universität
innsbruck

Neural Network Distributional Regression

Nikolaus Umlauf

http://eeecon.uibk.ac.at/~umlauf/

Overview

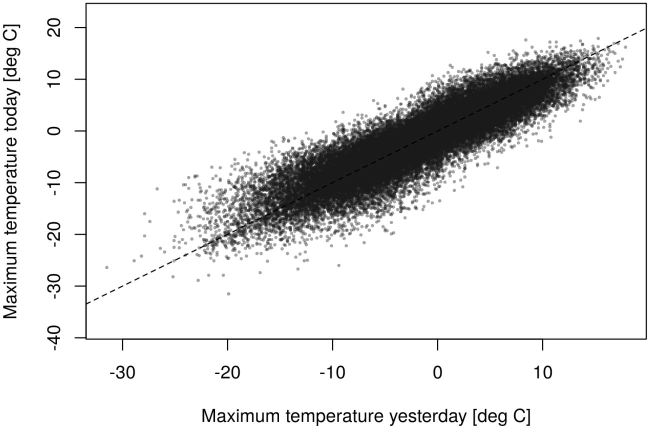
Joint work with Nadja Klein, Thorsten Simon and Achim Zeileis.

1. Introduction
2. Model Specification
3. Neural Network Distributional Regression
4. Application

Introduction

Zugspitze daily max. T (1900/8-2015/12).

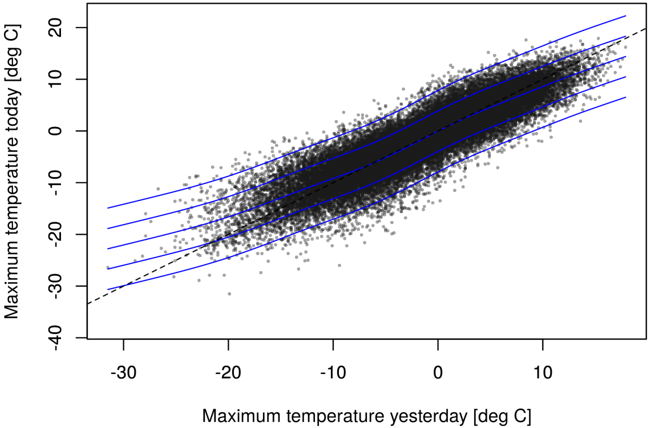
$$\mathtt{T} \sim N(\mu, \sigma^2).$$



Introduction

Zugspitze daily max. T (1900/8-2015/12).

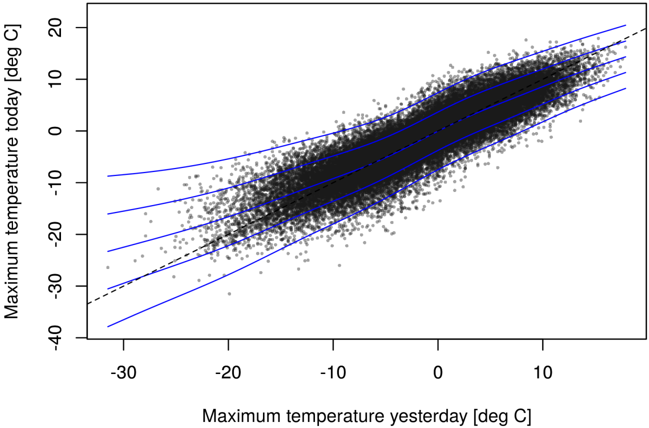
$$\mathtt{T} \sim N(\mu = f(\mathtt{T}_{t-1}), \, log(\sigma^2) = \beta_0).$$



Introduction

Zugspitze daily max. T (1900/8-2015/12).

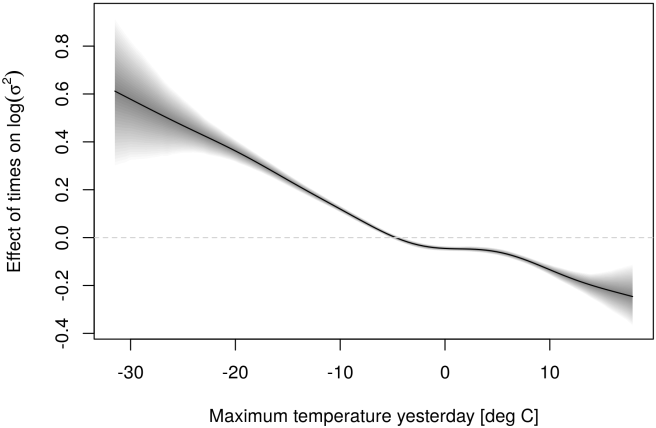
$$\mathtt{T} \sim N(\mu = f(\mathtt{T}_{t-1}),\ log(\sigma^2) = f(\mathtt{T}_{t-1})).$$



Introduction

Zugspitze daily max. T (1900/8-2015/12).

$$\texttt{T} \sim N(\mu = f(\texttt{T}_{t-1}),\ log(\sigma^2) = f(\texttt{T}_{t-1})).$$



