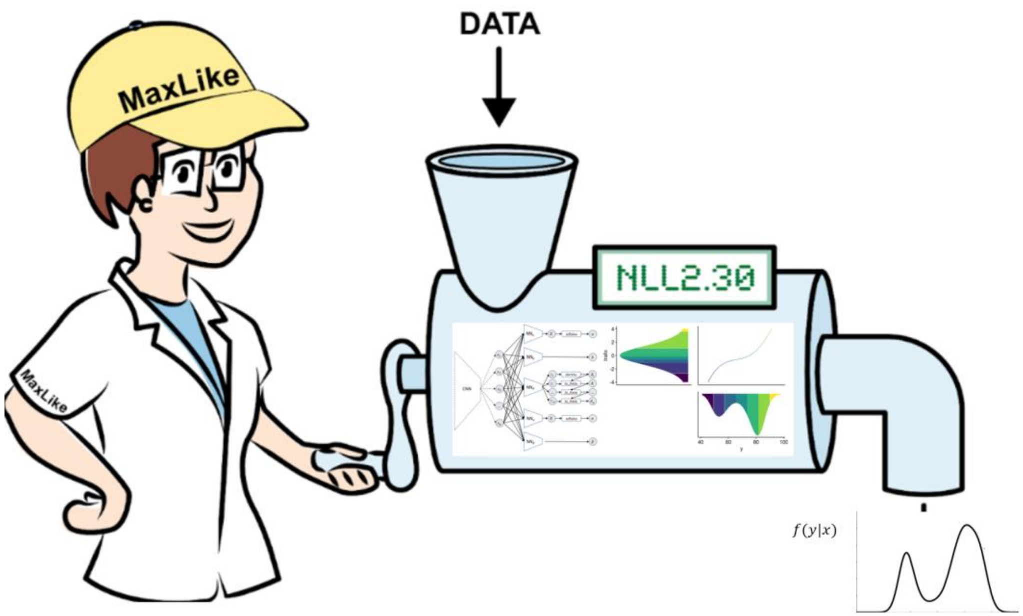
Deep transformation models
for tackling complex probabilistic regression problems

