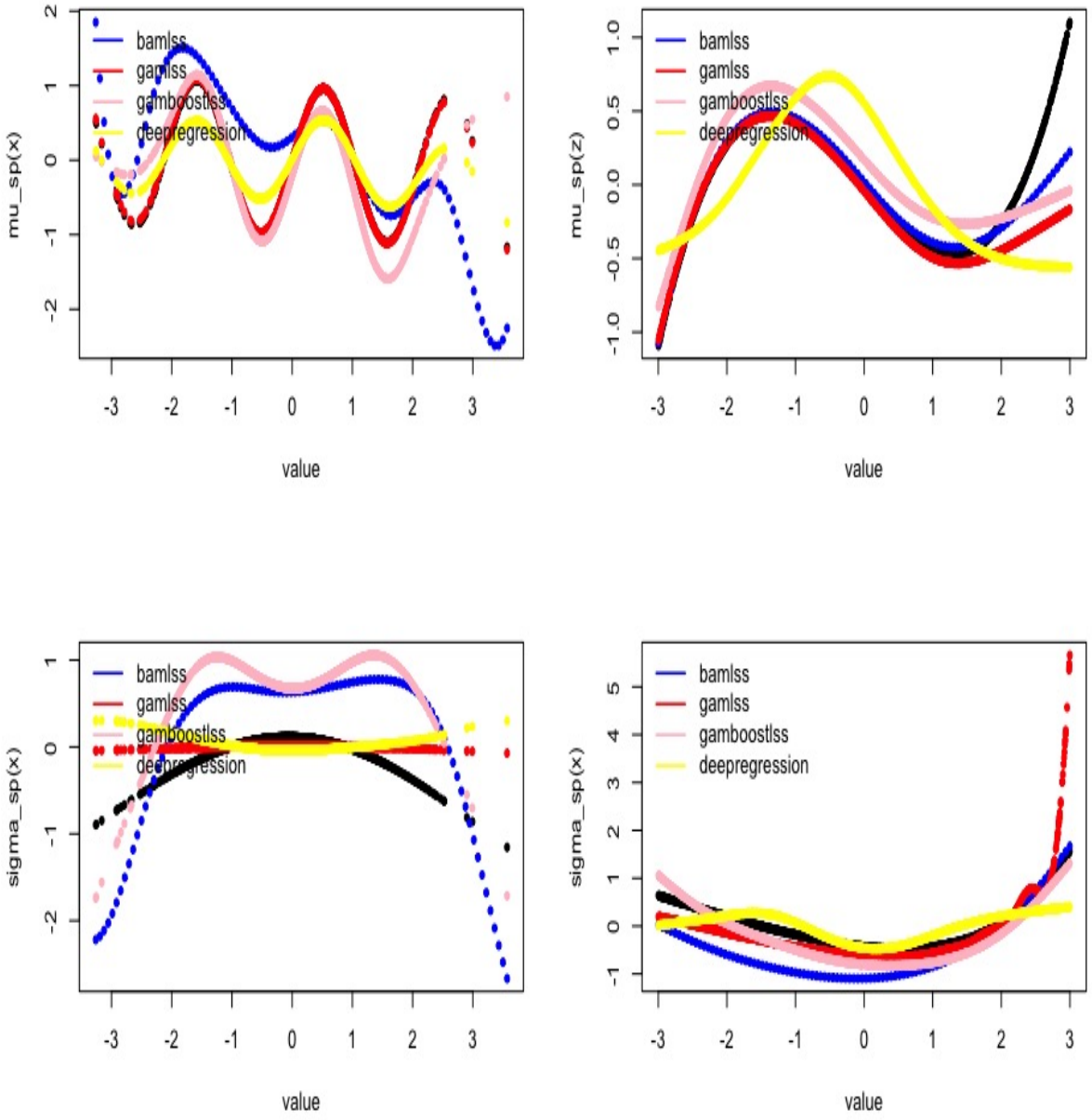


Figure 5.27: Averaged non-linear estimation by each model (Normal distribution, two-dimensional case)

5.2 Mixture Distribution: Zero-inflated Poisson

Zero-inflated Poisson Distribution

Data Generating

Zero modified counting distribution is a family of non-negative counting distributions where the probability of occurrence of zero is altered. There are two cases for such modification, one is zero-inflated model by increasing the probability of value zero and the other is called zero-truncated model by removing the occurrence of the model. In the paper, we focus on the first case and present an example by the zero-inflated Poisson distribution.

There are two parts of the zero-inflated model, the binary selection process κ under a Bernoulli distribution $Bernoulli(1-\pi)$ and the counting data $\tilde{y}$ under non-negative discrete distribution p. Then the zero-inflated variable could be interpreted as the product of these two independent variables as $y = \tilde{y} \cdot \kappa$. The distribution function of the zero-inflated variable conditioned on covariates $v$is then:

$$p(y|v) = \pi \, \mathbb{1}_{\{0\}}(y) + (1 - \pi) \, \tilde{p}(y|v)$$

For the zero-inflated Poisson model, we assume that the counting data $\tilde{y}$ is under Poisson distribution $Pois(\lambda)$ with density $\tilde{p}(\tilde{y}) = \frac{\lambda^{\tilde{y}} e^{-\lambda}}{\tilde{y}!}$. Thereafter, there are two parameters of our interest in zero-inflated Poisson distribution λ and π. For those two parameters we have respectively logit and log link functions:

$$\eta^{\lambda} = log(\lambda) = -0.5 + 0.5x + f_{1,x}(x) + f_{1,z}(z)$$
$$\eta^{\pi} = logit(\pi) = -2 - 0.5z + f_{2,x}(x) + f_{2,z}(z)$$

As for BAMLSS model, we have the elements of score vectors for both the zero-inflation and the Poisson parts:

$$v^{\lambda} = \frac{\pi\lambda}{\pi+(1-\pi)exp(-\lambda)} \mathbb{1}_{\{0\}}(y) + (y - \lambda)$$
$$v^{\pi} = \frac{\pi}{\pi+(1-\pi)exp(-\lambda)} \mathbb{1}_{\{0\}}(y) - \pi$$

and the working weights:

$$w^{\lambda} = \frac{\lambda(1-\pi)(\pi+(1-\pi)exp(-\lambda)-exp(-\lambda)\lambda\pi)}{\pi+(1-\pi)exp(-\lambda)}$$
$$w^{\pi} = \frac{\pi^2(1-\pi)(1-exp(-\lambda))}{\pi+(1-\pi)exp(-\lambda)}$$

Estimation result would be found in Appendix from figure 7.9 to 7.16.

6 Conclusion & Discussion

By introducing unstructured estimator – neural network – into GAMLSS, DGAMLSS performs generally better than other models in respect of accuracy in estimation on parameters of distributional function. Such better performance doesn't vanish by different distributional assumption. For more complicated distributional functions such as mixture distribution – Zero-inflated Poisson distribution, DGAMLSS provides more accurate and less time-consuming results. Convergence of model fitting process is guaranteed in most distributional cases.

Another advantage of DGAMLSS is that, model could stably converge where the scale of true values is large (lognormal distribution) or extreme values are included (unexpected large number of zeros in Zero-inflated Poisson distribution).

However, such advantage of DAMLSS is on the cost of dissatisfactory results in stability. In our hypothesis, such instability is partially due to the simple structure of neural network utilized in our model. A more complicated neural network may contribute to improvement in consistence of estimation, but could cause the issue of overfitting or time-consuming convergence as well. Therefore, new orthogonalization mechanism might be required to tackle such dilemma if sophisticated neural network structure is deployed into DGAMLSS.

On top of that, there is no strong evidence showing that, DGAMLSS works better in distinguishing structure linear and non-linear terms. Simulation work on classical distributional datasets gives evidence that all of four models are able to accurately estimate non-linear functions in most cases, but GAMLSS gives better estimation on linear coefficients. In contrast, estimation on linear coefficients by DGAMLSS is less reliable especially in most two-parametric distributional cases. The introduction of neural network may undermine DGAMLSS's performance on estimation of both linear and non-linear terms. This might result from the L2 orthogonalization mechanism. Similar issue occurs when we try to simulate a high-dimensional space with basic vectors. Ideally we have basic vectors $(1, 0, 0)$, $(0, 1, 0)$ and $(1, 0, 0)$ for a three-dimensional space, but any three of noncoplanar vectors are sufficient. Therefore, a sophisticated mechanism should be designed for choice of basics in addition to orthogonalization constrain. Hypothetically, idea of Instrumental Variable Regression[45] or Two-Stage Least Squares[46][47] is probably a solution for this issue.

However, GAMLSS couldn't provided accurate estimation after adding noises into date, while estimation by Deep Distributional Learning Model is thereby relatively reliable. Therefore, we may head to the conclusion that Deep distributional Learning Model provides clear separation between structured and unstructured predictors, and the deep neural network part is able to remove noise from the data for better estimation. In this case, we may assume that DGAMLSS might be more powerful in practical data analysis where noises are inevitable in data collecting process.