Model specification

Any parameter of a population distribution $\mathcal{D}$ may be modeled by explanatory variables

$$y \sim \mathcal{D}\left(h_1(\theta_1) = \eta_1,\ h_2(\theta_2) = \eta_2, \ldots,\ h_K(\theta_K) = \eta_K\right),$$

Each parameter is linked to a structured additive predictor

$$h_k(\theta_k) = \eta_k = \eta_k(\mathbf{x}; \boldsymbol{\beta}_k) = f_{1k}(\mathbf{x}; \boldsymbol{\beta}_{1k}) + \ldots + f_{J_k k}(\mathbf{x}; \boldsymbol{\beta}_{J_k k}),$$

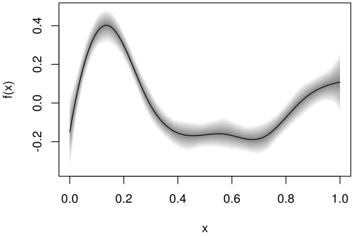
$j = 1, \ldots, J_k$ and $k = 1, \ldots, K$ and $h_k(\cdot)$ are link functions.

Vector of function evaluations $\mathbf{f}_{jk} = (f_{jk}(\mathbf{x}_1; \boldsymbol{\beta}_{jk}), \ldots, f_{jk}(\mathbf{x}_n; \boldsymbol{\beta}_{jk}))^\top$

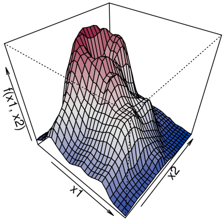
$$\mathbf{f}_{jk} = \begin{pmatrix} f_{jk}(\mathbf{x}_1; \boldsymbol{\beta}_{jk}) \\ \vdots \\ f_{jk}(\mathbf{x}_n; \boldsymbol{\beta}_{jk}) \end{pmatrix} = f_{jk}(\mathbf{X}_{jk}; \boldsymbol{\beta}_{jk}).$$

Model specification

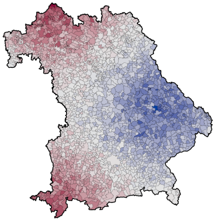
Nonlinear effects of continuous covariates



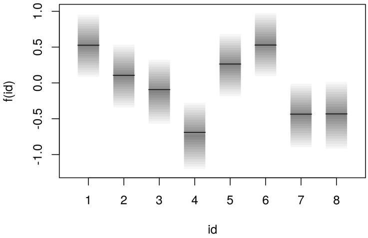
Two-dimensional surfaces



Spatially correlated effects f(x) = f(s)

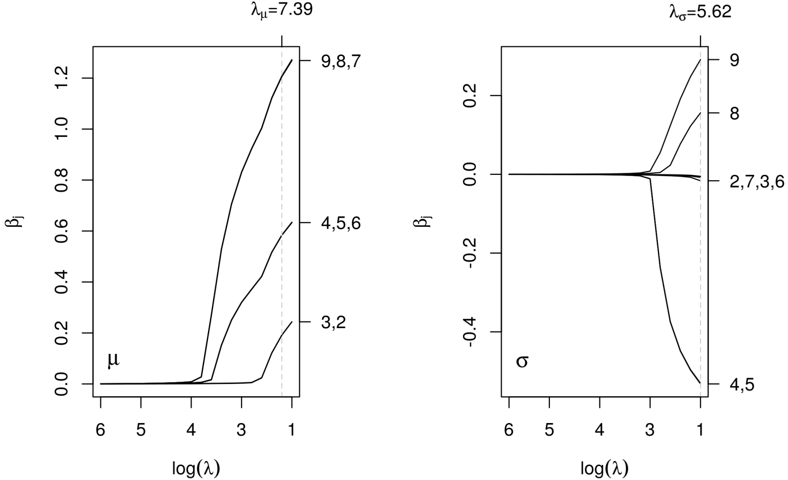


Random intercepts f(x) = f(id)



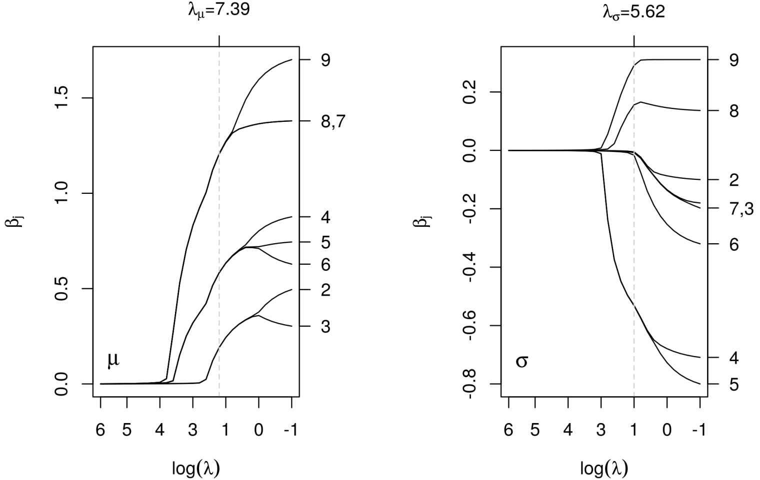
Model specification

Model terms $f_{jk}(\mathbf{x}; \beta_{jk})$ with LASSO-type penalties $J_c(\beta_{jk})$.