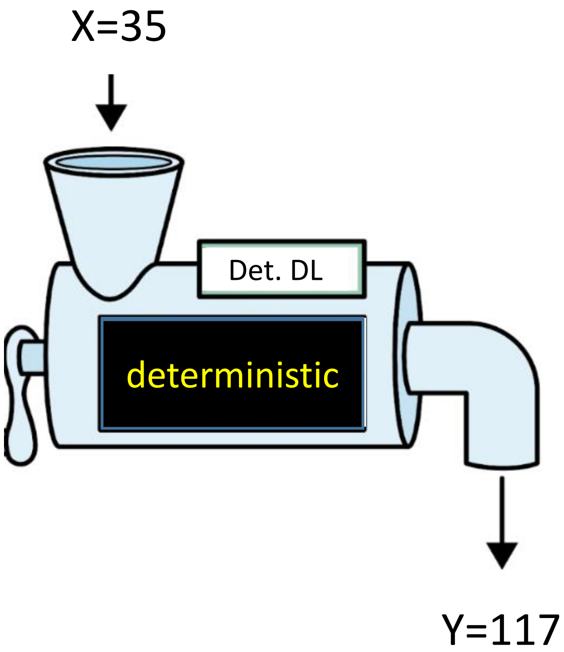


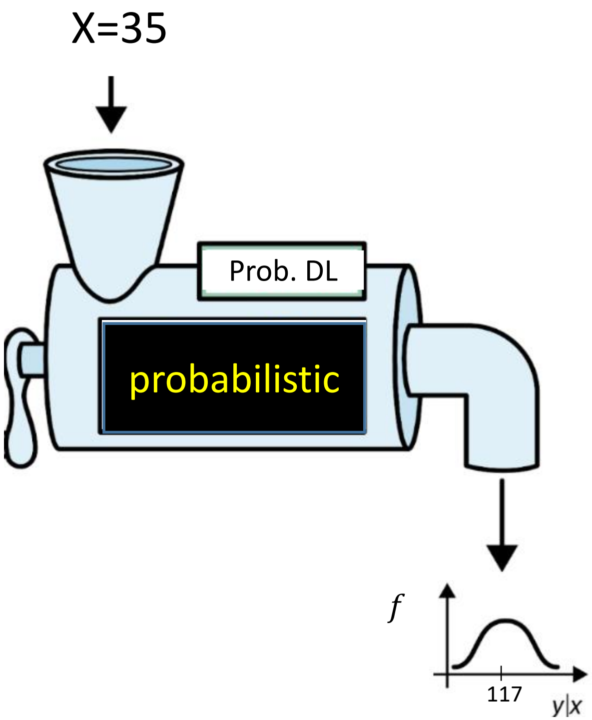
University of Zurich[UZH]

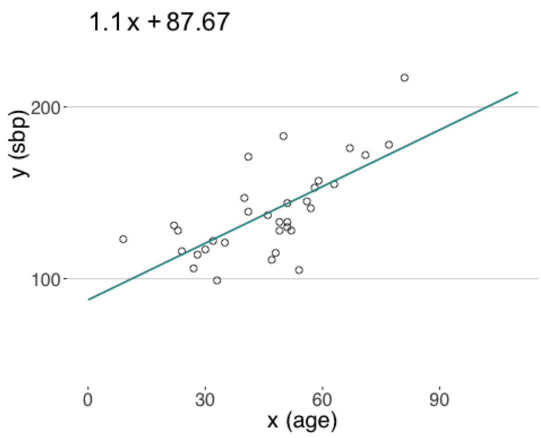
Beate Sick

Zurich University of Applied Sciences
zhaw School of Engineering
IDP Institute of Data Analysis and Process Design

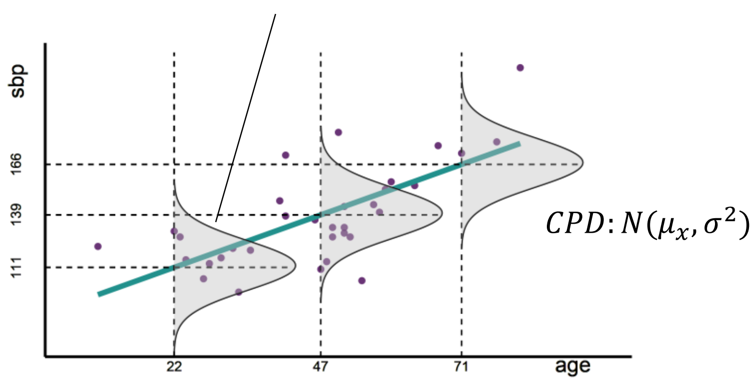
Non-probabilistic versus probabilistic regression DL models