7 Appendix

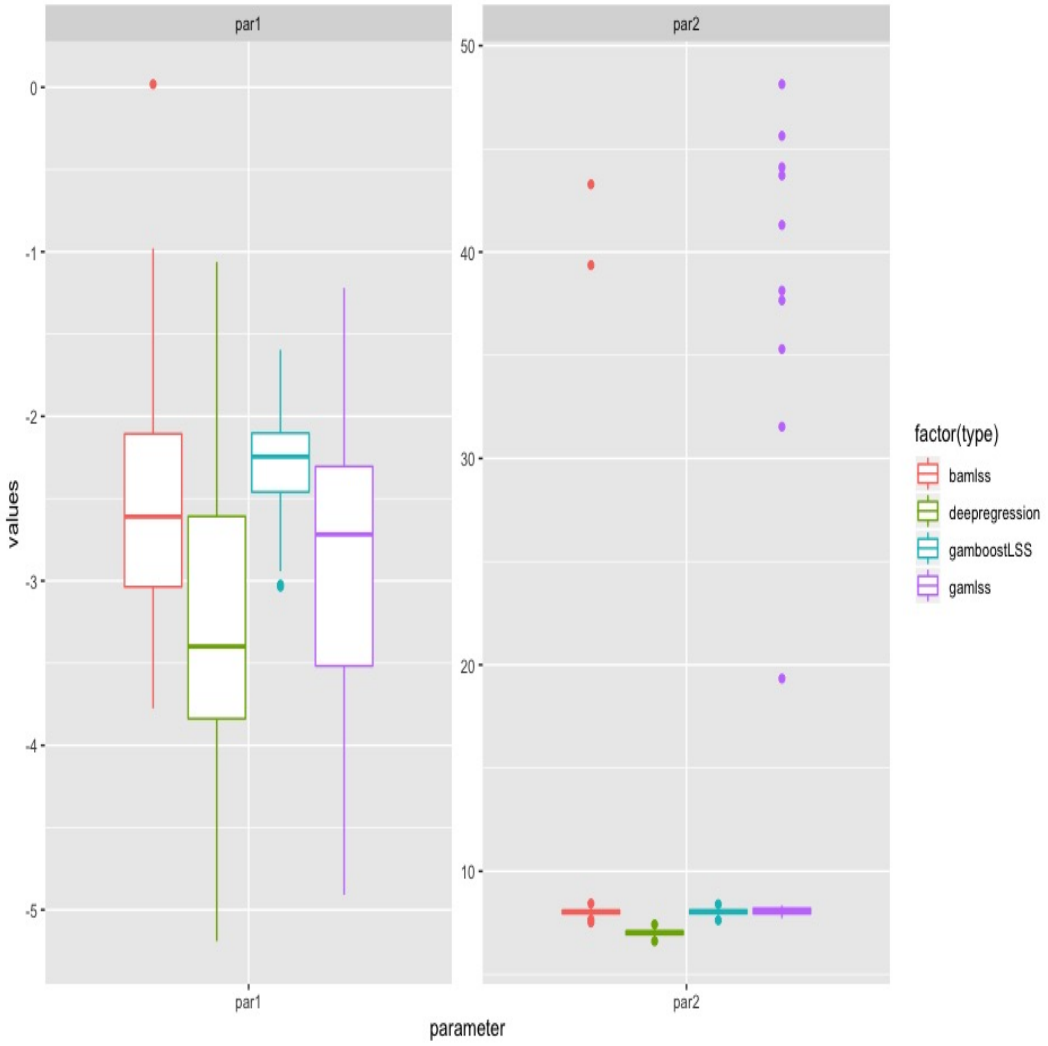


Figure 7.1: MSE of estimated parameters on test dataset (with noises, Normal distribution, two-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS)

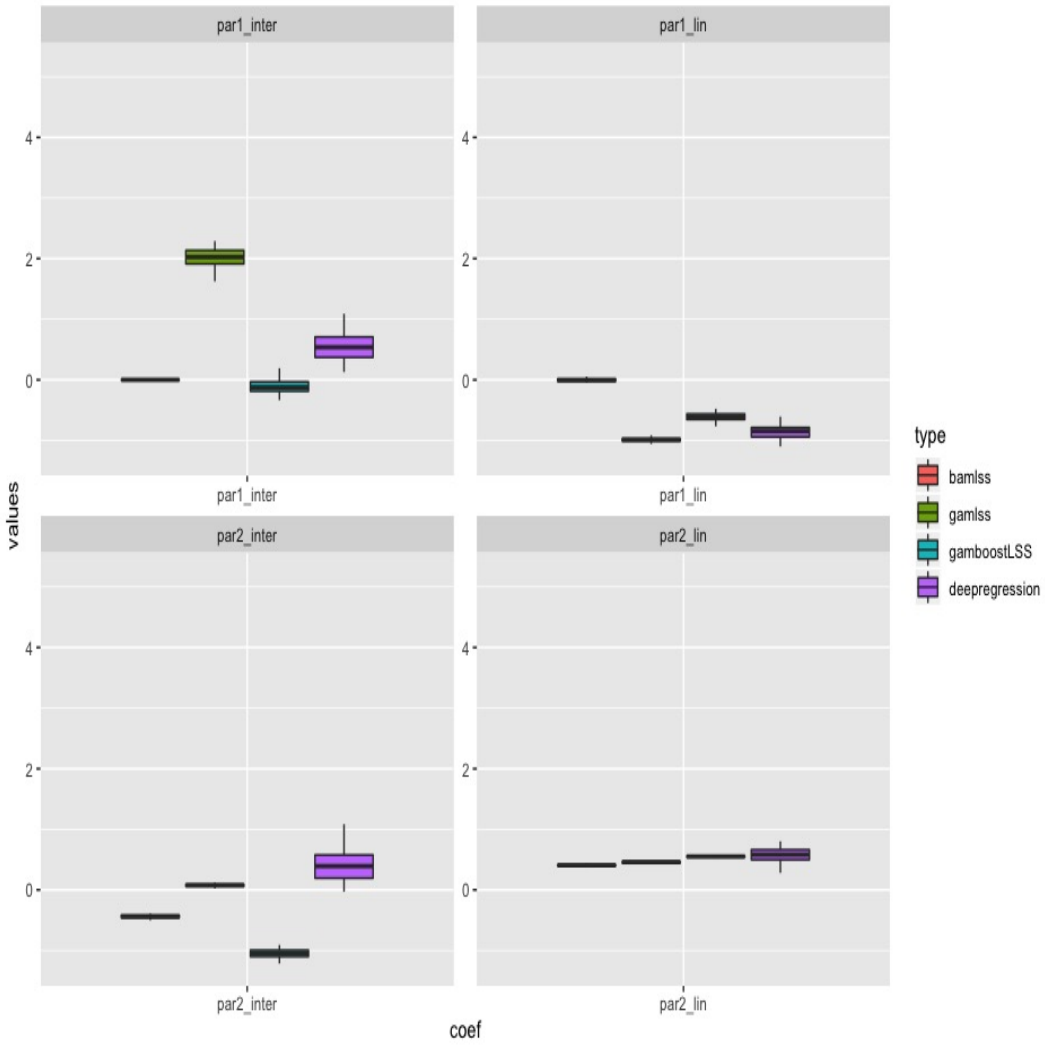


Figure 7.2: Linear Estimation (with noises, Normal distribution, two-dimensional case) by each model in 100 iteration (figure on the top-left: intercept in parameter 1; figure on the top-right: linear coefficient in parameter 1; figure on the bottom-left: intercept in parameter 2; figure on the bottom-right: linear coefficient in parameter 2; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression)

