DoDSc Kolloquium / SFB 876 Topical Seminar

Semi-Structured Deep Distributional Regression

David Rügamer

Joint work with: Chris Kolb and Nadja Klein (HU Berlin)

29.10.2020

Semi-Structured Deep Distributional Regression (SDDR)

- Semi-Structured:
  - Combine structured data & statistical regression components
  - with unstructured components such as **deep** neural networks
- Distributional Regression:
  - Modeling the whole distribution
  - Similar to location, scale and shape (LSS) approaches

- Main idea:
  - Fit all commonly used statistical models in a neural network (1)
  - but also allow to learn parts of the model “deep” (2a)
  - Ensure a meaningful behaviour between the two parts (2b)

Outline

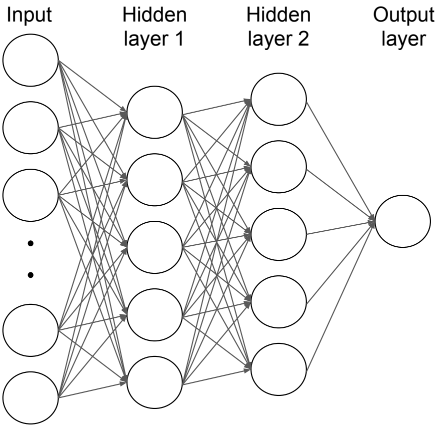
Statistical Regression in Neural Networks

- A neuron h in a neural network (NN) is a transformed linear combination
- For one observation ($n = 1$):

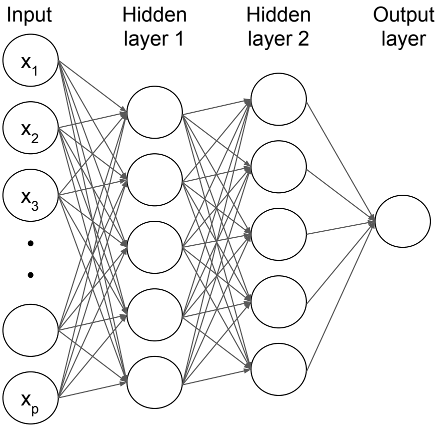
$$\begin{aligned}
\text{Input:} \quad & \boldsymbol{x} \in \mathbb{R}^{1\times p} \\
\text{Weights:} \quad & \boldsymbol{w} \in \mathbb{R}^{p} \\
\text{Input times weights:} \quad & \eta := \boldsymbol{x}\boldsymbol{w} = \textstyle\sum_j x_j w_j \in \mathbb{R} \\
\text{Output:} \quad & h = \sigma(\eta) \in \mathbb{R}
\end{aligned}$$

with activation function $\sigma(\cdot)$

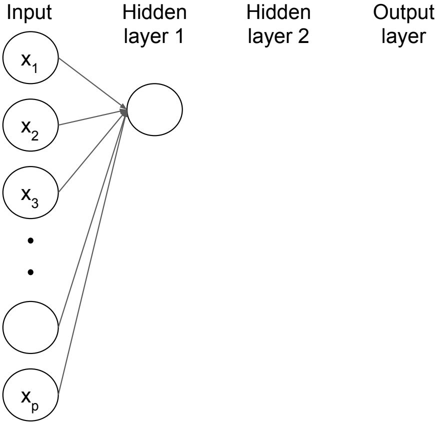
Some Neural Network



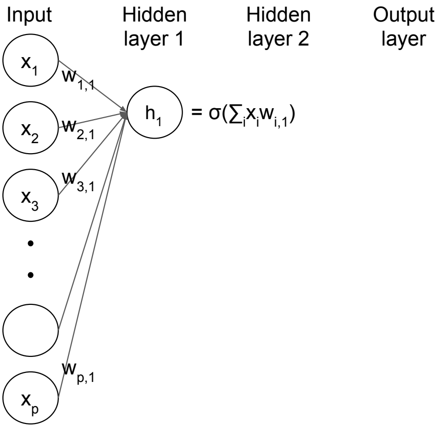
Some Neural Network

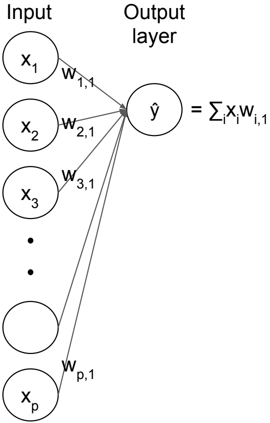


Some Neural Network

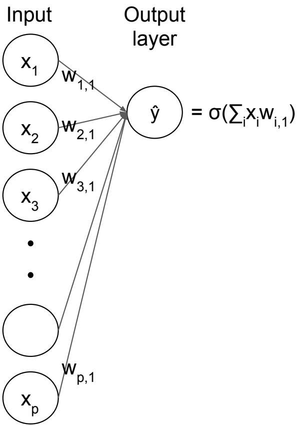


Some Neural Network



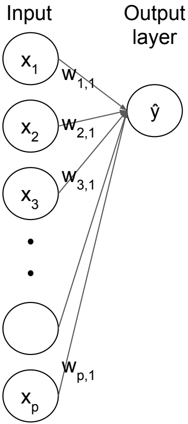


with loss $(\hat{y}-y)^2$



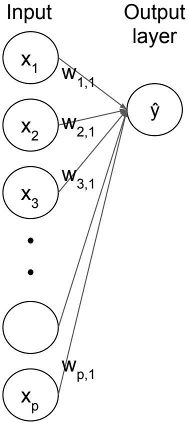
with loss $-\log \mathcal{L}(\hat{y}, y)$, $y \sim \text{ExpFam}(\boldsymbol{\theta})$

