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)
1 Statistical Regression in Neural Networks
2 Neural Networks beyond Classical Regression
3 A Unified Framework
4 Extensions
5 Software
6 Some Results
7 Summary
Statistical Regression in Neural Networks
• A neuron $h$ in a neural network (NN) is a transformed linear combination

• For one observation ($n = 1$):

  Input: $\mathbf{x} \in \mathbb{R}^{1 \times p}$
  Weights: $\mathbf{w} \in \mathbb{R}^p$

  Input times weights: $\eta := \mathbf{x} \mathbf{w} = \sum_j x_j w_j \in \mathbb{R}$
  Output: $h = \sigma(\eta) \in \mathbb{R}$

with activation function $\sigma(\cdot)$
Some Neural Network
Some Neural Network

Input  Hidden  Hidden  Output
\( x_1 \)  layer 1  layer 2  layer
\( x_2 \)
\( x_3 \)
\( \ldots \)
\( x_p \)
Some Neural Network

Input layer 1

Hidden layer 2

Output layer

\[ x_1 \]
\[ x_2 \]
\[ x_3 \]
\[ \ldots \]
\[ x_p \]
Some Neural Network

Input layer 1

Hidden layer 2

Output layer

\[ h_1 = \sigma(\sum x_i w_{i,1}) \]

\[ w_{1,1}, w_{2,1}, w_{3,1}, \ldots, w_{p,1} \]
A Linear Model

\[ \hat{y} = \sum_{i} x_i w_{i,1} \]

with loss \( (\hat{y} - y)^2 \)
A Generalized Linear Model

\[ \hat{y} = \sigma(\sum_i x_i w_{i,1}) \]

with loss \(-\log \mathcal{L}(\hat{y}, y), \quad y \sim \text{ExpFam}(\theta)\)
Ridge Regression

\[ \hat{y} = \sigma(\sum_i x_i w_{i,1}) \]

with loss 
\[ -\log \mathcal{L}(\hat{y}, y) + \lambda \sum_j w_j^2 \]
\[
\hat{y} = \sigma(\sum_i x_i w_{i,1}) \quad \text{with loss } -\log \mathcal{L}(\hat{y}, y) + \lambda \sum_j |w_j|
\]
More Complex Penalties

\[
\hat{y} = \sigma(\sum_i x_i w_{i,1})
\]

with loss \(-\log \mathcal{L}(\hat{y}, y) + \lambda w^T P w\)
A Generalized Additive Model (GAM)

\[ \hat{y} = \sigma(zw) \]

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

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...
Idea [2]: Instead of using an output layer that just creates $\hat{y}$
- why not return a parametric distribution $\mathcal{F}(\theta)$
- with parameters $\theta$ learned by the neural network?
The Network from before

\[ \hat{y} = \sigma(zw) \]

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

\[ \mathcal{L}(\theta, y) + \lambda \mathbf{w}^\top \mathbf{P} \mathbf{w} \]

with options to output any distribution characteristic
A GAMLSS

\[ \text{Distribution layer with loss } -\log L(\theta, y) + \lambda w^T P w \]

\[ \text{with options to output any distribution characteristic} \]

Basis Trafo

\[ x_1 \rightarrow z_1 \]
\[ x_2 \rightarrow z_2 \]
\[ x_3 \rightarrow z_3 \]
\[ \ldots \]
\[ x_p \rightarrow z_p \]

Distribution layer

\[ \theta_1 \]
\[ \theta_2 \]
\[ \theta_K \]
\[ \sigma_1 \]
\[ \sigma_2 \]
\[ \sigma_K \]
\[ \mathcal{F}(\theta) \]
• not restricted to standard distributions:
• not restricted to standard distributions:
  • arbitrary mixtures of distributions
• not restricted to standard distributions:
  • arbitrary mixtures of distributions
  • bijectors allowing for diffeomorphisms
• 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:
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
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

Closed question:

Why don’t you eat ice cream at Fictionals Ice Cream Parlour?
(Choose at least one answer.)

- I don’t like the flavours
- It’s too expensive
- The service is bad
- I don’t like the ice cream
- It’s too far from my house
- I don’t know

Open-ended question:

Why don’t you eat ice cream at Fictionals Ice Cream Parlour?

I am lactose intolerant so I can’t eat most ice creams, and it’s really hard to find a store that offers good lactose-free ice cream. I’ve never heard of Fictionals but if I knew that they offered some, I would definitely try them out because I love ice cream!