Model specification

Model terms $f_{jk}(\mathbf{x}; \beta_{jk})$ with LASSO-type penalties $J_f(\beta_{jk})$.



Neural Network Distributional Regression

How to capture complex nonlinearities? Additive predictors $\eta_k(\mathbf{x}; \boldsymbol{\beta}_k)$ using regression splines have great performance, but can we do better?

- Feedforward neural networks (FNN) are extensively used in regression and classification applications.
- FNNs are universal function approximators (Hornik 1991).
- However, estimation is usually difficult and can involve thousands of parameters.
- Which makes the problem even harder in a full distributional regression setting (full Bayesian inference?).

$\Rightarrow$ Use FNN model term $f_{jk}(\mathbf{X}_{jk}; \boldsymbol{\beta}_{jk})$ additional to all other effects.

Neural Network Distributional Regression

Setup:

A FNN model term has a simple structure

$$f_{jk}(\mathbf{X}_{jk}; \boldsymbol{\beta}_{jk}) = \mathbf{X}_{jk}\boldsymbol{\beta}_{jk},$$

where the columns of $\mathbf{X}_{jk}$ are a decomposition of activation functions, e.g., using the sigmoid the l-th column (node) is

$$h_l(\mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{w}_l^\top \mathbf{x} + b_l))},$$

where $\mathbf{w}_l$ and b_l are inner weights and biases.

The activation function $h_l(\cdot)$ could also be Gauss (radial basis function network), sin, etc.

Neural Network Distributional Regression

Basic idea:

Reduce computational complexity, avoid non-convex optimization (time consuming, sensitive to initial values, local minima), by randomly selecting $\mathbf{w}_l$ and b_l, i.e., compute a random design matrix $\mathbf{X}_{jk}$.

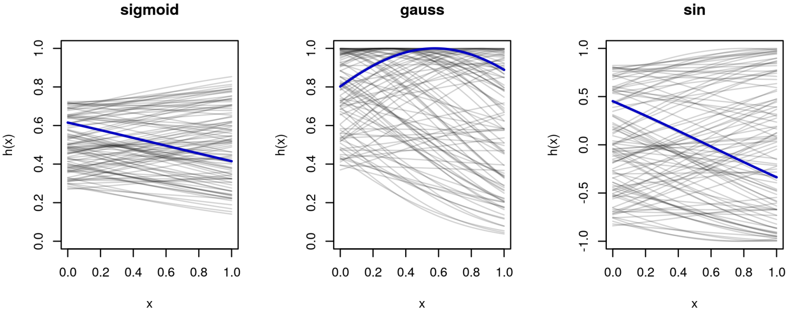
Although the idea is not new, this is now also known by the controversial name *extreme learning machine* (ELM, Huang 2006).

There are theoretical results that ELMs are also universal function approximators using symmetric intervals for the parameter scope (Husmeier 1999), a.o.

Neural Network Distributional Regression

Problems: How to randomly select $\mathbf{w}_l$ and b_l?

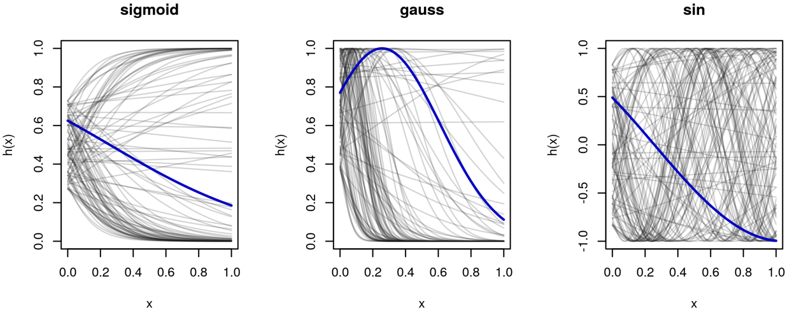
Sample $w_{ld}, b_l \sim \mathcal{U}(-1, 1)$. (Schmidt et al. 1992)



Neural Network Distributional Regression

Problems: How to randomly select $\mathbf{w}_l$ and b_l?

Sample $w_{ld} \sim \mathcal{U}(-10, 10)$ and $b_l \sim \mathcal{U}(-1, 1)$



Neural Network Distributional Regression

- Too small values for $\mathbf{w}_l$ and b_l lead to poor distribution of the basis functions (activation functions).
- Too large values will lead to saturated functions.
- Some literature about tuning the sampling range.
- Need a method that controls the flatness and steepness in the input hypercube.

$\Rightarrow$ Dudek (2017) gives a detailed description of how to select weights and biases for different activation functions.

Neural Network Distributional Regression

Sampling weights: Dudek (2017)

For $[0, 1]$ scaled inputs, weights are sampled such that the most nonlinear and steepest parts are inside the data region.