Ridge Regression



$= \sigma(\sum_i x_i w_{i,1})$

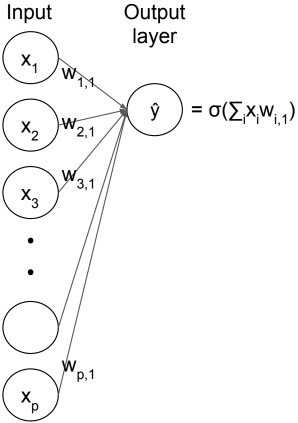
with loss $-\log \mathcal{L}(\hat{y}, y) + \lambda \sum_j w_j^2$



$= \sigma(\sum_i x_i w_{i,1})$

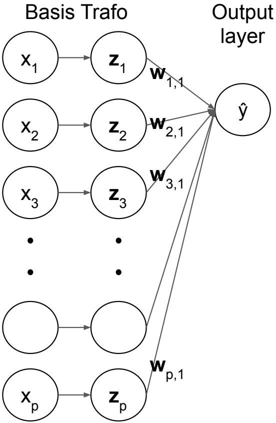
with loss $-\log \mathcal{L}(\hat{y},y) + \lambda \sum_j |w_j|$

More Complex Penalties



with loss $-\log \mathcal{L}(\hat{y},y) + \lambda\, \mathbf{w}^\top \mathbf{P}\, \mathbf{w}$

A Generalized Additive Model (GAM)



$= \sigma(\mathbf{zw})$

with loss $-\log \mathcal{L}(\hat{y},y) + \lambda\, \mathbf{w}^\top \mathbf{P}\, \mathbf{w}$

Advantages of GAMs in NN

- No limitation to $p \leq n$
- No limitation for large n
- Any differentiable loss function ($\leftrightarrow$ distribution)
- Auto differentiation

Advantages of GAMs in NN

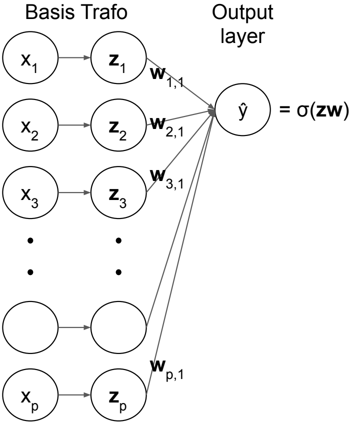
- No limitation to $p \leq n$
- No limitation for large n
- Any differentiable loss function ($\leftrightarrow$ distribution)
- Auto differentiation

But we can also learn whole distributions...

Distribution Layers

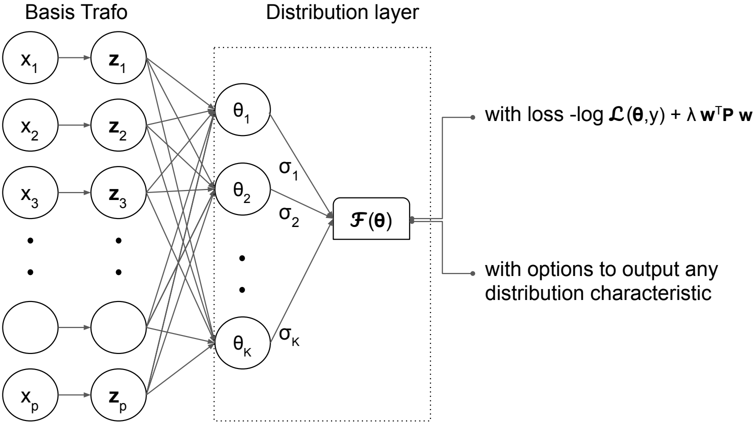
Idea [2]: Instead of using an output layer that just creates $\hat{y}$

- why not return a parametric distribution $\mathcal{F}(\boldsymbol{\theta})$
- with parameters $\boldsymbol{\theta}$ learned by the neural network?

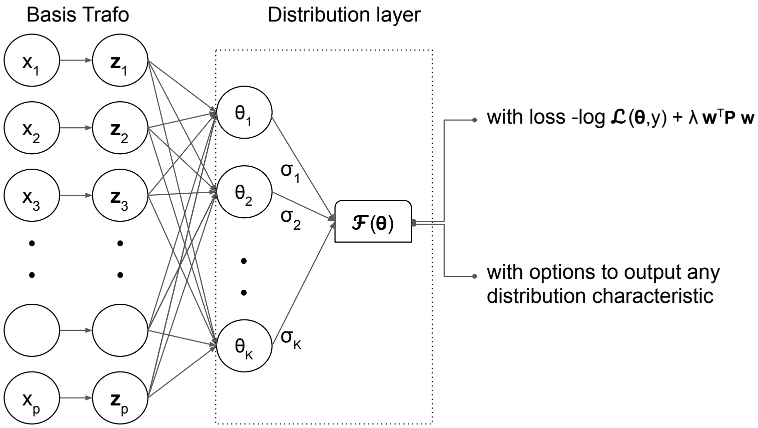


with loss $-\log \mathcal{L}(\hat{y}, y) + \lambda \mathbf{w}^T \mathbf{P} \mathbf{w}$

Network with Distribution Layer



A GAMLSS



Basis Trafo
Distribution layer
x_1
x_2
x_3
x_p
$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_3$
$\mathbf{z}_p$
θ_1
θ_2
θ_K
σ_1
σ_2
σ_K
$\mathcal{F}(\boldsymbol{\theta})$
with loss $-\log \mathcal{L}(\boldsymbol{\theta}, y) + \lambda\, \mathbf{w}^\top \mathbf{P}\, \mathbf{w}$
with options to output any distribution characteristic