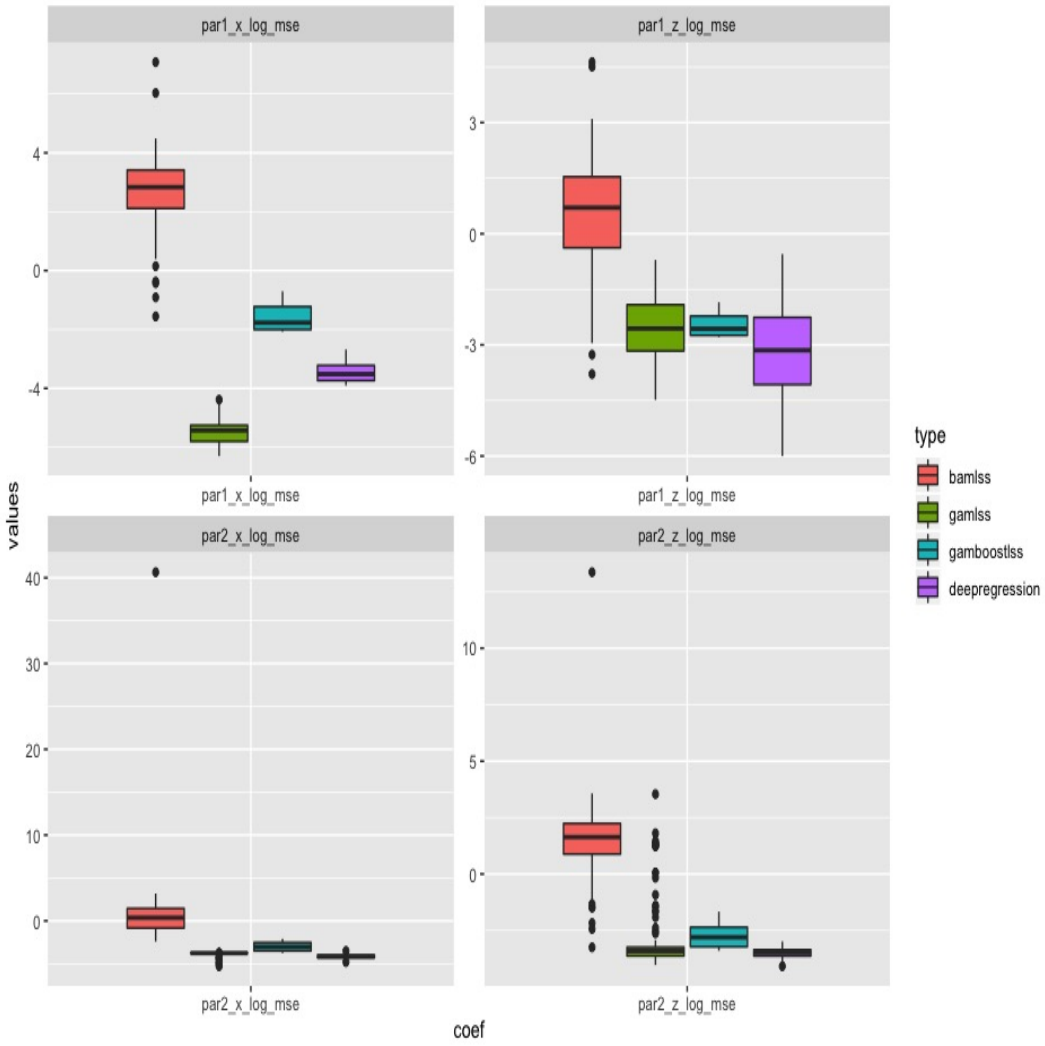


Figure 7.3: Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i) - f(x_i)]^2)$> of non-linear estimation by each model (with noises, Normal distribution, two-dimensional case)

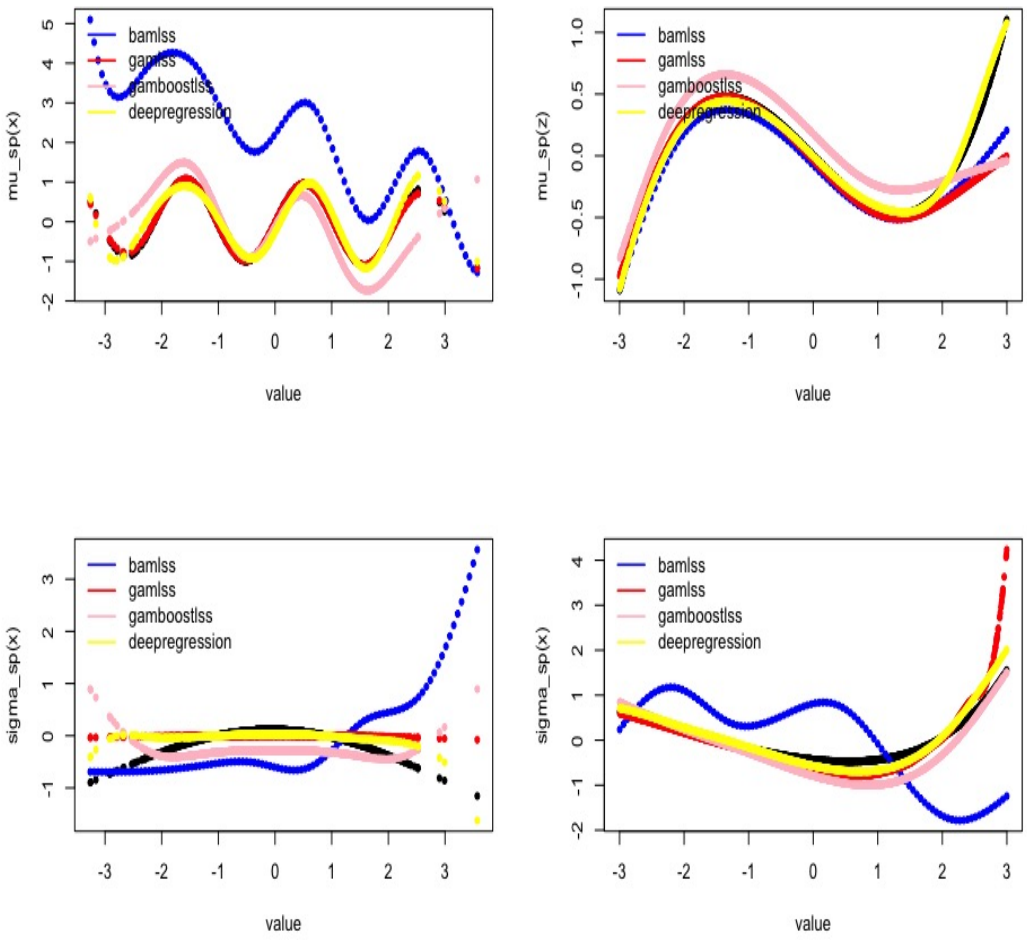


Figure 7.4: Averaged non-linear estimation by each model (with noises, Normal distribution, two-dimensional case)

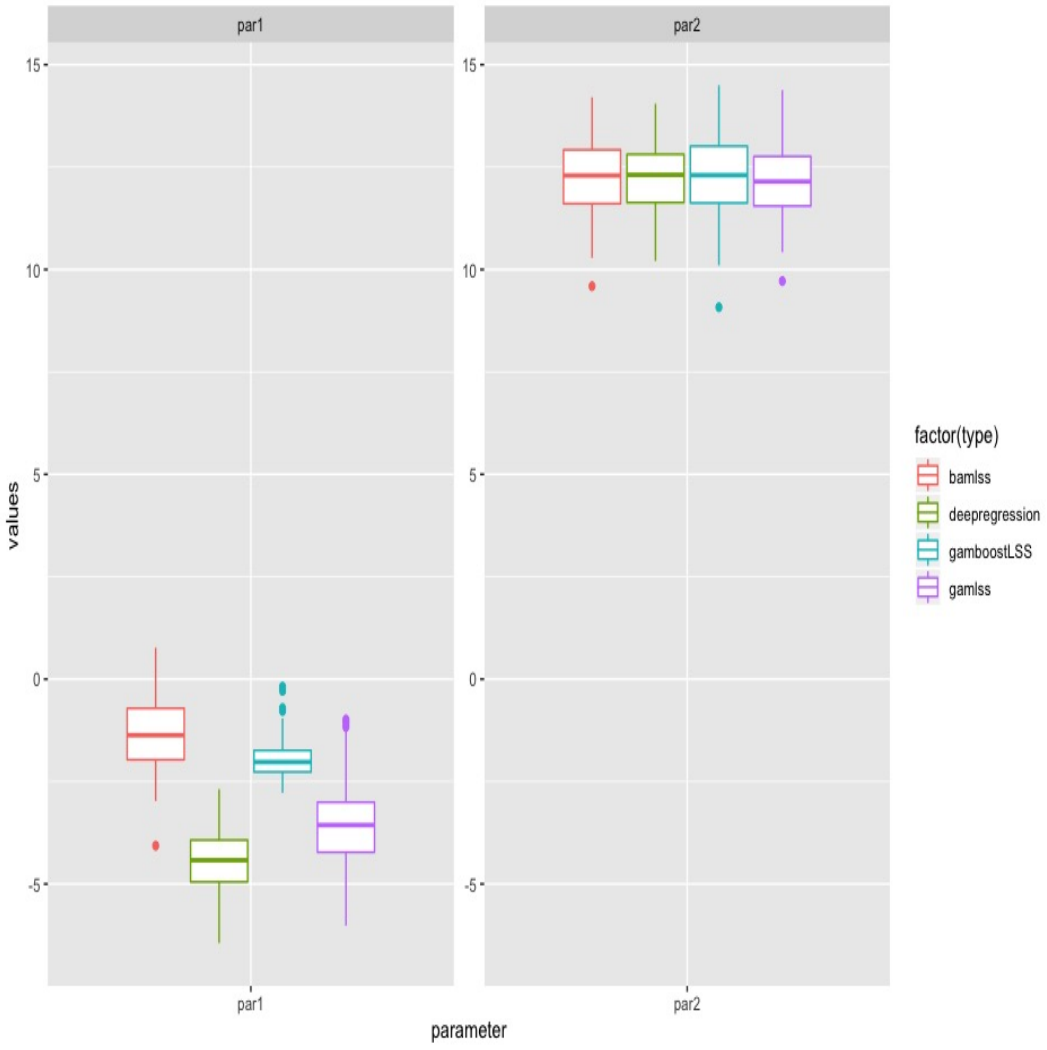


Figure 7.5: MSE of estimated parameters on test dataset (Normal distribution, three-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS)