1. Given r and s, sample sum of input weights

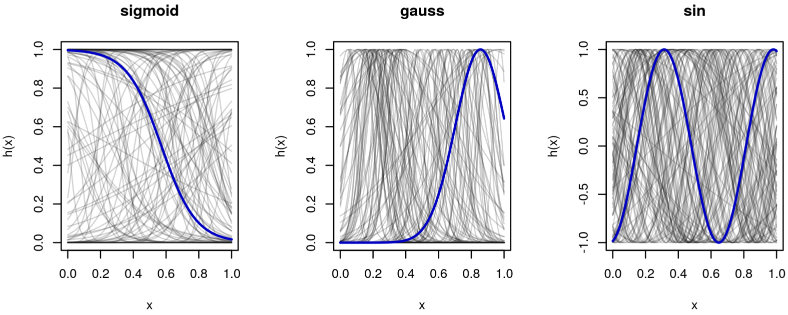
$$\sum\nolimits_{[l]} \sim \mathcal{U}\left(\log\left[\frac{1-r}{r}\right], s \cdot \log\left[\frac{1-r}{r}\right]\right).$$

2. For $\mathbf{w}_l$ sample $\zeta_d \sim \mathcal{U}(-1, 1)$.
3. Set $w_{ld} = \zeta_d \frac{\sum_{[l]}}{\sum_d \zeta_d}$.
4. Set $b_l = -\sum_d w_{ld} z_l$, where $z_l \sim \mathcal{U}(0, 1)$.

Depending on the activation functions, r and s can have different ranges.

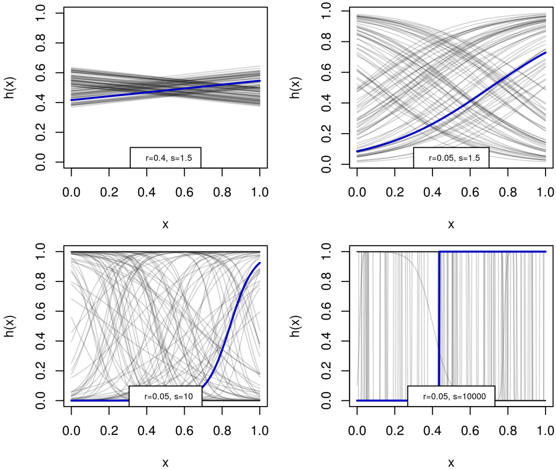
Neural Network Distributional Regression

Sampling weights: Dudek (2017)



Neural Network Distributional Regression

Sampling weights: Scaling with r and s.



Neural Network Distributional Regression

Overfitting:

We use elastic net regularization

$$\lambda_{jk1} \cdot J_{\mathrm{L}}(\boldsymbol{\beta}_{jk}) + \lambda_{jk2} \cdot J_{\mathrm{R}}(\boldsymbol{\beta}_{jk}),$$

with quadratic approximations of the LASSO penalties (compare Oelker & Tutz, 2017; Groll et al., 2018)

$$J_{\mathrm{L}}(\boldsymbol{\beta}_{jk}) \approx J_{\mathrm{L}}(\boldsymbol{\beta}_{jk}^{(t)}) + \frac{1}{2}\left(\boldsymbol{\beta}_{jk}^{\top}\mathbf{P}_{jk}(\boldsymbol{\beta}_{jk})\boldsymbol{\beta}_{jk} + (\boldsymbol{\beta}_{jk}^{(t)})^{\top}\mathbf{P}_{jk}(\boldsymbol{\beta}_{jk}^{(t)})\boldsymbol{\beta}_{jk}^{(t)}\right),$$

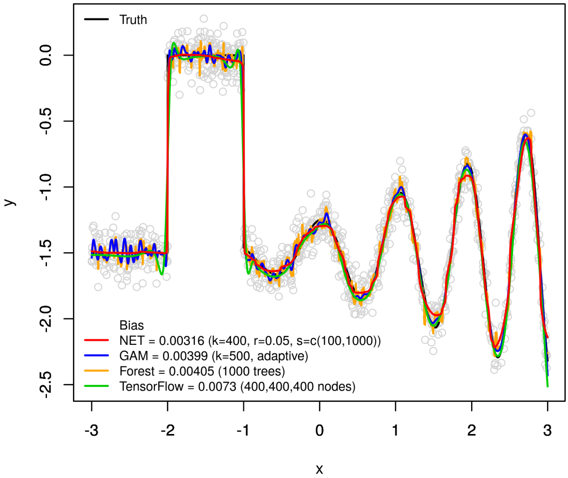
with

$$\mathbf{P}_{jk}(\boldsymbol{\beta}_{jk}^{(t)}) = q'_{jk}\left(\left\|\mathbf{a}_{jk}^{\top}\boldsymbol{\beta}_{jk}^{(t)}\right\|_{N_{jk}}\right) \cdot \frac{D_{jk}(\mathbf{a}_{jk}^{\top}\boldsymbol{\beta}_{jk}^{(t)})}{\mathbf{a}_{jk}^{\top}\boldsymbol{\beta}_{jk}^{(t)}} \cdot \mathbf{a}_{jk}\mathbf{a}_{jk}^{\top}.$$

E.g., $\|\beta\|_1 = |\beta|$ is approximated by $\sqrt{\beta^2 + c}$, hence, IWLS based updating functions are relatively easy to implement.