Distribution Layers Extended

- not restricted to standard distributions:

Distribution Layers Extended

- not restricted to standard distributions:
  - arbitrary mixtures of distributions

Distribution Layers Extended

- not restricted to standard distributions:
  - arbitrary mixtures of distributions
  - bijectors allowing for diffeomorphisms

Distribution Layers Extended

- not restricted to standard distributions:
  - arbitrary mixtures of distributions
  - bijectors allowing for diffeomorphisms
  - (autoregressive) flows [6]

- not restricted to standard distributions:
  - arbitrary mixtures of distributions
  - bijectors allowing for diffeomorphisms
  - (autoregressive) flows [6]
  - ...

Distribution Layers Extended

- not restricted to standard distributions:
  - arbitrary mixtures of distributions
  - bijectors allowing for diffeomorphisms
  - (autoregressive) flows [6]
  - ...
- still auto differentiation

Neural Networks beyond Classical Regression

NNs are very flexible in their model specification.

We can add:

The benefit of NN + Stat.Regression

NNs are very flexible in their model specification.

We can add:

- Higher-order Effects
    - Additive models are easy to interpret and understand,
    - but what if the data generating process is more complex

The benefit of NN + Stat.Regression

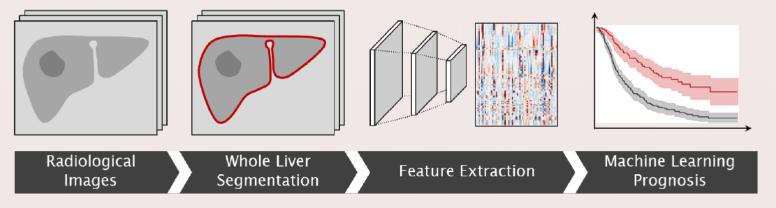
NNs are very flexible in their model specification.

We can add:

- Higher-order Effects
  - Additive models are easy to interpret and understand,
  - but what if the data generating process is more complex
- Additional non-tabular data
  - Extracting features from images, texts, ... is tedious
  - Not an end-to-end approach

Many use cases have additional non-tabular data

- Medicine: Patient info (tabular), but also with scans



Source: Ingrisch (2020)

- Psychology: Questionnaires, with open-ended questions

Source: `https://trinachi.github.io/`

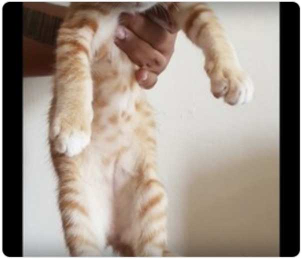
Learn from multiple data modalities + tabular data

There are 8,193 homeless doggies here.
Let's find a home for them now!

Lavender
Female, 2 Mths, Mixed Breed
Kuala Lumpur, by mobula

Lavender, a 10-week (estimated) old girl has a short but very eventful history. She was abandoned together with her two sisters by the roadside around 12 November. And a few days after a group of dog lovers in the neighborhood in..

There are 6,348 neglected kitties here.
Shall we give them a loving family?



Noodle
Male, 4 Mths, Domestic Short Hair
Selangor, by catleyow

Noodle is a gentle, affectionate rascal, who loves human company and is always ready to play. He has left the 'teething' age, so won't be a bitey or scratchy kitten; he'd rather lie in your arms and be cuddled! His sociabl..

Source: Petfinder.my

A Unified Framework

Goal

1. All commonly used regression models within NN
   - with the exact same model being optimized
   - comparable results to classical statistical routines

Goal

1. All commonly used regression models within NN
    - with the exact same model being optimized
    - comparable results to classical statistical routines
2. Also allow for deep neural network (DNN)
    - ensure that this does not conflict with first goal

Free to add any other deep neural network to θ_k

$$\theta_k = \sigma_k(\eta_k) = \sigma(\beta_{k,0} + \boldsymbol{x}\boldsymbol{\beta}_k + \mathrm{DNN}_k)$$

$\Rightarrow$ Framework = Structured Effects + Unstructured Effects
$\Rightarrow$ DNN_k allows to

- include unstructured data sources, like images, texts, etc.
- model higher-order (non-linear) interaction effects

Free to add any other deep neural network to θ_k

$$\theta_k = \sigma_k(\eta_k) = \sigma(\beta_{k,0} + \boldsymbol{x}\boldsymbol{\beta}_k + \mathrm{DNN}_k)$$

$\Rightarrow$ Framework = Structured Effects + Unstructured Effects
$\Rightarrow$ DNN_k allows to

- include unstructured data sources, like images, texts, etc.
- model higher-order (non-linear) interaction effects

... but we have to enforce

- identifiability of structured effects
- meaningful decomposition

Axiomatic Properties

1. SDDR without DNN / DNN with no influence
   $\rightarrow$ should yield the structured additive model

1. SDDR without DNN / DNN with no influence
   $\longrightarrow$ should yield the structured additive model
2. SDDR with no influence of structured model part
   $\longrightarrow$ structured effects should be zero

Axiomatic Properties

1. SDDR without DNN / DNN with no influence
   $\longrightarrow$ should yield the structured additive model
2. SDDR with no influence of structured model part
   $\longrightarrow$ structured effects should be zero
3. Structured effect and DNN effect of same covariates
   $\longrightarrow$ ensure identifiability