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.
A Unified Framework
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 $\theta_k$

$$\theta_k = \sigma_k(\eta_k) = \sigma(\beta_{k,0} + x\beta_k + \text{DNN}_k)$$

$\Rightarrow$ 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 $\theta_k$

$$
\theta_k = \sigma_k(\eta_k) = \sigma(\beta_{k,0} + x\beta_k + \text{DNN}_k)
$$

⇒ Framework = Structured Effects + Unstructured Effects
⇒ 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
1. SDDR without DNN / DNN with no influence → should yield the structured additive model
Axiomatic Properties

1. SDDR without DNN / DNN with no influence
   → should yield the structured additive model

2. SDDR with no influence of structured model part
   → structured effects should be zero
Axiomatic Properties

1. SDDR without DNN / DNN with no influence → should yield the structured additive model

2. SDDR with no influence of structured model part → structured effects should be zero

3. Structured effect and DNN effect of same covariates → ensure identifiability
1. SDDR without DNN / DNN with no influence
   → should yield the structured additive model

2. SDDR with no influence of structured model part
   → structured effects should be zero

3. Structured effect and DNN effect of same covariates
   → ensure identifiability

⇒ via Orthogonalization
Separates space in its different components

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

$\Rightarrow$ Ensures identifiability
Deep Neural Network

Input

\[ z_1, z_2, z_3, z_4, z_5 \]

Last hidden layer

\[ \eta_k, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5 \]

Deep Neural Network

Orthogonalization - High Level View

David Rügamer (LMU) Semi-Structured Deep Distributional Regression
Orthogonalization - High Level View

Deep Neural Network

Input

Last hidden layer

Orthogonalization
Hidden Units

Structured Subnetwork

\( \eta_k \)
Orthogonalization - High Level View

Deep Neural Network

Input

Structured Subnetwork

Last hidden layer

Orthogonalization

Orthogon. Hidden Units

Deep Neural Network

\( z_1 \)
\( z_2 \)
\( z_3 \)
\( z_4 \)
\( z_5 \)

Orthogonalization - High Level View

\( z̃_1 \)
\( z̃_2 \)
\( z̃_3 \)
\( z̃_4 \)
\( z̃_5 \)

\( \eta_k \)

\( \gamma_1 \)
\( \gamma_2 \)
\( \gamma_3 \)
\( \gamma_4 \)
\( \gamma_5 \)
Extensions
Extensions

- **Within SDDR**
  - Multivariate outcome models
  - Inference for weights → 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
• Implemented in R package deepregression
• Basis Transformations using \texttt{s-/ti-/te-terms} from \texttt{mgcv}
• DNN definition in \texttt{keras}
• Graph building and model training in Python / TensorFlow
Example:

\[ Y = \beta_0 + f_1(x) + \text{DNN}_1(x) + \epsilon \]

with

\[ \epsilon \sim \mathcal{N}(0, \sigma) \]

and we model

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

```r
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)
```
Many convenience functions available for the fitted model:

- `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
- ...
• 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:

\[
\mathbb{E}(Y|x) = \beta_0 + x^T \beta + \sum_{j=1}^{10} f_j(x_j) + \prod_{j=1}^{10} x_j
\]
Decomposition

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

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

- Model:

\[
\text{location} = +1 + x1 + \ldots + x10 + \\
\quad s(x1) + \ldots + s(x10) + \\
\quad \text{deep}(x1, \ldots, x10)
\]
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)

Simulation Results - Comparisons II

David Rügamer (LMU) Semi-Structured Deep Distributional Regression
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:

$$P(Y \leq y | x) = F_Z(h(y|x))$$

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:

$$P(Y \leq y|\mathbf{x}) = F_{Z}(h(y|\mathbf{x}))$$

with error distribution $F_{Z}$ and transformation function

$$h(y|\mathbf{x}) = a(y)^\top \vartheta(\mathbf{x}) + \beta(\mathbf{x})$$

Interaction Term  Shift Term
For tabular data, we define the predictor

\[
\sum_{r=1}^{20} \beta_{r,t}(\text{genre}_i = r) + s_{1,t}(\text{popularity}_i) + \\
\quad s_{2,t}(\text{releasedate}_i) + s_{3,t}(\text{budget}_i) + s_{4,t}(\text{runtime}_i)
\]

for both Shift and Interaction
• For tabular data, we define the predictor

\[ \sum_{r=1}^{20} \beta_{r,t} I(\text{genre}_i = r) + s_{1,t}(\text{popularity}_i) + s_{2,t}(\text{releasedate}_i) + s_{3,t}(\text{budget}_i) + s_{4,t}(\text{runtime}_i) \]

for both Shift and Interaction

• We also use the movie description in a DNN:
• For tabular data, we define the predictor

\[
\sum_{r=1}^{20} \beta_{r,t}(\text{genre}_i = r) + s_{1,t}(\text{popularity}_i) + \\
 s_{2,t}(\text{releasedate}_i) + s_{3,t}(\text{budget}_i) + s_{4,t}(\text{runtime}_i)
\]

for both Shift and Interaction

• We also use the movie description in a DNN:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured:</td>
<td>no DNN</td>
</tr>
<tr>
<td>Deep Shift:</td>
<td>embedding + FC layer in Shift term</td>
</tr>
<tr>
<td>Deep Interaction:</td>
<td>embedding + FC layer in Interaction term</td>
</tr>
<tr>
<td>Deep Combination:</td>
<td>embedding + FC layer fed in both terms</td>
</tr>
</tbody>
</table>
Application - Deep Transformation Model III