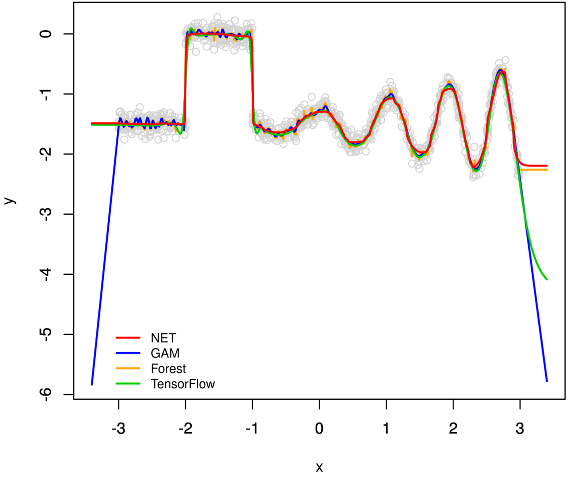
Neural Network Distributional Regression

Simulated example: Sigmoid activation.



Neural Network Distributional Regression

Simulated example: Out of range predictions.



Leukemia Survival Example

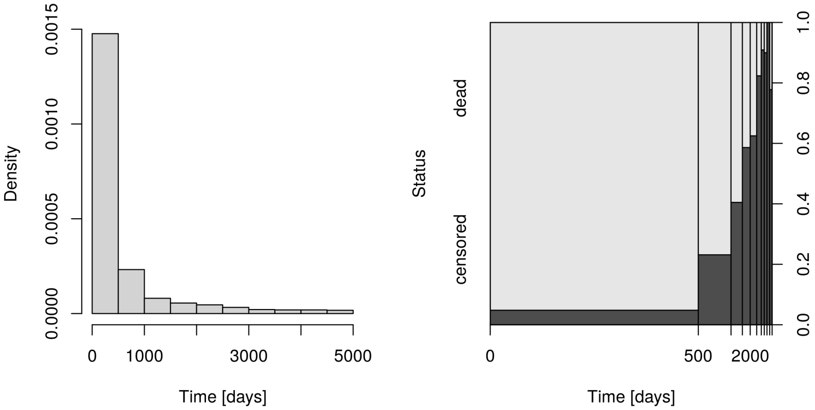
Data structure:

First analyzed by Henderson et al. (2002), investigate spatial variation in survival after accounting for subject-specific factors in northwest England. ($n = 1043$ patients)

Variable	Description.
`time`	Survival time in days.
`cens`	Right censoring status 0=censored, 1=dead.
`xcoord`	Coordinates in x-axis of residence.
`ycoord`	Coordinates in y-axis of residence.
`age`	Age in years.
`sex`	male=1 female=0.
`wbc`	White blood cell count at diagnosis, truncated at 500.
`tpi`	The Townsend score for which higher values indicates less affluent areas.
`district`	Administrative district of residence.

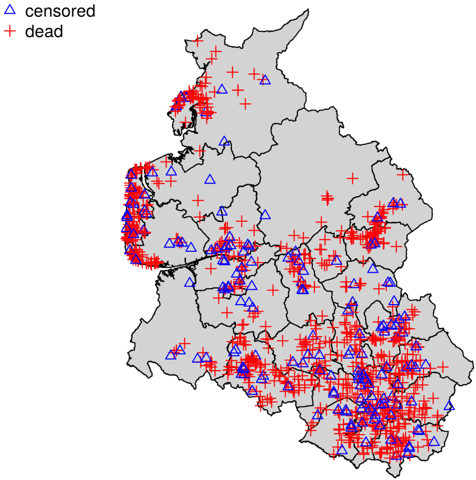
Leukemia Survival Example

Survival times:



Leukemia Survival Example

Spatial distribution:



Leukemia Survival Example

Cox model:

The hazard of an event (status `dead`) at time t can be described with a relative additive risk model of the form:

$$\lambda(t) = \exp\left(\eta(t)\right) = \exp\left(\eta_\lambda(t) + \eta_\gamma\right),$$

i.e., a model for the instantaneous risk conditional on being alive before time t.

The probability to not survive after time t is

$$S(t) = Prob(T > t) = \exp\left(-\int_0^t \lambda(u)du\right).$$

Leukemia Survival Example

For the leukemia survival example, we use the following additive predictors

$$\eta_\lambda(\texttt{time}) = f_1(\texttt{time}) + f_2(\texttt{time}, \texttt{sex}, \texttt{age}, \texttt{wbc}, \texttt{tpi}, \texttt{xcoord}, \texttt{ycoord})$$

and

$$\begin{aligned} \eta_\gamma &= \beta_0 + \texttt{sex} + f_3(\texttt{age}) + f_4(\texttt{wbc}) + f_5(\texttt{tpi}) + \\ &\quad f_6(\texttt{xcoord}, \texttt{ycoord}) + \\ &\quad f_7(\texttt{sex}, \texttt{age}, \texttt{wbc}, \texttt{tpi}, \texttt{xcoord}, \texttt{ycoord}). \end{aligned}$$

Here, functions $f_2(\cdot)$ and $f_7(\cdot)$ represent a time dependent and a time constant neural network model term.