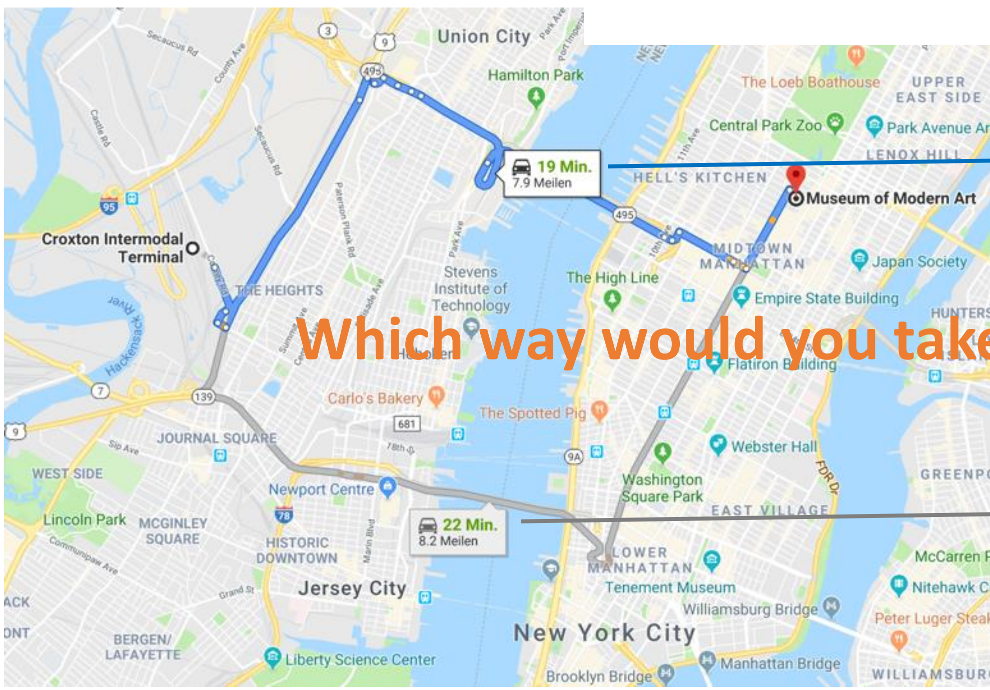
CPD: **C**onditional **P**robability **D**istribution



How can we benefit from a probabilistic model?

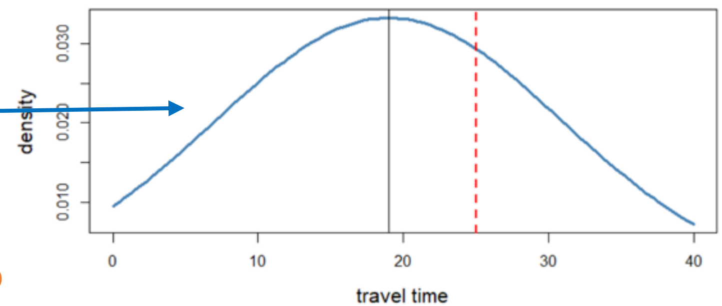
You'll get 500$ tip if I arrive at MOMA within 25 minutes!

Let's use my probabilistic travel time gadget!

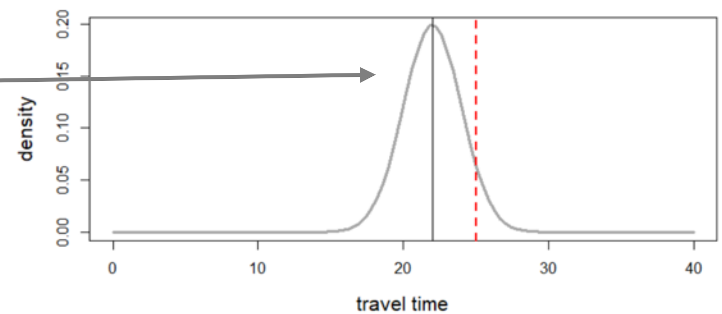


Which way would you take?