Axiomatic Properties

1. SDDR without DNN / DNN with no influence
   $\longrightarrow$ should yield the structured additive model
2. SDDR with no influence of structured model part
   $\longrightarrow$ structured effects should be zero
3. Structured effect and DNN effect of same covariates
   $\longrightarrow$ ensure identifiability

$\Rightarrow$ via Orthogonaliazation

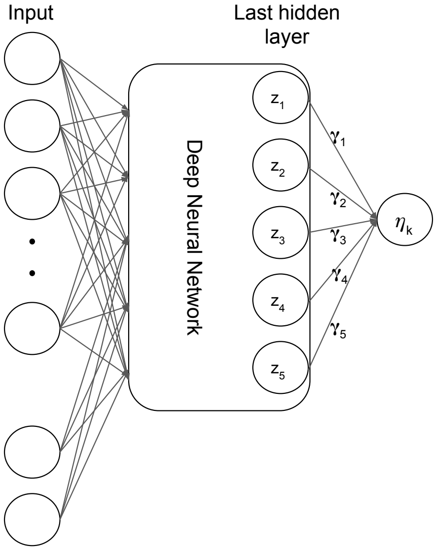
Orthogonalization

Separates space in its different components

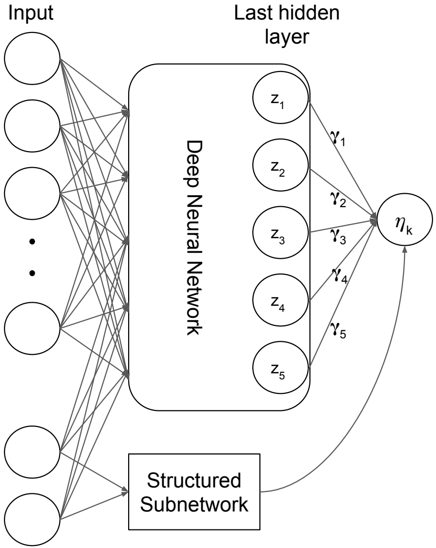
- Linear space spanned by columns of structured covariates $\mathcal{X}$
- Orthogonal complement $\mathcal{X}^\perp$
- DNN is projected into $\mathcal{X}^\perp$ $\rightarrow$ happens within the graph

$\Rightarrow$ Ensures identifiability

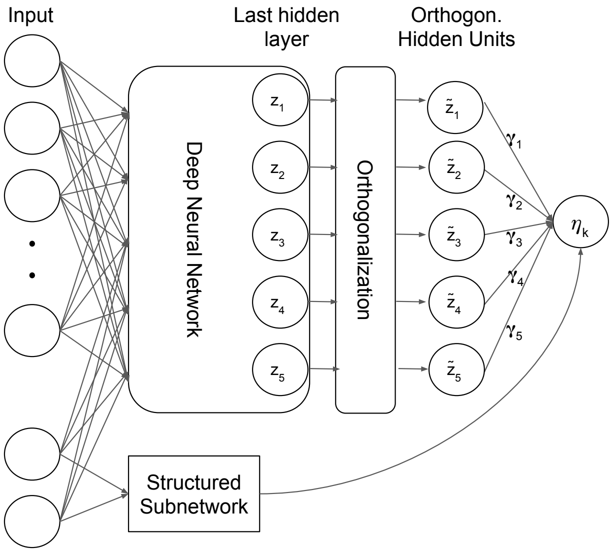
Orthogonalization - High Level View



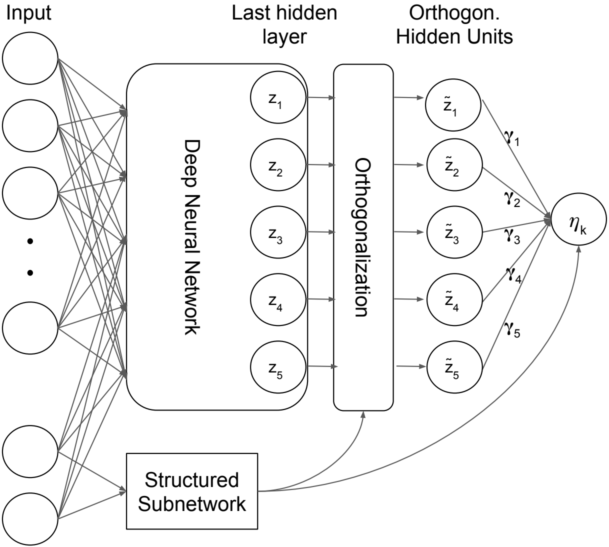
Orthogonalization - High Level View



Orthogonalization - High Level View



Orthogonalization - High Level View



Extensions

Extensions

- Within SDDR
  - Multivariate outcome models
  - Inference for weights $\rightarrow$ turn SDDR into a Bayesian NN
- Other proposed derivates
  - Neural Mixture Density Regression (w/ Pfisterer & Bischl)
  - Deep Conditional Transformation Models (w/ Baumann & Hothorn)
  - Deep Piecewise Exponential Models (w/ Kopper, Bender et al.)
- Upcoming
  - SDDR for functional data
  - SDDR for time series data
  - Further Orthogonalization use cases

Software

Implementation

- Implemented in R package `deepregression`
- Basis Transformations using `s`-/`ti`-/`te`-terms from `mgcv`
- DNN definition in `keras`
- Graph building and model training in Python / TensorFlow

Example and Details I

Example:

$$Y = \beta_0 + f_1(x) + \mathrm{DNN}_1(x) + \varepsilon$$

with

$$\varepsilon \sim \mathcal{N}(0, \sigma)$$

and we model

$$\sigma = \exp\left(\sum_k I(fac = k)\beta_k + f_2(z) + \mathrm{DNN}_2(a, b, c, d)\right).$$