For the other functions we use regression splines.

Leukemia Survival Example

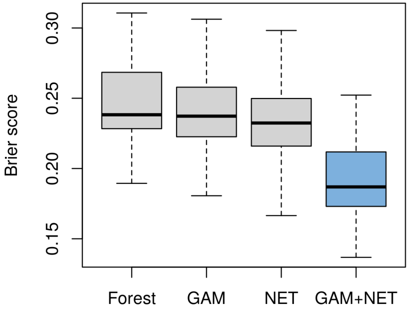
In R we set up the model by

```
R> library("bamlss")
R> library("survival")
R> data("LeukSurv", package = "spBayesSurv")
R> ftd <- ~ time + sex + age + wbc + tpi + xcoord + ycoord
R> ftc <- ~ sex + age + wbc + tpi + xcoord + ycoord
R> f <- list(
+   Surv(time, cens) ~ s(time) +
+     n(ftd,k=300,pt="lasso",
+       rint=list("sigmoid"=0.1,"gauss"=0.1),
+       sint=list("sigmoid"=c(5,10),"gauss"=5),
+       afun=c("sigmoid","gauss"),ndf=50),
+   gamma ~ sex + s(age) + s(wbc) + s(tpi) + s(xcoord,ycoord,k=100) +
+     n(ftc,k=300,pt="lasso",
+       rint=list("sigmoid"=0.1,"gauss"=0.1),
+       sint=list("sigmoid"=c(5,10),"gauss"=5),
+       afun=c("sigmoid","sin","gauss"),ndf=50)
+ )
R> b <- bamlss(f, data = LeukSurv, family = "cox")
```

Leukemia Survival Example

Performance:

We evaluate the performance of the neural network Cox model by randomly sampling 100 individuals that serve as a hold out sample and compare using the Brier score. This is done 50 times.



In sample Brier score: GAM=0.24, GAM+NET=0.18.

Leukemia Survival Example

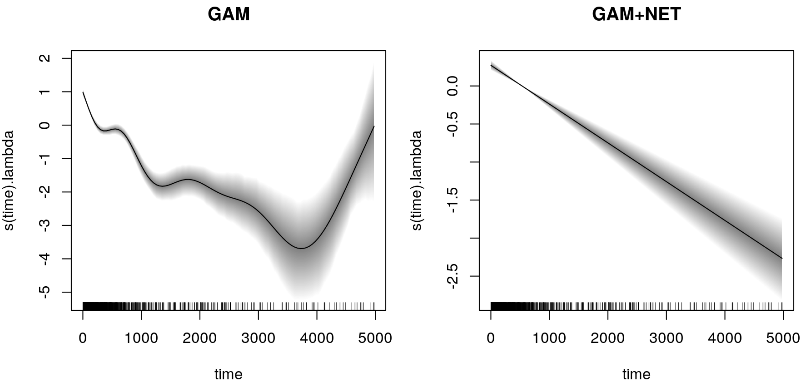
```
R> summary(b)

## Subset of full model summary.

Formula lambda:
---
Surv(time, cens) ~ s(time) + n(ftd, k = 300, pt = "lasso",
    rint = list(sigmoid = 0.1, gauss = 0.1),
    sint = list(sigmoid = c(5, 10), gauss = 5),
    afun = c("sigmoid", "gauss"), ndf = 50)

-
Smooth terms:
              parameters
s(time).tau21      0.000
s(time).edf        0.984
n(ftd).tau21      76.543
n(ftd).edf        34.061
---
```

Leukemia Survival Example

```
Formula gamma:
---
gamma ~ sex + s(age) + s(wbc) + s(tpi) + s(xcoord, ycoord, k = 100) +
    n(ftc, k = 300, pt = "lasso", rint = list(sigmoid = 0.1,
        gauss = 0.1), sint = list(sigmoid = c(5, 10), gauss = 5),
        afun = c("sigmoid", "sin", "gauss"), ndf = 50)
-
Smooth terms:
                       parameters
s(age).tau21                0.000
s(age).edf                  0.997
s(wbc).tau21                0.000
s(wbc).edf                  0.977
s(tpi).tau21               86.135
s(tpi).edf                  7.954
s(xcoord,ycoord).tau21      0.147
s(xcoord,ycoord).edf        7.935
n(ftc).tau21                0.000
n(ftc).edf                  0.000
---
```

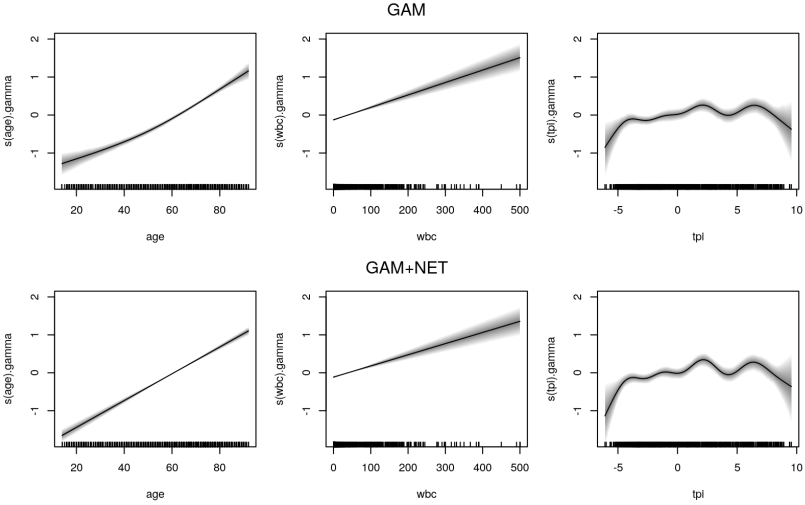
Leukemia Survival Example

```
R> plot(b, model = "lambda", term = "s(time)")
```