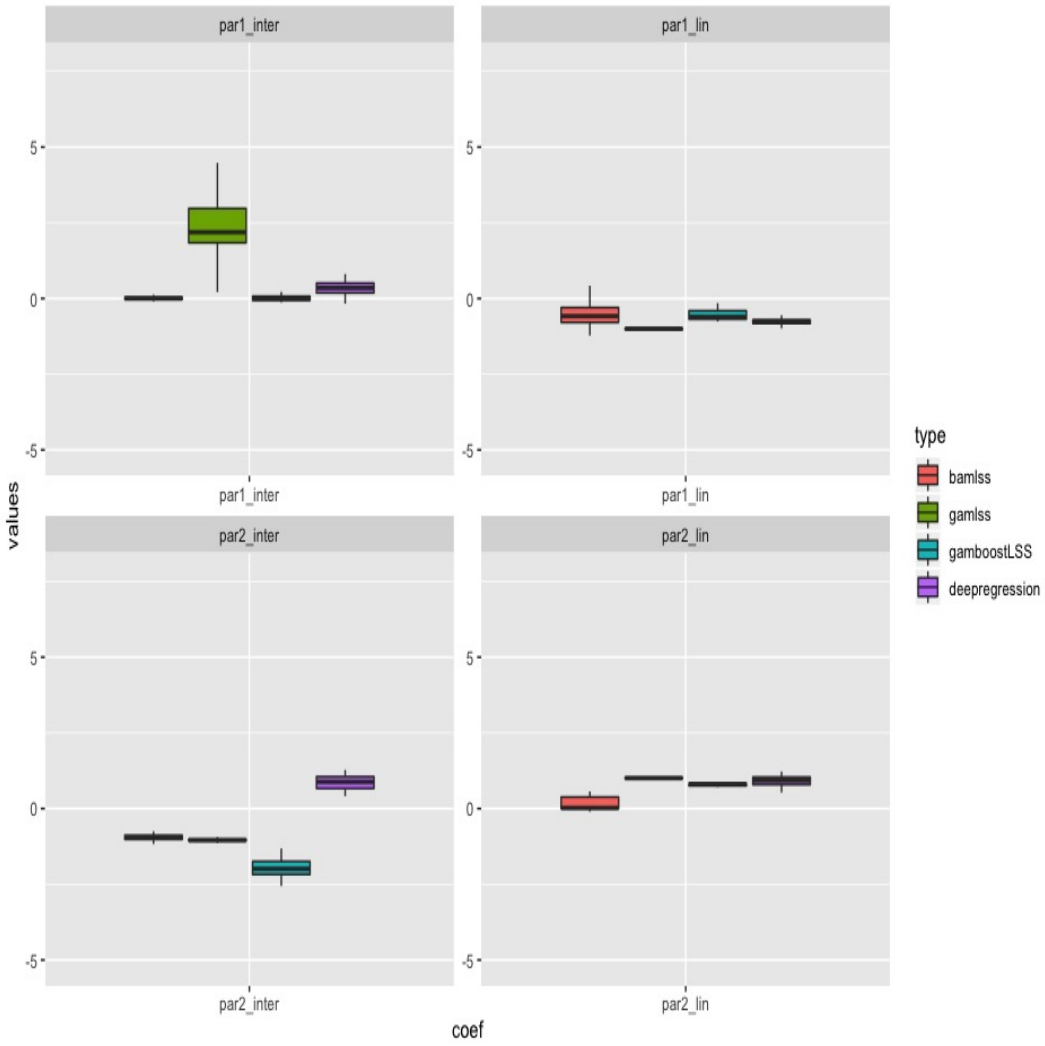


Figure 7.6: Linear Estimation (Normal distribution, three-dimensional case) by each model in 100 iteration (figure on the top-left: intercept in parameter 1; figure on the top-right: linear coefficient in parameter 1; figure on the bottom-left: intercept in parameter 2; figure on the bottom-right: linear coefficient in parameter 2; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression)

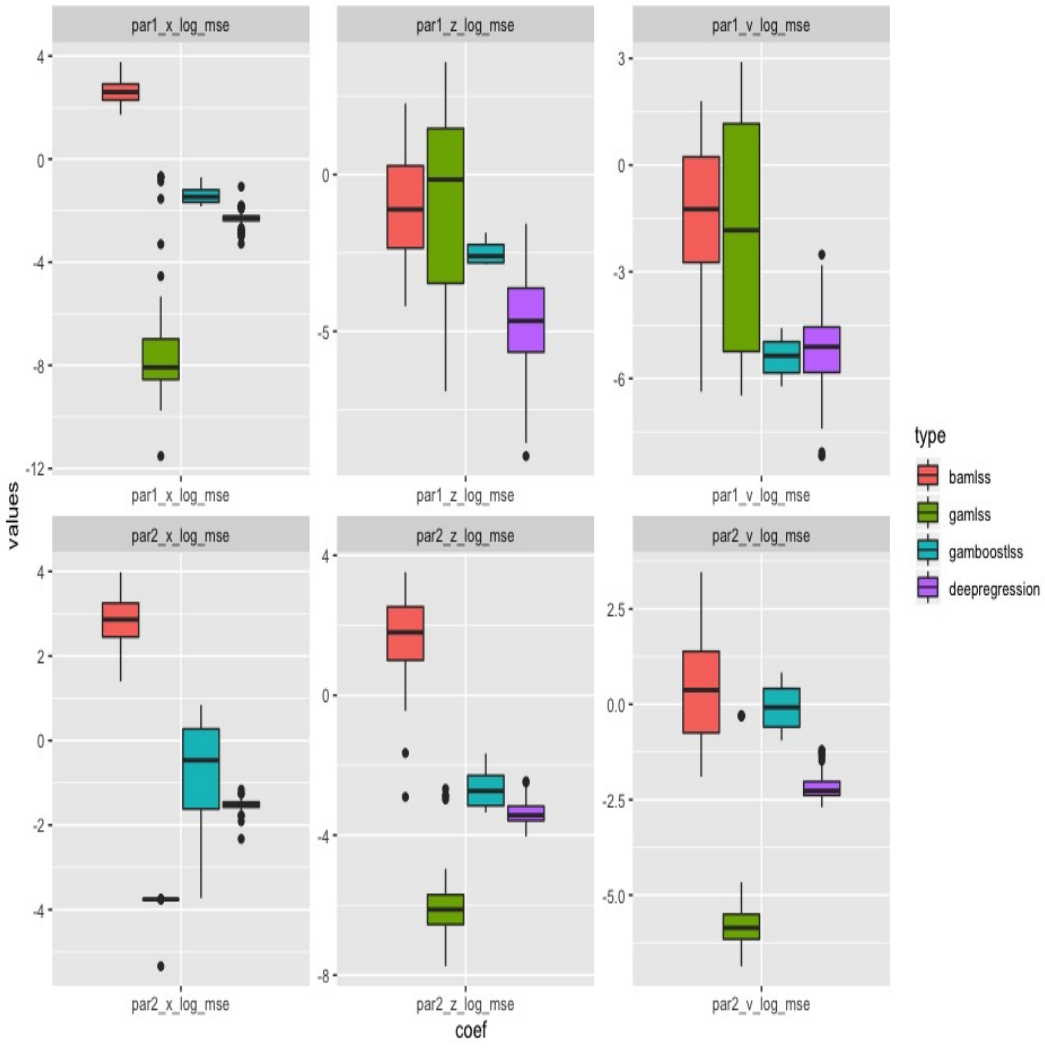


Figure 7.7: Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i) - f(x_i)]^2)$> of non-linear estimation by each model (Normal distribution, three-dimensional case)

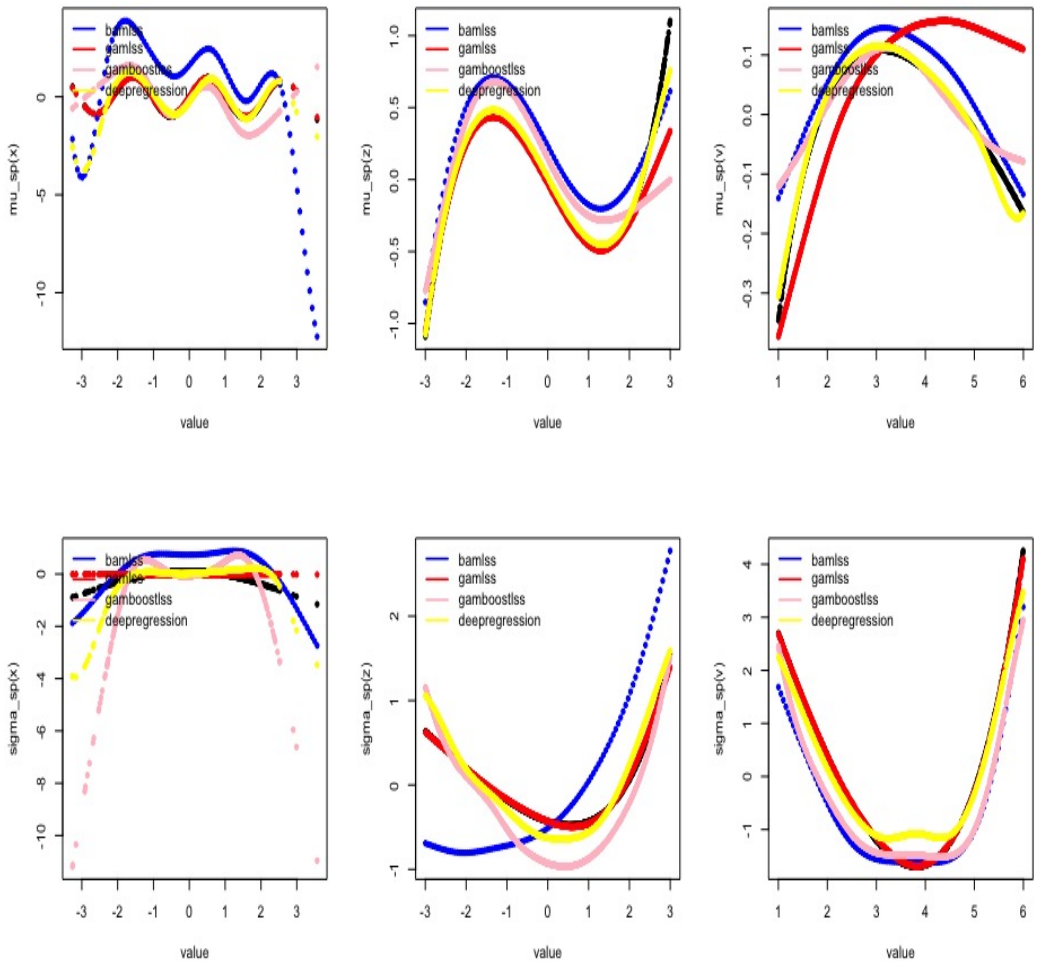


Figure 7.8: Averaged non-linear estimation by each model (Normal distribution, three-dimensional case)