Chance to get tip: 69%



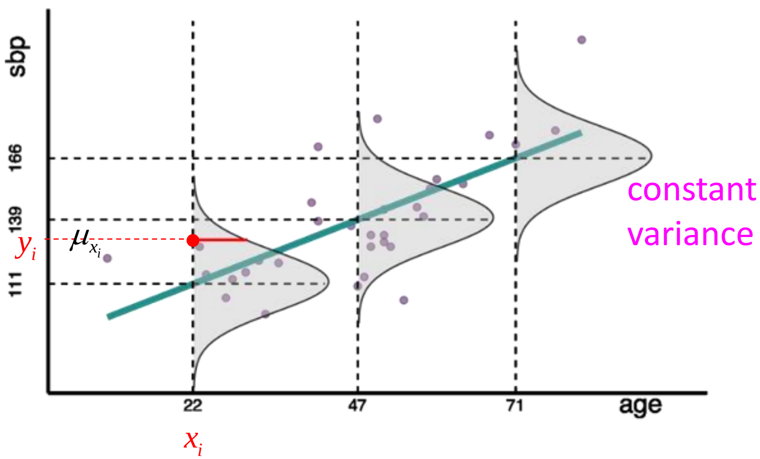
Chance to get tip: 93%

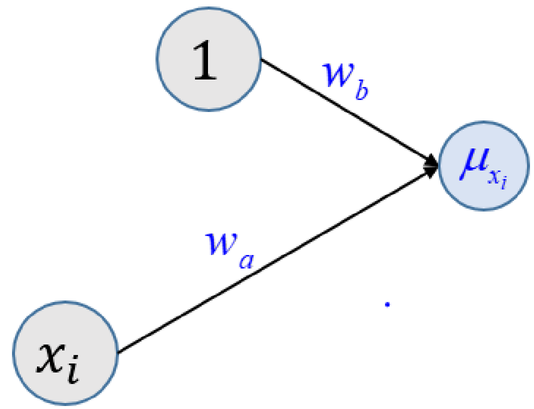


How to train a NN to output the parameter of a CPD?

→ use the beautiful maximum likelihood principle

$$Y_{X_i} \sim \mathrm{N}(\mu_{x_i}, \sigma^2)$$





Maximum likelihood:

$$\boldsymbol{w}_{\mathrm{ML}} = \underset{w}{\operatorname{argmax}} \prod_i \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{(y_i - \mu(\boldsymbol{w}))^2}{2\sigma(w)^2}}$$

$$= \underset{w}{\operatorname{argmin}} \underbrace{\sum_i - \left(\log\left(\cancel{\frac{1}{\sqrt{2\pi\sigma}}}\right) + \frac{(y_i - \mu_i(\boldsymbol{w}))^2}{\cancel{2\sigma^2}} \right)}_{\text{Negative Log-Likelihood (NLL)}}$$

$$(\hat{w}_a, \hat{w}_b)_{\mathrm{ML}} = \underset{w_a, w_b}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - (w_a \cdot x_i + w_b) \right)^2$$