![Graph showing the partial effect of various factors on budget, popularity, release date, and runtime. The factors include Structured, Deep Interaction, Deep Shift, and Deep Combination.](image)

- **Structured**
- **Deep Interaction**
- **Deep Shift**
- **Deep Combination**

David Rügamer (LMU) Semi-Structured Deep Distributional Regression
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

<table>
<thead>
<tr>
<th>Model</th>
<th>mean PLS (SD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured</td>
<td>-4.84 (3.10)</td>
</tr>
<tr>
<td>Deep Shift</td>
<td>-52.58 (21.06)</td>
</tr>
<tr>
<td>Deep Interaction</td>
<td>-20.68 (11.80)</td>
</tr>
<tr>
<td>Deep Combination</td>
<td>-24.64 (13.00)</td>
</tr>
<tr>
<td>TBM-Shift</td>
<td>-23.31 (0.83)</td>
</tr>
<tr>
<td>TBM-Distribution</td>
<td>-22.38 (0.31)</td>
</tr>
</tbody>
</table>
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
Movie Review Dataset

- Movie Reviews from 0 to 10
- Tabular information like revenue, release date, ...
- genres → one movie can have multiple genres
Ratings for different genres

genres
- Action
- Adventure
- Animation
- Comedy
- Crime
- Documentary
- Drama
- Family
- Fantasy
- History
- Horror
- Music
- Mystery
- Romance
- Science Fiction
- Thriller
- War
- Western
• We define a mixture model of 18 beta distributions
• We define a mixture model of 18 beta distributions
• distribution parameters $c_0, c_1$ of all 18 mixtures are modeled via

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

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

$$s_{1,m,k}(\text{budget}_i) + s_{2,m,k}(\text{popularity}_i) +$$
$$s_{3,m,k}(\text{runtime}_i) + s_{4,m,k}(\text{releasedate}_i)$$

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

• Movie description $\rightarrow$ 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 $\pi$
(VI) Combination of (IV) and (V)
Mean RMSE values (standard deviation in brackets) on test data

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(I)</td>
<td>0.242 (0.128)</td>
</tr>
<tr>
<td>(II)</td>
<td>0.176 (0.122)</td>
</tr>
<tr>
<td>(III)</td>
<td>0.213 (0.117)</td>
</tr>
<tr>
<td>(IV)</td>
<td>0.321 (0.156)</td>
</tr>
<tr>
<td>(V)</td>
<td>0.117 (0.026)</td>
</tr>
<tr>
<td>(VI)</td>
<td>0.190 (0.090)</td>
</tr>
</tbody>
</table>
Estimated mixture components for each model
t-SNE of model (V) embedding space for 50 most freq. words
SDDR can also be turned into a Bayesian NN (BNN)
SDDR can also be turned into a Bayesian NN (BNN)

- A BNN defines (prior) distributions over weights $w$
SDDR can also be turned into a Bayesian NN (BNN)

- A BNN defines (prior) distributions over weights \( w \)
- The corresponding posterior \( p(w|x) \) is usually intractable
SDDR can also be turned into a Bayesian NN (BNN)

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

\[ \text{KL}[q(w|\vartheta)||p(w|x)] - \mathbb{E}_q[\log L(w)] \]
SDDR can also be turned into a Bayesian NN (BNN)

- A BNN defines (prior) distributions over weights $w$
- The corresponding posterior $p(w|x)$ is usually intractable
- Variational inference: Define approximate posterior
  - variational posterior $q(w|\vartheta)$
  - variational parameters $\vartheta$
SDDR can also be turned into a Bayesian NN (BNN)

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

$$
\text{KL}_q[q(w|\theta) \mid \mid p(w|x)] - E_q[\log L(w)]
$$
SDDR can also be turned into a Bayesian NN (BNN)

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

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

using the *Bayes by Backprop* [1] algorithm
Bayesian / Variational Layers (ctd.)

\[ h_1 = \sigma(\sum_i x_i w_{i,1}) \]

\[ \text{loss} = -\log \mathcal{L}(\hat{y}, y) \]
Bayesian / Variational Layers (ctd.)

\[ h_1 = \sigma(\sum_i x_i w_{i,1}) \]

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