Exemplary DNN specifications:

```
deep_mod1 <- function(x) x %>%
      layer_dense(units = 1, activation = "linear")
```

```
deep_mod2 <- function(x) x %>%
      layer_dense(units = 32, use_bias = FALSE) %>%
      layer_dropout(rate = 0.2) %>%
      layer_dense(units = 8, activation = "relu") %>%
      layer_dense(units = 1, activation = "linear")
```

Example ctd.:

```
mod <- deepregression(
           y = y,
           data = data,
           list_of_formulae = list(
             location = 1 + s(x, bs = 'ps') + deep_mod1(x),
             scale    = 0 + fac + s(z) + deep_mod2(a,b,c,d)),
           list_of_deep_models = list(deep_mod1, deep_mod2),
           family = "normal",
           df = 10
       )

history <- mod %>% fit(epochs=100)
```

Example and Details IV

Many convenience functions available for the fitted mod:

- **coef** for structured model coefficients
- **plot** for plotting smooth terms
- **cv** for tuning the model
- **get_distribution** to access fitted distribution
- **predict** for prediction on new data
- ...

Release

- Still in a private Github repo
- If you want to use the Beta version, let me know
- Hopefully open-sourced end of year

Some Results

Decomposition

- Simulation to demonstrate identifiability of structured effects in DNN presence

Decomposition

- Simulation to demonstrate identifiability of structured effects in DNN presence
- Data generating process:

$$\mathbb{E}(Y|\boldsymbol{x}) = \beta_0 + \boldsymbol{x}\boldsymbol{\beta} + \sum_{j=1}^{10} f_j(x_j) + \prod_{j=1}^{10} x_j$$

Decomposition

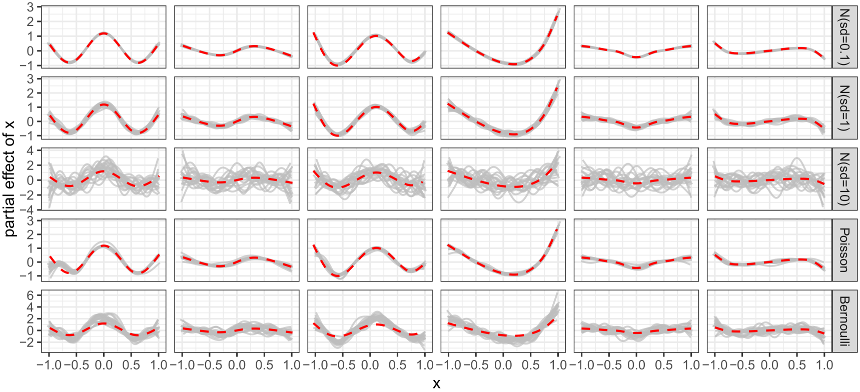
- Simulation to demonstrate identifiability of structured effects in DNN presence
- Data generating process:

$$\mathbb{E}(Y|\boldsymbol{x}) = \beta_0 + \boldsymbol{x}\boldsymbol{\beta} + \sum_{j=1}^{10} f_j(x_j) + \prod_{j=1}^{10} x_j$$

- Model:

```
location = +1 + x1 + ...+ x10 +
              s(x1) + ...+ s(x10) +
              deep(x1, ..., x10)
```

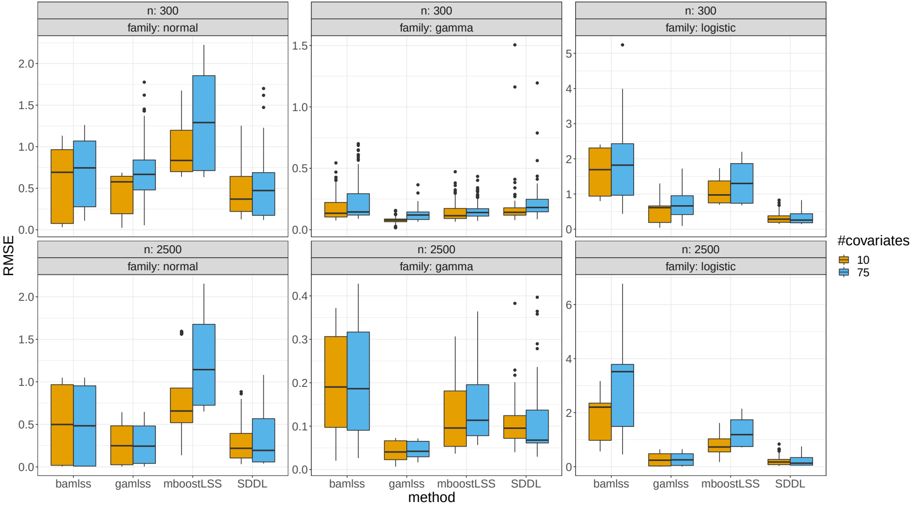
Estimated additive effects $f_1, \ldots, f_6$ (columns):



Model Comparison LSS Approaches

- Simulation to demonstrate estimation performance and goodness-of-fit
- Compared to `bamlss`, `gamlss`, `mboostLSS`
- Additive model with Normal, Gamma and Logistic distribution
- $n \in \{300, 2500\}$, $p \in \{10, 75\}$

RMSE of coefficients (w/o outliers from `bamlss` and `gamlss`)



Deep Conditional Transformation Models

- Movies Review Dataset

Deep Conditional Transformation Models

- Movies Review Dataset
- Model the conditional CDF of movie revenue Y non-parametrically