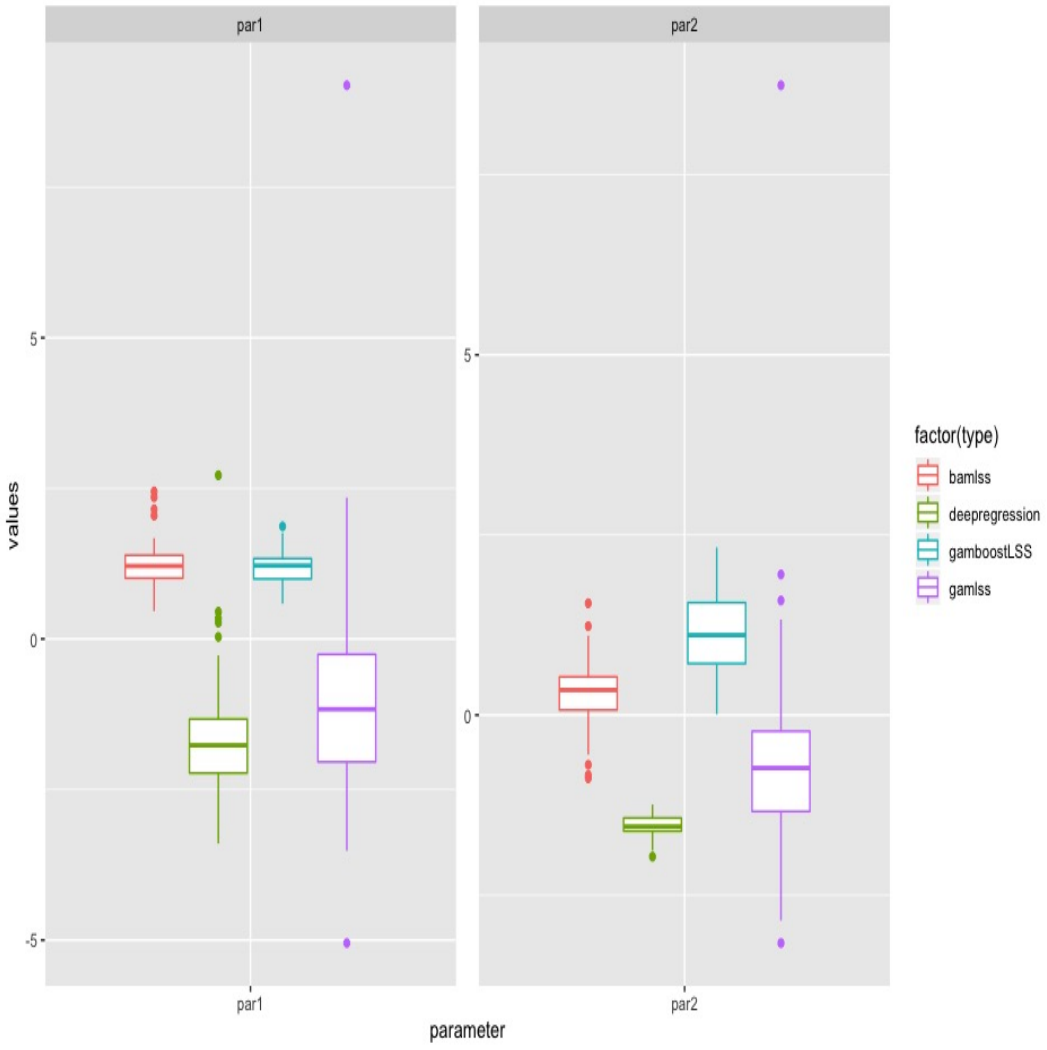


Figure 7.9: MSE of estimated parameters on test dataset (ZIP distribution, two-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS)

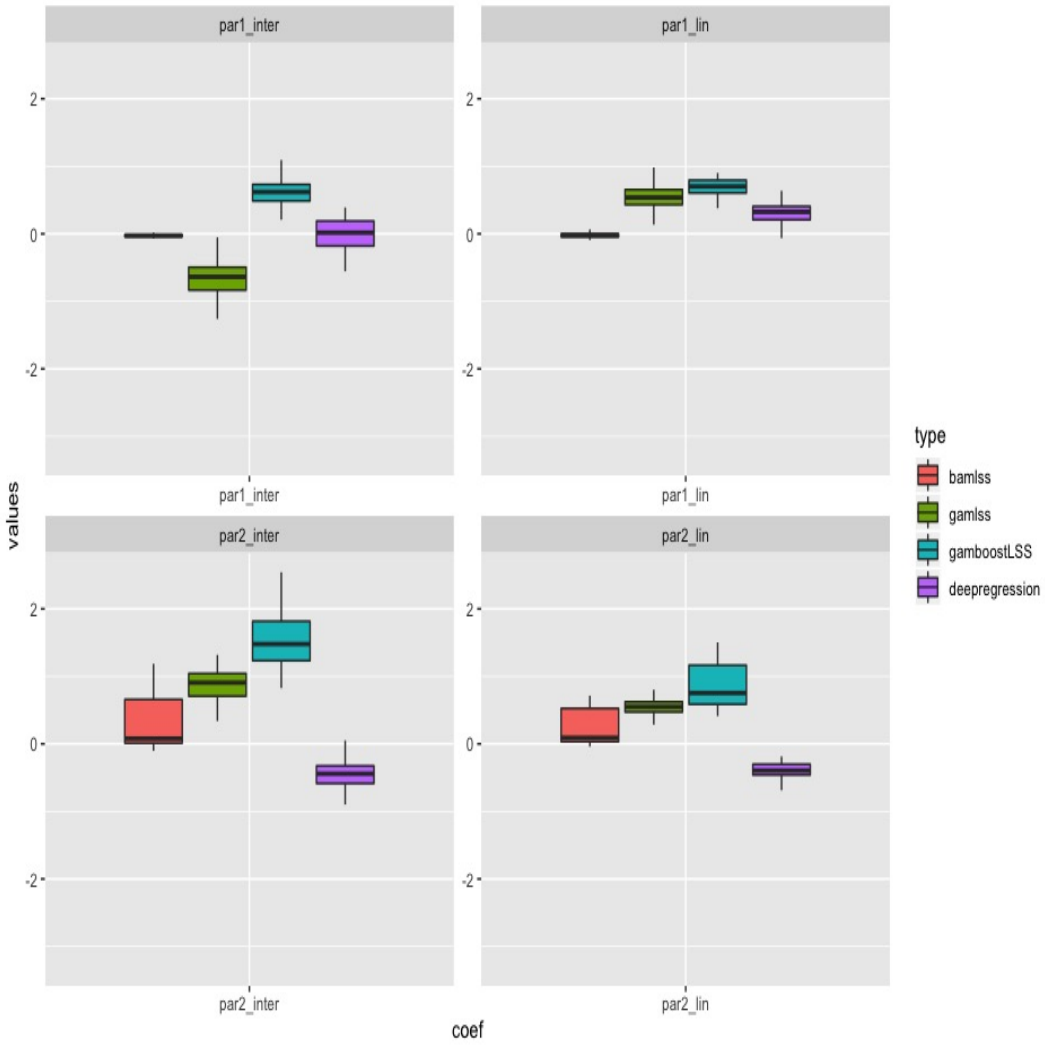


Figure 7.10: Linear Estimation (ZIP distribution, two-dimensional case) by each model in 100 iteration (figure on the top-left: intercept in parameter 1; figure on the top-right: linear coefficient in parameter 1; figure on the bottom-left: intercept in parameter 2; figure on the bottom-right: linear coefficient in parameter 2; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression)