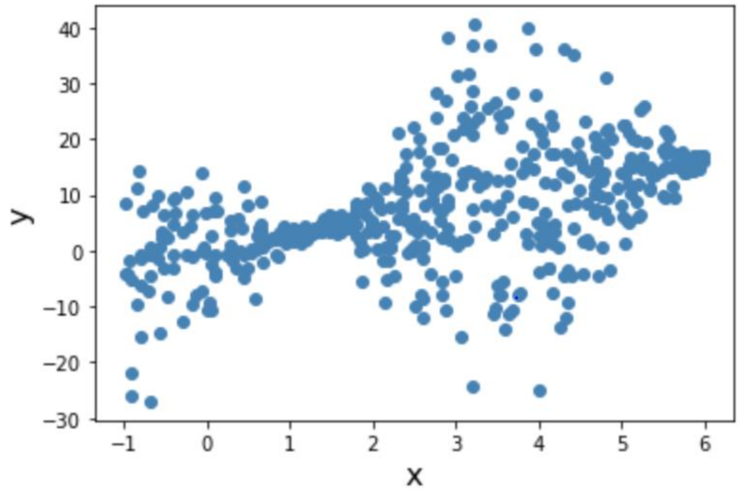
gradient descent with MSE loss

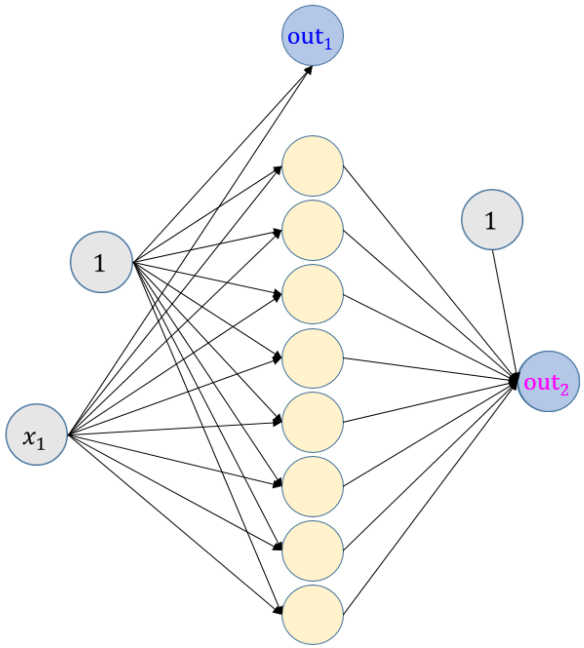
$\hat{w}_a$ $\hat{w}_b$

Minimizing NLL loss $\xRightarrow{\sigma\ const}$ Minimizing MSE loss

Fit a probabilistic regression with flexible non-constant variance

$$Y_{X_i} = (Y|X_i) \sim N(\mu_{x_i}, \sigma_x^2)$$





$$\mu_{x_i} = \text{out}_{1_i}$$

$$\sigma_{x_i} = e^{\text{out}_{2_i}}$$

Minimize the mean negative log-likelihood (NLL) on train data:

$$\text{NLL}(w) = \sum_i -\log\left(f_{pred,w}(y_i|x_i)\right)$$

$$\text{w}_{\text{ML}} = \underset{w}{\text{argmin}} -\sum_i \left(\log\left(\frac{1}{\sqrt{2\pi\sigma_i(\boldsymbol{w})}} \right) + \frac{(y_i - \mu_i(\boldsymbol{w}))^2}{2\sigma_i(\boldsymbol{w})^2} \right)$$

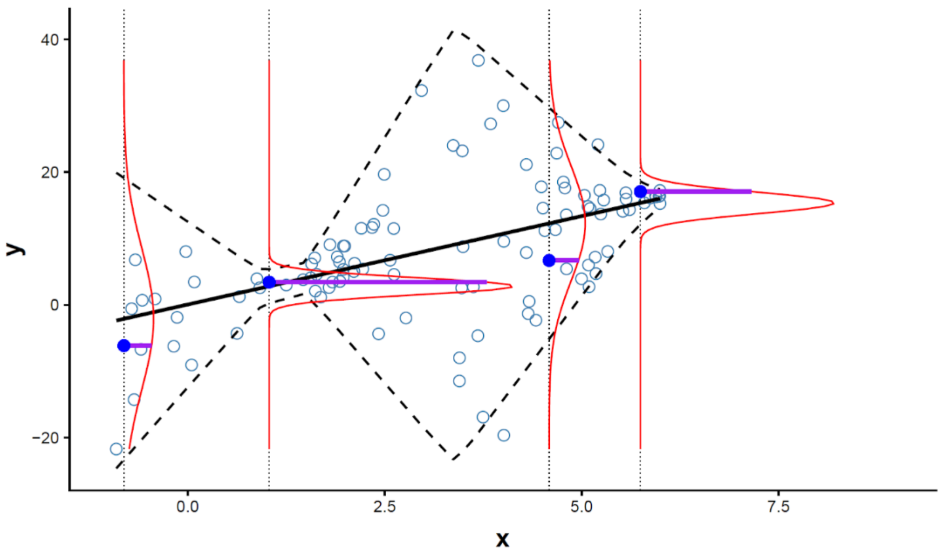
gradient descent with NLL loss

$$\hat{w}_1, \ \hat{w}_{.2}, \ \ldots, \ \hat{w}_{.27}$$

Note: we do not need to know the "ground truth for s" – the likelihood does the job!