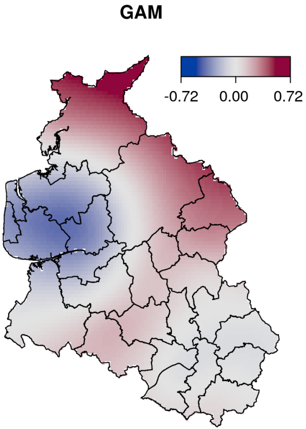
Leukemia Survival Example

```
R> plot(b, model = "gamma", term = c("s(age)", "s(wbc)", "s(tpi)"))
```



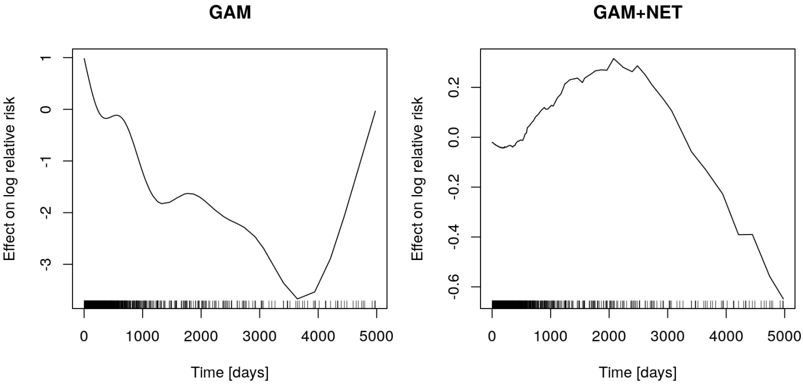
Leukemia Survival Example

```
R> predict(b, newdata = nd,
+    model = "gamma", term = "s(xcoord,ycoord)")
```



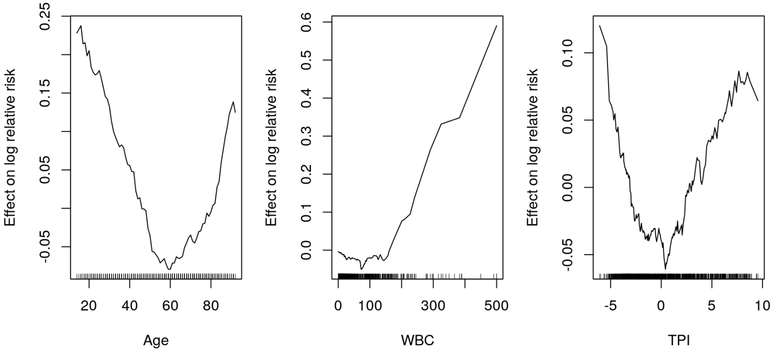
Leukemia Survival Example

Accumulated local effects (ALE) plots: (Apley D.W., 2016)



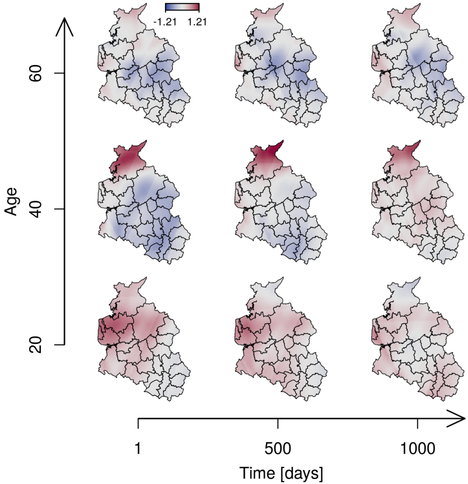
Leukemia Survival Example

Accumulated local effects (ALE) plots: (Apley D.W., 2016)



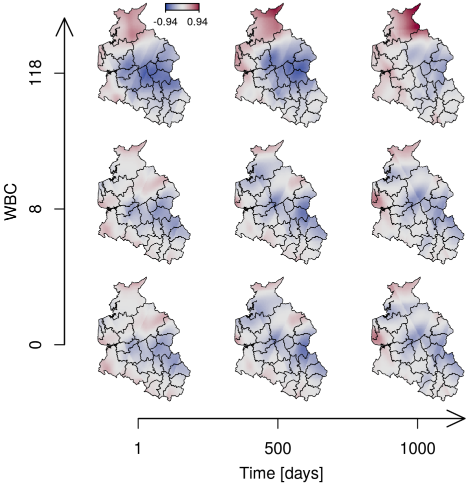
Leukemia Survival Example

Interaction plots: (females, remaining variables fixed at means)