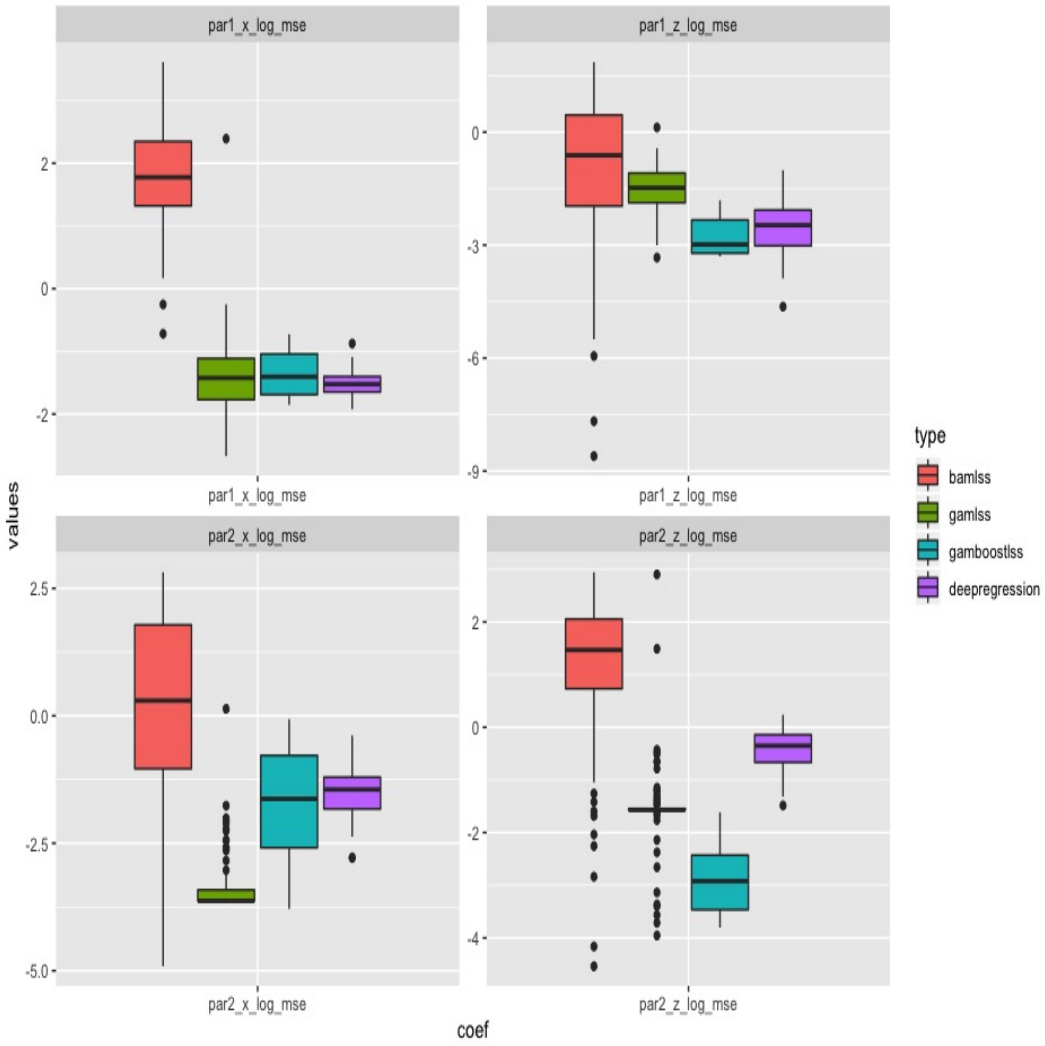


Figure 7.11: Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i) - f(x_i)]^2)$> of non-linear estimation by each model (ZIP distribution, two-dimensional case)

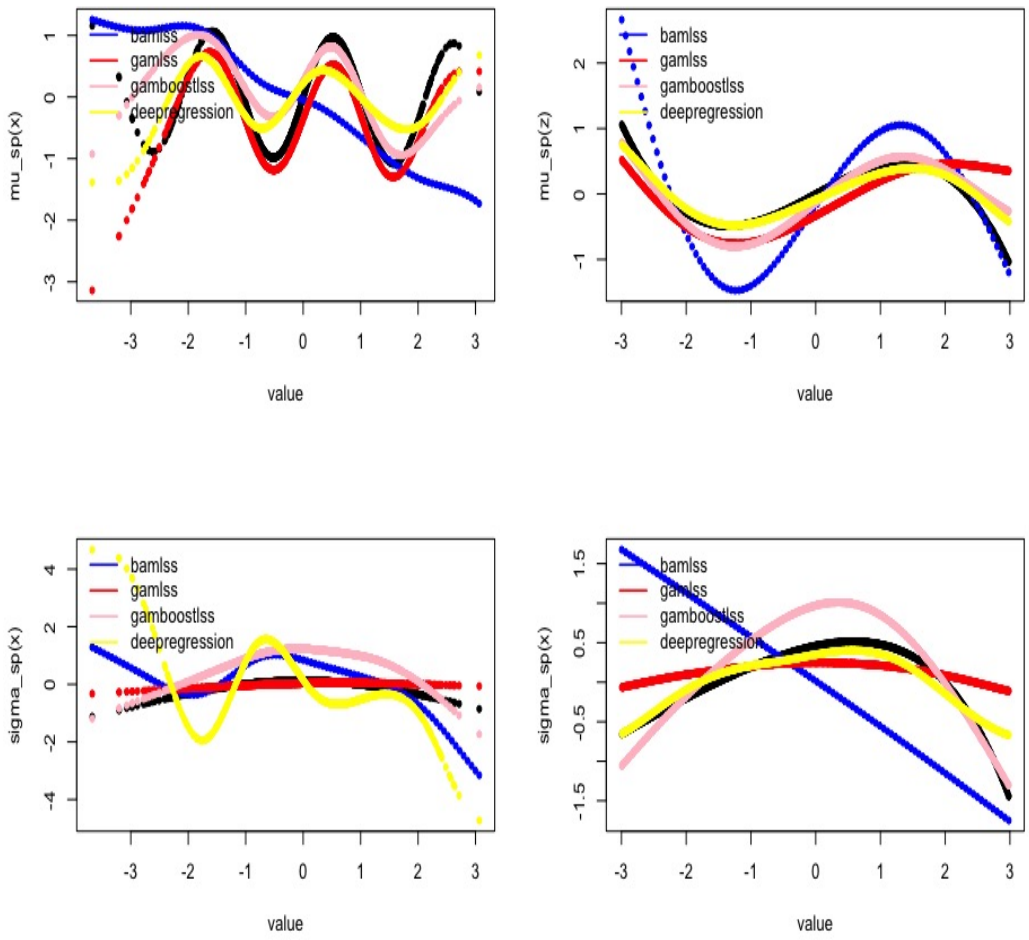


Figure 7.12: Averaged non-linear estimation by each model (ZIP distribution, two-dimensional case)

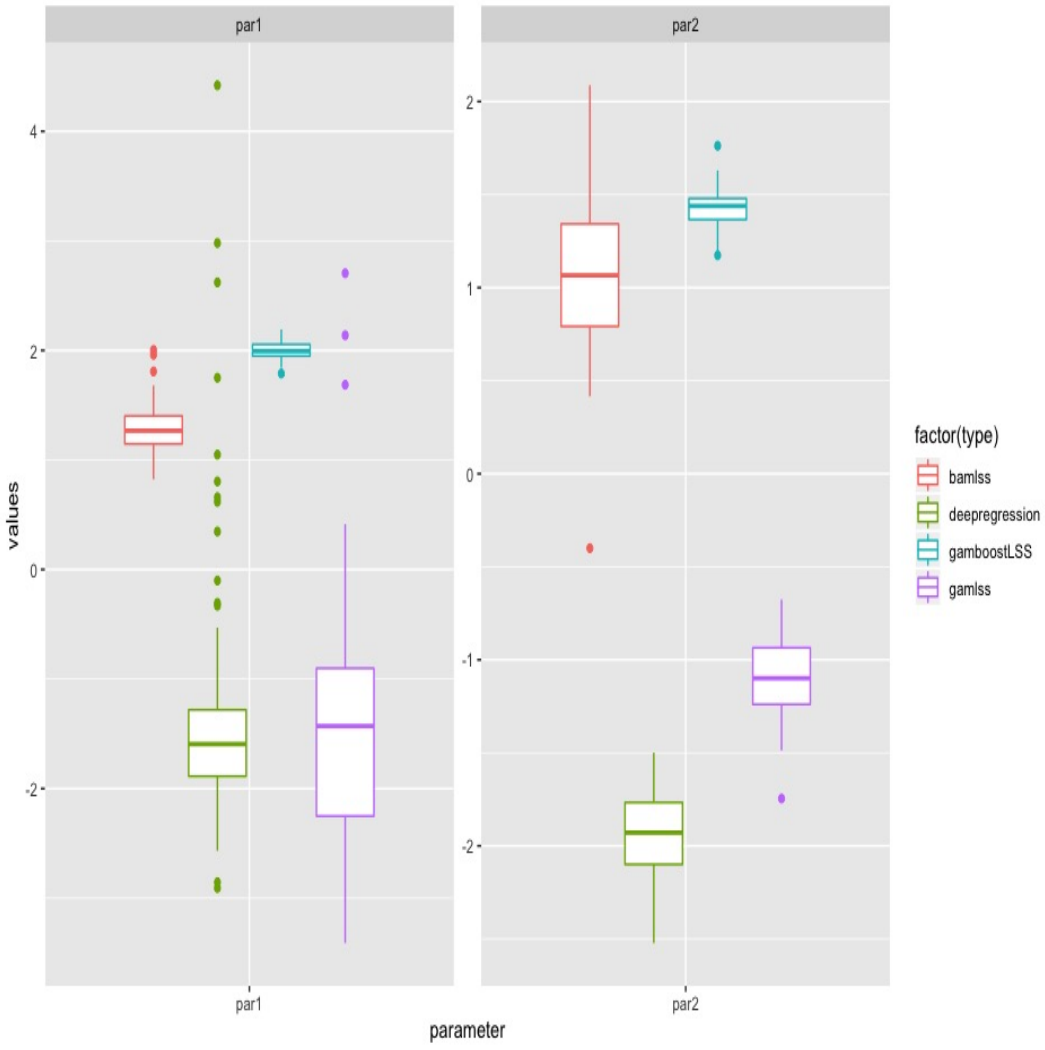


Figure 7.13: MSE of estimated parameters on test dataset (ZIP distribution, three-dimensional case, from left to right: BAMLSS, deep-regression, gamboostLSS, GAMLSS)