Use the NLL on test data to assess the prediction performance

$$\text{NLL} = \sum_{test-data} -\log\big(f_{pred}(y_i|x_i)\big)$$



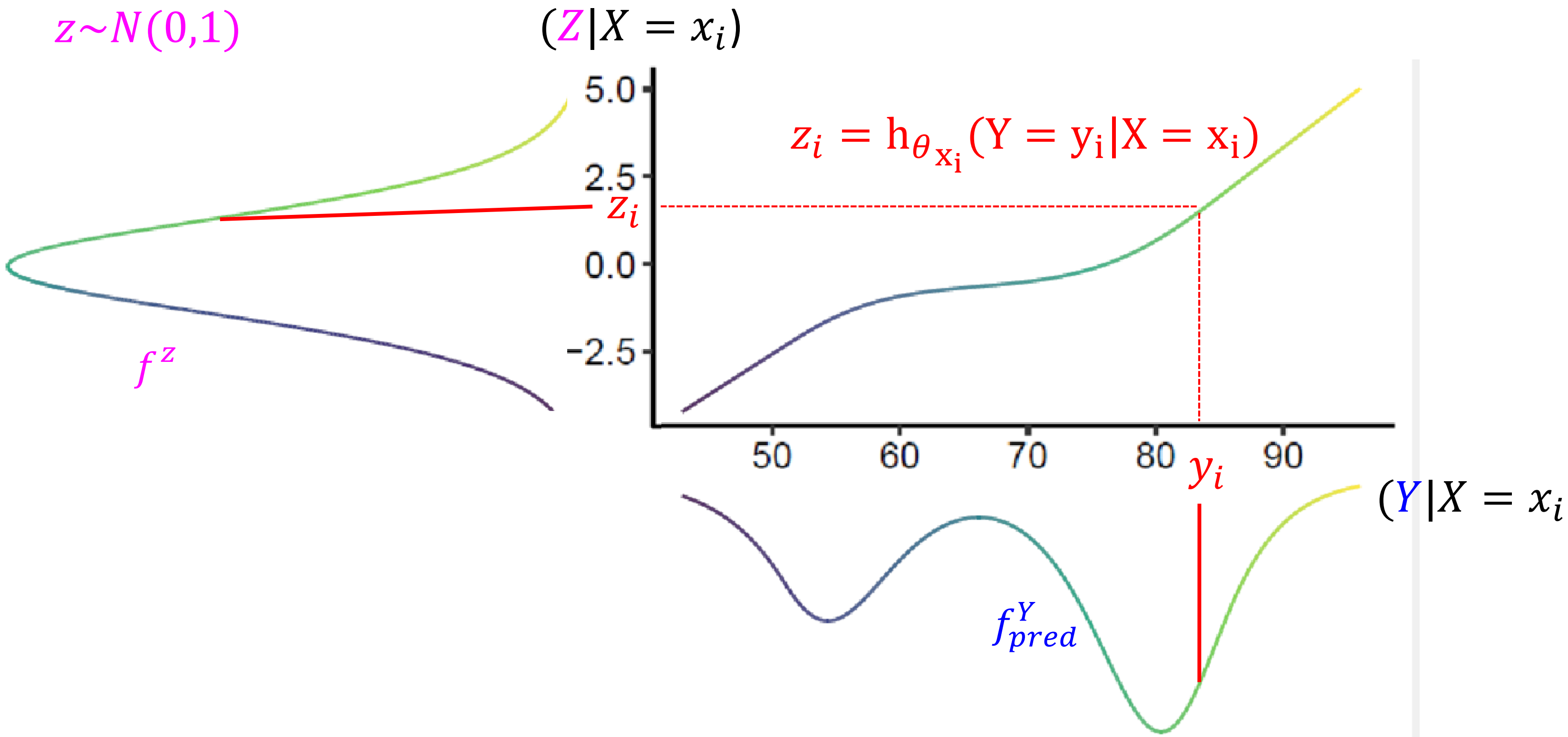
Tensorchief's youtu.be channel

Side remark: The NLL is "strictly proper":
The NLL is then and only then minimal, when the predicted CPD matches the data generating CPD.