Deep Conditional Transformation Models

- Movies Review Dataset
- Model the conditional CDF of movie revenue Y non-parametrically
- Using transformation models:

$$\mathbb{P}(Y \leq y|\boldsymbol{x}) = F_Z(h(y|\boldsymbol{x}))$$

with error distribution F_Z

Deep Conditional Transformation Models

- Movies Review Dataset
- Model the conditional CDF of movie revenue Y non-parametrically
- Using transformation models:

$$\mathbb{P}(Y \leq y|\boldsymbol{x}) = F_Z(h(y|\boldsymbol{x}))$$

with error distribution F_Z and transformation function

$$h(y|\boldsymbol{x}) = \boldsymbol{a}(y)^\top \underbrace{\boldsymbol{\vartheta}(\boldsymbol{x})}_{\text{Interaction Term}} + \underbrace{\beta(\boldsymbol{x})}_{\text{Shift Term}}$$

Application - Deep Transformation Model II

- For tabular data, we define the predictor

$$\sum_{r=1}^{20} \beta_{r,t} I(genre_i = r) + s_{1,t}(popularity_i) + \\ s_{2,t}(releasedate_i) + s_{3,t}(budget_i) + s_{4,t}(runtime_i)$$

for both Shift and Interaction

- For tabular data, we define the predictor

$$\sum_{r=1}^{20} \beta_{r,t} I(genre_i = r) + s_{1,t}(popularity_i) + \\ s_{2,t}(releasedate_i) + s_{3,t}(budget_i) + s_{4,t}(runtime_i)$$

  for both Shift and Interaction
- We also use the movie description in a DNN:

- For tabular data, we define the predictor

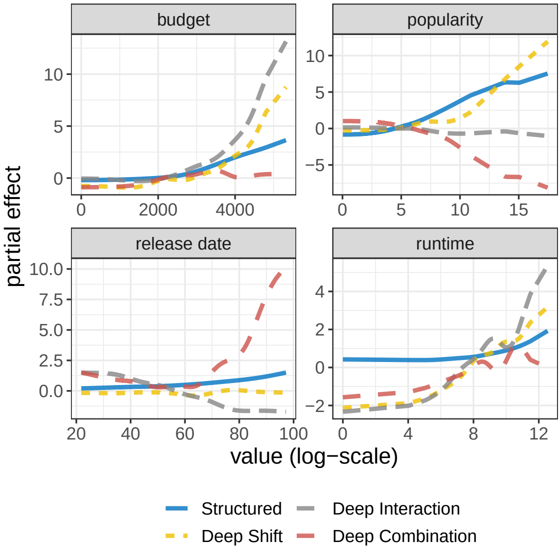
$$\sum_{r=1}^{20} \beta_{r,t} I(genre_i = r) + s_{1,t}(popularity_i) + \\ s_{2,t}(releasedate_i) + s_{3,t}(budget_i) + s_{4,t}(runtime_i)$$

for both Shift and Interaction

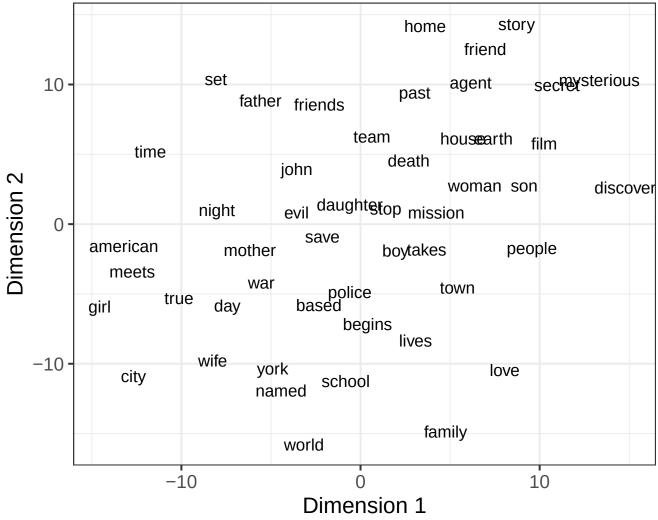
- We also use the movie description in a DNN:

Structured:	no DNN
Deep Shift:	embedding + FC layer in Shift term
Deep Interaction:	embedding + FC layer in Interaction term
Deep Combination:	embedding + FC layer fed in both terms

Application - Deep Transformation Model III



t-SNE of learned embedding space for 50 most freq. words



- Comparison with Transformation Boosting Machines (TBM, [3])
- Based on averaged predicted log-scores (PLS) on test data

Model	mean PLS (SD)
Structured	**-4.84** (3.10)
Deep Shift	-52.58 (21.06)
Deep Interaction	-20.68 (11.80)
Deep Combination	-24.64 (13.00)
TBM-Shift	-23.31 (0.83)
TBM-Distribution	-22.38 (0.31)

Summary

Summary

- Statistical Regression can be embedded into NN
  - feasibility in high-dimensional settings
  - straightforward extensions of existing model classes
- SDDR
  - unified network architecture
  - to fit (distributional) regression models
  - options to add arbitrary DNN
  - ensures identifiability
- `deepregression`
  - implementation of SDDR in R
  - various models using familiar R interface