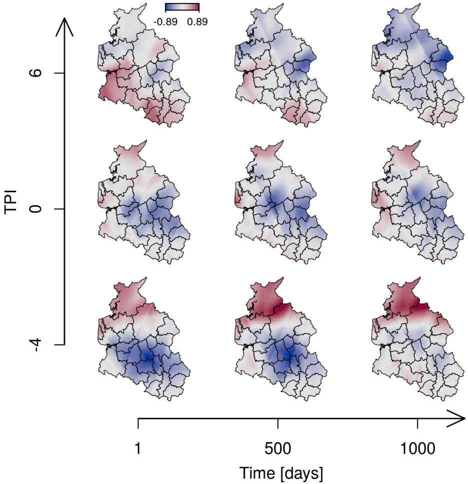
Leukemia Survival Example

Interaction plots: (females, remaining variables fixed at means)



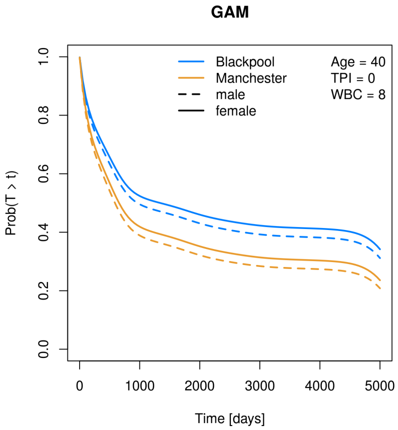
Leukemia Survival Example

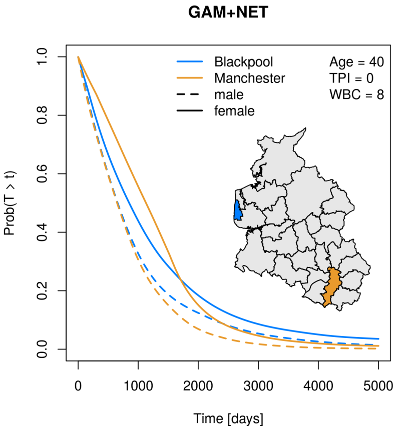
Interaction plots: (females, remaining variables fixed at means)



Leukemia Survival Example

Probabilities: Blackpool vs. Manchester.





Summary & Outlook

- Neural networks really seem to have good approximation skills.
- Capable to find high-order interactions.
- However, this needs to be further investigated.
- Good predictive performance, but interpretation is still difficult.

- Linears vs. nonlinear direct connectors?
- Tune weights instead of random sampling?
- Full Bayesian inference for weights?
- Deep networks?

References & Software

- Apley D.W. (2016). Visualizing the effects of predictor variables in black box supervised learning models. arXiv:1612.08468.
- Dudek G. (2017). A method of generating random weights and biases in feedforward neural networks with random hidden nodes. arXiv:1710.04874.
- Groll A., Hambuckers J., Kneib T. & and Umlauf N. (2018). Lasso-type penalization in the framework of generalized additive models for location, scale and shape. Working papers, Faculty of Economics and Statistics, University of Innsbruck. `https://econpapers.repec.org/paper/innwpaper/2018-16.htm`.
- Henderson R., Shimakura S. & and Gorst D. (2002). Modeling spatial variation in leukemia survival data. *Journal of the American Statistical Association* **70**(460).
- Hornik K. (1991). Approximation capabilities of multilayer feedforward networks *Neural Networks* **4(2)**, 251 – 257.
- Huang G.B., Zhu Q.Y. & Siew, C.K. (2006). Extreme Learning Machine: Theory and Applications. *Neurocomputing* **70**, 489 – 501.
- Husmeier D. (1999). Random vector functional link (RVFL) networks. *Neural Networks for Conditional Probability Estimation: Forecasting Beyond Point Predictions*, Chapter 6, Springer.
- Lang S., Umlauf N., Wechselberger P., Harttgen K. & and Kneib T. (2012). Multilevel structured additive regression. *Statistics and Computing*.
  URL: `http://link.springer.com/article/10.1007/s11222-012-9366-0`

References & Software

Oelker M.R. & Tutz G. (2017). A uniform framework for the combination of penalties in generalized structured models. *Advances in Data Analysis and Classification* **11**(1), 97 – 120.

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

Schmidt W.F., Kraaijveld M. & Duin R.P. (1992). Feedforward neural networks with random weights. *Proc. 11th IAPR IEEE Inter. Conf. on Pattern Recognition*, 1 – 4.

Stasinopoulos D.M., & Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software* **23**(7).

Tibshirani R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B 58* 267 – 288.

Umlauf N., Klein N. & Zeileis A. (2018). BAMLSS: Bayesian additive models for location, scale and shape (and beyond). Journal of Computational and Graphical Statistics, 2018. to appear. `doi:10.1080/10618600.2017.1407325`.

Umlauf N., Klein N., Zeileis A., Köhler M. & Thorsten S. (2019). **bamlss**: Bayesian additive models for location, scale and shape (and beyond). R package version 1.0-1, URL: `http://cran.r-project.org/package=bamlss`.