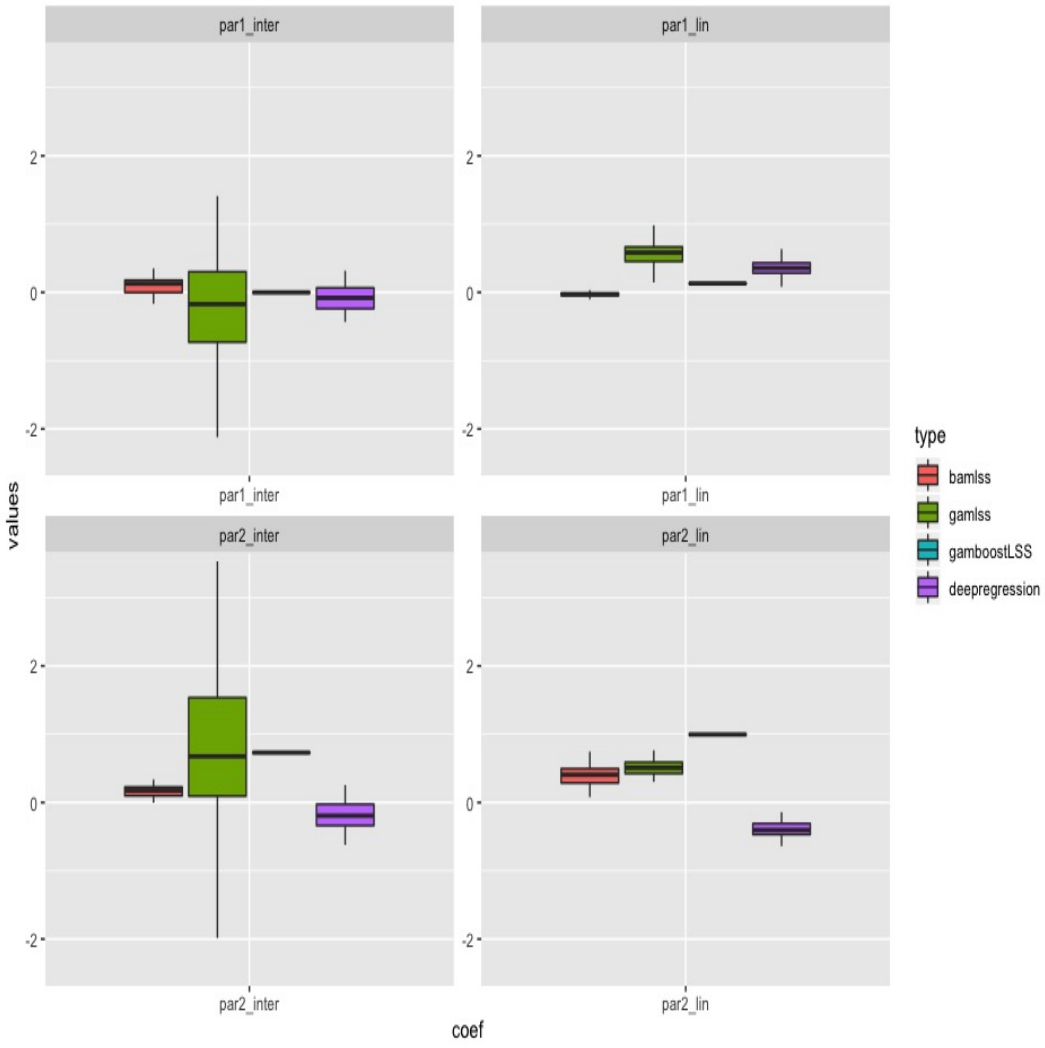


Figure 7.14: Linear Estimation (ZIP distribution, three-dimensional case) by each model in 100 iteration (figure on the top-left: intercept in parameter 1; figure on the top-right: linear coefficient in parameter 1; figure on the bottom-left: intercept in parameter 2; figure on the bottom-right: linear coefficient in parameter 2; from left to right in each figure: BAMLSS, GAMLSS, gamboostLSS, deep-regression)

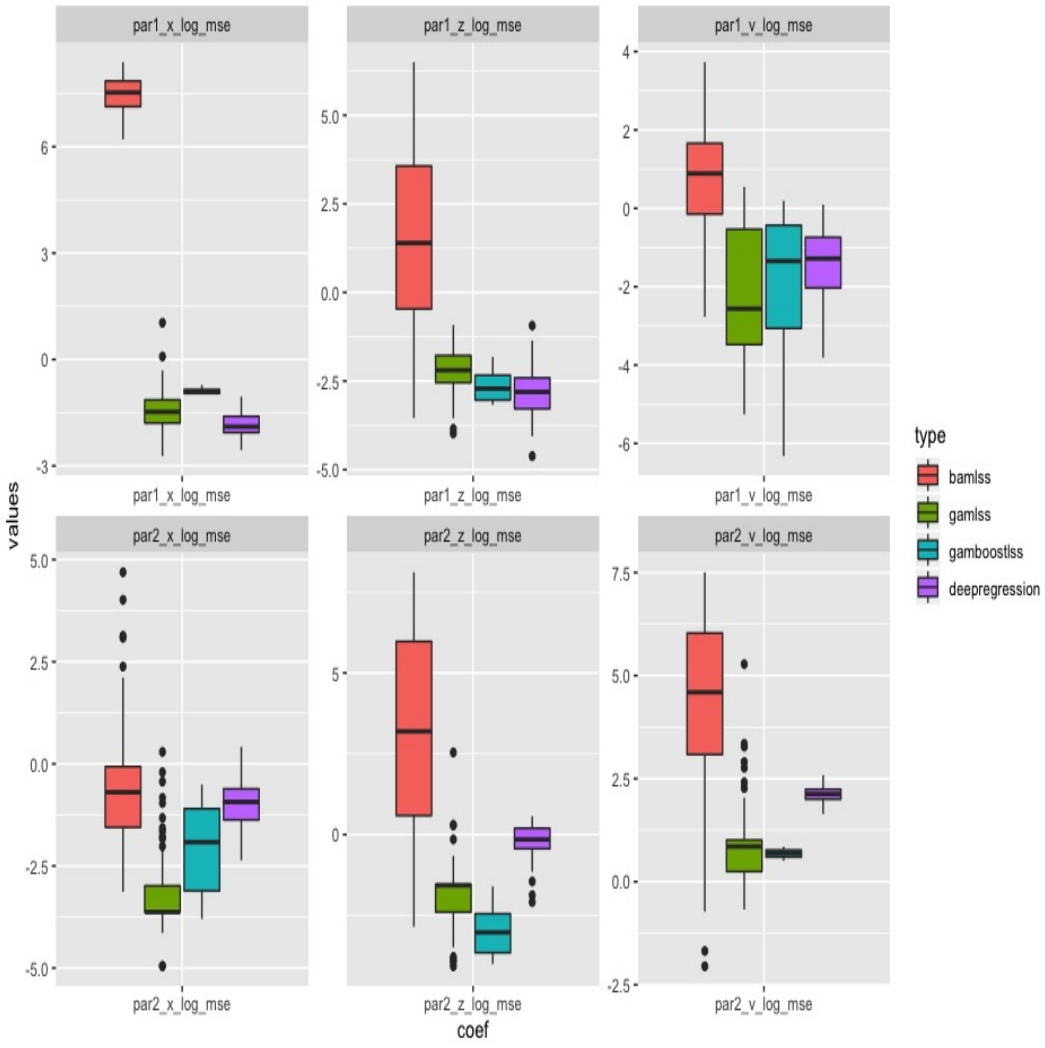


Figure 7.15: Adjusted MSE <$log(\frac{1}{n}\sum_{i=1}^{n}[\hat{f}_n(x_i) - f(x_i)]^2)$> of non-linear estimation by each model (ZIP distribution, three-dimensional case)