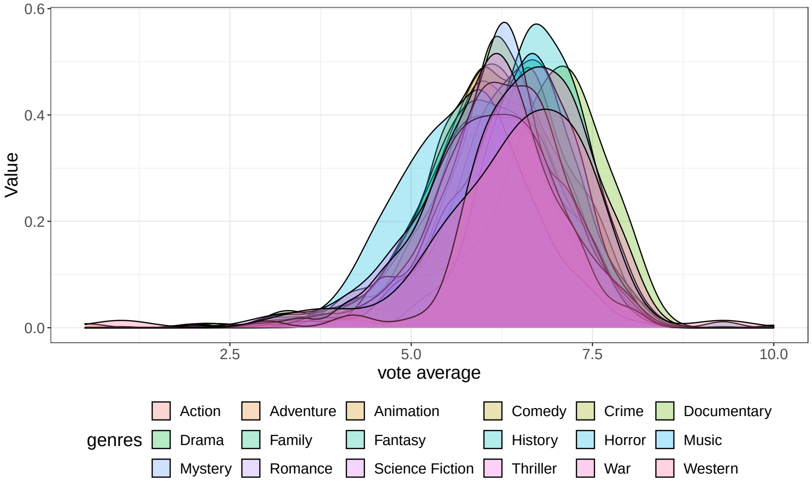
Appendix

Application - Mixture Models I

Movie Review Dataset

- Movie Reviews from 0 to 10
- Tabular information like revenue, release date, ...
- genres $\rightarrow$ one movie can have multiple genres

Ratings for different genres



- We define a mixture model of 18 beta distributions

Application - Mixture Models III

- We define a mixture model of 18 beta distributions
- distribution parameters c_0, c_1 of all 18 mixtures are modeled via

$$s_{1,m,k}(budget_i) + s_{2,m,k}(popularity_i) +$$
$$s_{3,m,k}(runtime_i) + s_{4,m,k}(releasedate_i)$$

for mixture m and parameter $k \in \{0, 1\}$

- We define a mixture model of 18 beta distributions
- distribution parameters c_0, c_1 of all 18 mixtures are modeled via

$$s_{1,m,k}(budget_i) + s_{2,m,k}(popularity_i) +$$
$$s_{3,m,k}(runtime_i) + s_{4,m,k}(releasedate_i)$$

  for mixture m and parameter $k \in \{0, 1\}$
- Movie description $\longrightarrow$ embedding layer + FC layer

Models:

(I) : Only structured predictor

(II) DNN with 18 output units fed into c_0's

(III) DNN with 18 output units fed into c_1's

(IV) DNN with 36 output units fed into c_0's and c_1's

(V) DNN with 1 output unit fed into linear predictor of $\boldsymbol{\pi}$

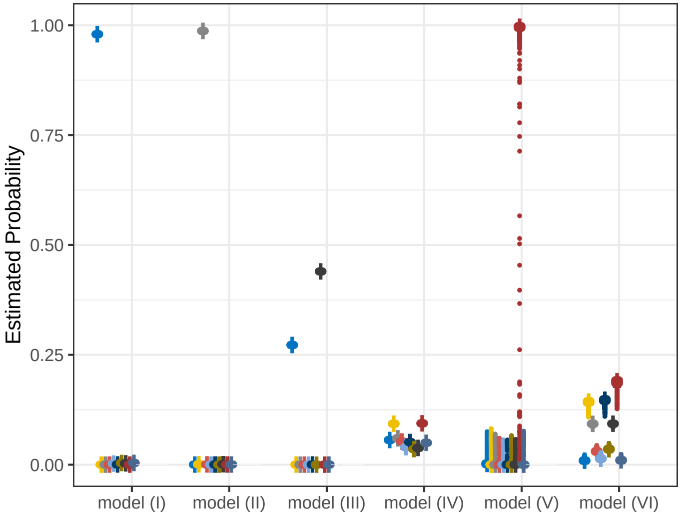
(VI) Combination of (IV) and (V)

Application - Mixture Models V

Mean RMSE values (standard deviation in brackets) on test data

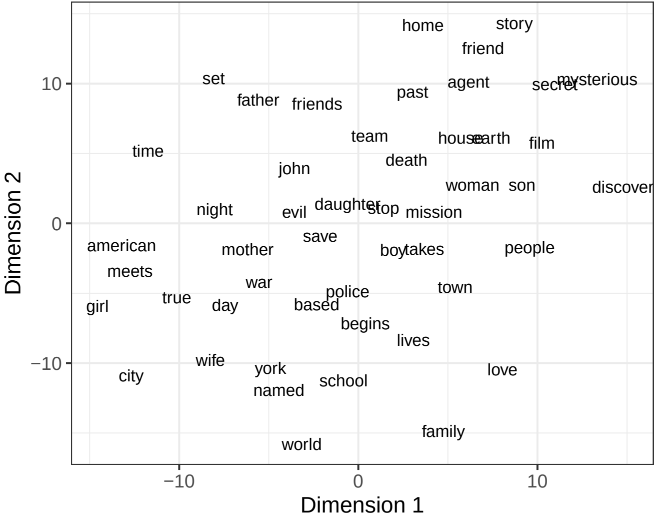
Model	Mean RMSE
(I)	0.242 (0.128)
(II)	0.176 (0.122)
(III)	0.213 (0.117)
(IV)	0.321 (0.156)
(V)	**0.117** (0.026)
(VI)	0.190 (0.090)

Estimated mixture components for each model



LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

t-SNE of model (V) embedding space for 50 most freq. words



SDDR can also be turned into a Bayesian NN (BNN)

Extension: Epistemic uncertainty

SDDR can also be turned into a Bayesian NN (BNN)

- A BNN defines (prior) distributions over weights $\boldsymbol{w}$

Extension: Epistemic uncertainty

SDDR can also be turned into a Bayesian NN (BNN)

- A BNN defines (prior) distributions over weights $\boldsymbol{w}$
- The corresponding posterior $p(\boldsymbol{w}|\boldsymbol{x})$ is usually intractable

Extension: Epistemic uncertainty

SDDR can also be turned into a Bayesian NN (BNN)

- A BNN defines (prior) distributions over weights $\boldsymbol{w}$
- The corresponding posterior $p(\boldsymbol{w}|\boldsymbol{x})$ is usually intractable
- Variational inference: Define approximate posterior

SDDR can also be turned into a Bayesian NN (BNN)

- A BNN defines (prior) distributions over weights $\boldsymbol{w}$
- The corresponding posterior $p(\boldsymbol{w}|\boldsymbol{x})$ is usually intractable
- Variational inference: Define approximate posterior
  - variational posterior $q(\boldsymbol{w}|\boldsymbol{\vartheta})$
  - variational parameters $\boldsymbol{\vartheta}$

SDDR can also be turned into a Bayesian NN (BNN)

- A BNN defines (prior) distributions over weights $\boldsymbol{w}$
- The corresponding posterior $p(\boldsymbol{w}|\boldsymbol{x})$ is usually intractable
- Variational inference: Define approximate posterior
    - variational posterior $q(\boldsymbol{w}|\boldsymbol{\vartheta})$
    - variational parameters $\boldsymbol{\vartheta}$
- network is trained by minimizing the ELBO criterion

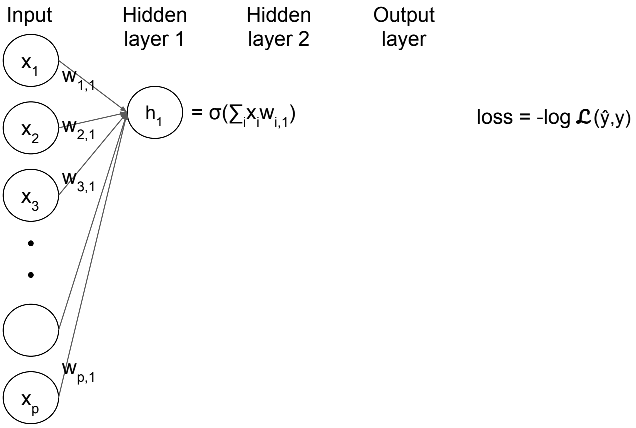
$$\mathrm{KL}_q[q(\boldsymbol{w}|\boldsymbol{\vartheta}) \,||\, p(\boldsymbol{w}|\boldsymbol{x})] - \mathbb{E}_q[\log \mathcal{L}(\boldsymbol{w})]$$

SDDR can also be turned into a Bayesian NN (BNN)

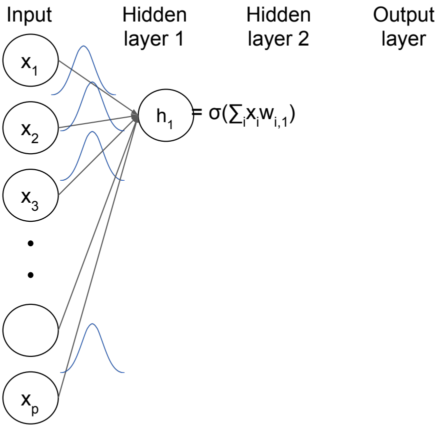
- A BNN defines (prior) distributions over weights $\boldsymbol{w}$
- The corresponding posterior $p(\boldsymbol{w}|\boldsymbol{x})$ is usually intractable
- Variational inference: Define approximate posterior
  - variational posterior $q(\boldsymbol{w}|\boldsymbol{\vartheta})$
  - variational parameters $\boldsymbol{\vartheta}$
- network is trained by minimizing the ELBO criterion

$$\mathrm{KL}_q[q(\boldsymbol{w}|\boldsymbol{\vartheta}) \;||\; p(\boldsymbol{w}|\boldsymbol{x})] - \mathbb{E}_q[\log \mathcal{L}(\boldsymbol{w})]$$

using the *Bayes by Backprop* [1] algorithm



Input
Hidden layer 1
Hidden layer 2
Output layer
x_1
x_2
x_3
x_p
$w_{1,1}$
$w_{2,1}$
$w_{3,1}$
$w_{p,1}$
h_1
$= \sigma(\sum_i x_i w_{i,1})$
loss = $-\log \mathcal{L}(\hat{y}, y)$



$$\text{loss} = E_q \, -\log \mathcal{L}(\hat{y}, y) + \text{KL(prior, var. posterior)}$$

Bibliography I

[1] Charles Blundell et al. “Weight uncertainty in neural networks”. In: *Proceedings of the 32nd International Conference on Machine Learning* 37 (2015), pp. 1613–1622.

[2] Joshua V Dillon et al. “Tensorflow distributions”. In: *arXiv preprint arXiv:1711.10604* (2017).

[3] Torsten Hothorn. “Transformation boosting machines”. In: *Statistics and Computing* 30.1 (2020), pp. 141–152.

[4] Jiquan Ngiam et al. “Multimodal deep learning”. In: *Proceedings of the 28th International Conference on Machine Learning (ICML)*. 2011, pp. 689–696.

[5] Victor M-H Ong, David J Nott, and Michael S Smith. “Gaussian variational approximation with a factor covariance structure”. In: *Journal of Computational and Graphical Statistics* 27.3 (2018), pp. 465–478.

[6] George Papamakarios, Theo Pavlakou, and Iain Murray. “Masked autoregressive flow for density estimation”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 2338–2347.

[7] Hao Song et al. “Distribution calibration for regression”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 5897–5906.