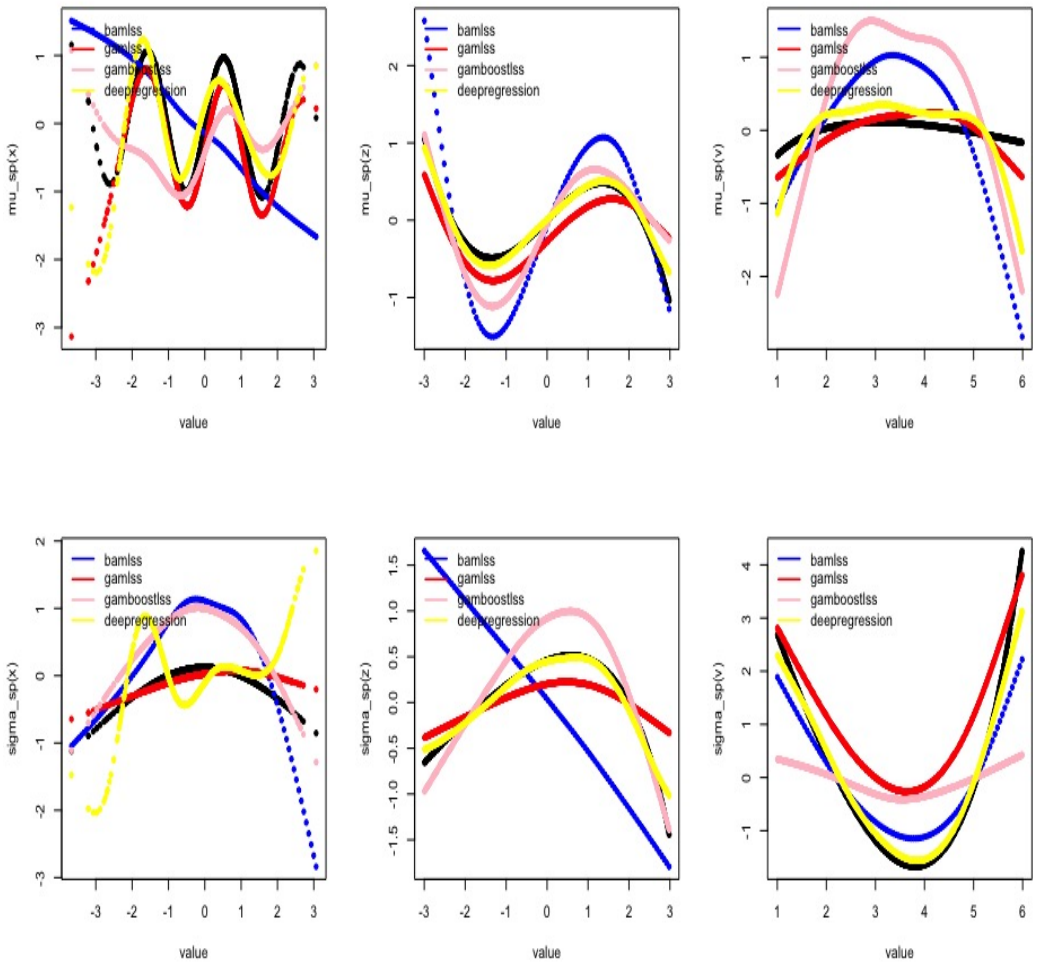


Figure 7.16: Averaged non-linear estimation by each model (ZIP distribution, three-dimensional case)

References

[1] David Rügamer, Chris Kolb, and Nadja Klein. A unifying network architecture for semi-structured deep distributional learning. *arXiv preprint arXiv:2002.05777*, 2020.

[2] D Mikis Stasinopoulos, Robert A Rigby, et al. Generalized additive models for location scale and shape (gamlss) in r. *Journal of Statistical Software*, 23(7):1–46, 2007.

[3] Nikolaus Umlauf, Nadja Klein, and Achim Zeileis. Bamlss: Bayesian additive models for location, scale, and shape (and beyond). *Journal of Computational and Graphical Statistics*, 27(3):612–627, 2018.

[4] Andreas Mayr, Nora Fenske, Benjamin Hofner, Thomas Kneib, and Matthias Schmid. Generalized additive models for location, scale and shape for high dimensional data—a flexible approach based on boosting. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 61(3):403–427, 2012.

[5] M-N Tran, Nghia Nguyen, David Nott, and Robert Kohn. Bayesian deep net glm and glmm. *Journal of Computational and Graphical Statistics*, pages 1–17, 2019.

[6] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[9] Ttcvor Hastie and Robert Tibahirani. Generalized additive models. 1984.

[10] John Ashworth Nelder and Robert WM Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972.

[11] Robert A Rigby and D Mikis Stasinopoulos. Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 54(3):507–554, 2005.

[12] Alain F Zuur, Elena N Ieno, Neil J Walker, Anatoly A Saveliev, and Graham M Smith. Zero-truncated and zero-inflated models for count data. In *Mixed effects models and extensions in ecology with R*, pages 261–293. Springer, 2009.

[13] Peter Bühlmann and Bin Yu. Boosting with the l 2 loss: regression and classification. *Journal of the American Statistical Association*, 98(462):324–339, 2003.

[14] Matthias Schmid and Torsten Hothorn. Boosting additive models using component-wise p-splines. *Computational Statistics & Data Analysis*, 53(2):298–311, 2008.

[15] Andreas Brezger, Thomas Kneib, and Stefan Lang. Bayesx: Analysing bayesian structured additive regression models. Technical report, Discussion paper//Sonderforschungsbereich 386 der Ludwig-Maximilians . . . , 2003.

[16] Nikolaus Umlauf, Daniel Adler, Thomas Kneib, Stefan Lang, and Achim Zeileis. Structured additive regression models: An r interface to bayesx. Technical report, Working Papers in Economics and Statistics, 2012.

[17] Nadja Klein, Thomas Kneib, and Stefan Lang. Bayesian generalized additive models for location, scale, and shape for zero-inflated and overdispersed count data. *Journal of the American Statistical Association*, 110(509):405–419, 2015.

[18] Andreas Brezger and Stefan Lang. Generalized structured additive regression based on bayesian p-splines. *Computational Statistics & Data Analysis*, 50(4):967–991, 2006.

[19] Dave Anderson and George McNeill. Artificial neural networks technology. *Kaman Sciences Corporation*, 258(6):1–83, 1992.

[20] Davide Castelvecchi. Can we open the black box of ai? *Nature News*, 538(7623):20, 2016.

[21] Ludwig Fahrmeir, Thomas Kneib, and Stefan Lang. Penalized structured additive regression for space-time data: a bayesian perspective. *Statistica Sinica*, pages 731–761, 2004.

[22] Robert A Rigby and Mikis D Stasinopoulos. Mean and dispersion additive models. In *Statistical theory and computational aspects of smoothing*, pages 215–230. Springer, 1996.

[23] RA Rigby and DM Stasinopoulos. A semi-parametric additive model for variance heterogeneity. *Statistics and Computing*, 6(1):57–65, 1996.

[24] Timothy J Cole and Pamela J Green. Smoothing reference centile curves: the lms method and penalized likelihood. *Statistics in medicine*, 11(10):1305–1319, 1992.

[25] John O Rawlings, Sastry G Pantula, and David A Dickey. *Applied regression analysis: a research tool*. Springer Science & Business Media, 2001.

[26] Sonja Greven and Thomas Kneib. On the behaviour of marginal and conditional aic in linear mixed models. *Biometrika*, 97(4):773–789, 2010.

[27] Nadja Klein, Thomas Kneib, Michel Denuit, and Stefan Lang. Bayesian generalized additive models for location, scale and shape applied to insurance data.

[28] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[29] Simon N Wood. mgcv: Gams and generalized ridge regression for r. *R news*, 1(2):20–25, 2001.

[30] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[31] Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017.

[32] Krishnamurthy Dvijotham, Marta Garnelo, Alhussein Fawzi, and Pushmeet Kohli. Verification of deep probabilistic models. *arXiv preprint arXiv:1812.02795*, 2018.

[33] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[34] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[36] Paul HC Eilers and Brian D Marx. Flexible smoothing with b-splines and penalties. *Statistical science*, pages 89–102, 1996.

[37] Finbarr O'Sullivan. A statistical perspective on ill-posed inverse problems. *Statistical science*, pages 502–518, 1986.

[38] Finbarr O'Sullivan. Fast computation of fully automated log-density and log-hazard estimators. *SIAM Journal on scientific and statistical computing*, 9(2):363–379, 1988.

[39] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[40] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[41] Geoffrey E Hinton, Alexander Krizhevsky, Ilya Sutskever, and Nitish Srivastva. System and method for addressing overfitting in a neural network, August 2 2016. US Patent 9,406,017.

[42] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[43] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[44] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[45] Douglas Staiger and James H Stock. Instrumental variables regression with weak instruments. Technical report, National Bureau of Economic Research, 1994.

[46] Henri Theil. Repeated least squares applied to complete equation systems. *The Hague: central planning bureau*, 1953.

[47] Robert L Basmann. A generalized classical method of linear estimation of coefficients in a structural equation. *Econometrica: Journal of the Econometric Society*, pages 77–83, 1957.

Acknowledgements

I would like to express my special thanks of gratitude to my second supervisor, Dr.David Rügamer for patiently instructing me to DGAMLSS model, for suggestions on programming in R and the structure of my thesis. Secondly I would also like to address my appreciation to Prof.Dr.Sonja Greven and her Phd students for orientation to academic writing in their seminars, and materials in generalized regression model in course "Generalized Regression" as the vital groundwork for my research in this project. Lastly, thanks to my supervisor, Prof.Dr.Nadja Klein, who has been the source of motivation and verbally spurred me on to push forward.

Declaration of Academic Honesty

I, Kang Yang, hereby declare that I have not previously submitted the present work for other examinations. I wrote this work independently. All sources, including sources from the Internet, that I have reproduced in either an unaltered or modified form (particularly sources for texts, graphs, tables and images), have been acknowledged by me as such. I understand that violations of these principles will result in proceedings regarding deception or attempted deception.

Kang Yang

Berlin, March 02, 2020