What to do if we do not know the family of the conditional outcome distribution?

- Model CPD as mixture (e.g. Gaussians)
- **Model CPD via a transformation model!**

We get the likelihood after transformation to a known distribution

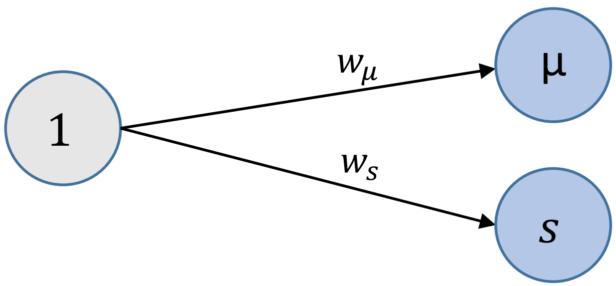


$$\text{NLL} = \sum_i -\log\left(f^y_{pred}\,(y_i|x_i)\right) = \sum_i -\log\left(f^z\,(z_i)\cdot\left|\frac{\partial h_{\theta_{x_i}}}{\partial y}\right|\Big|_{y_i}\right)$$

“change of variable” formula

Going back again: Gauss fit as usual

```
##############K
# sample 5 values from N(4,2)
set.seed(0815)
y_obs = rnorm(5, mean=4, sd=2)
# 6.66  2.53  5.94  4.33  1.50
```

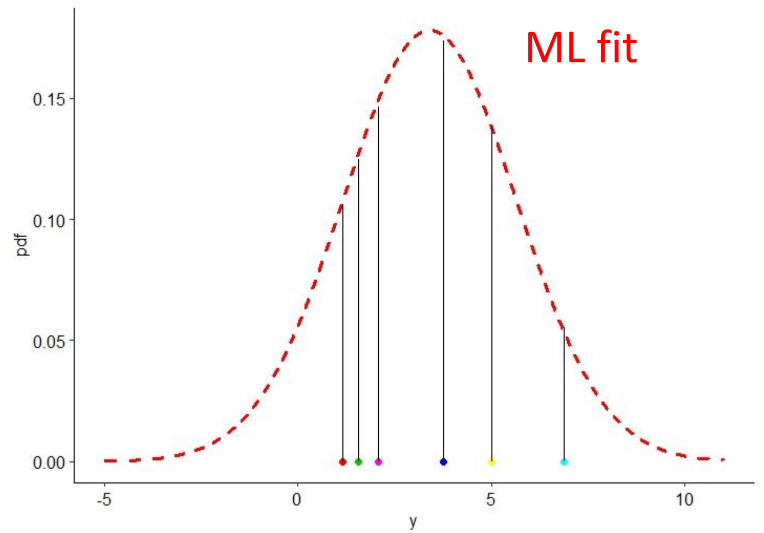


$\sigma = e^s$

```
# minimize NLL with known CPD family
```

$$\mathrm{NLL}(w) = \sum_i -\log\left(f_{pred}^{y}\,(y_i|x_i)\right) = -\sum_i \log\left(\frac{1}{\sqrt{2\pi\sigma}}\,e^{\frac{(y_i-\mu_y(w))^2}{2\sigma_y(w)^2}}\right)$$

```
# optimized parameters:
mu_ml = mean(y_obs)   # 3.1
sd_ml = sd(y_obs)     # 2.3

# optimal NLL
NLL=-sum(log(dnorm(y_obs, mean=3.1, sd=2.3)))
NLL  # 11
```



Gauss fit via transformation approach

Task: find transformation $h: y \rightarrow z = h(y)$ so that $z \sim N(0,1)$

Easy! We know that $y \sim N(\mu, \sigma)$, therefor we look for a linear transformation $z = a \cdot y + b$

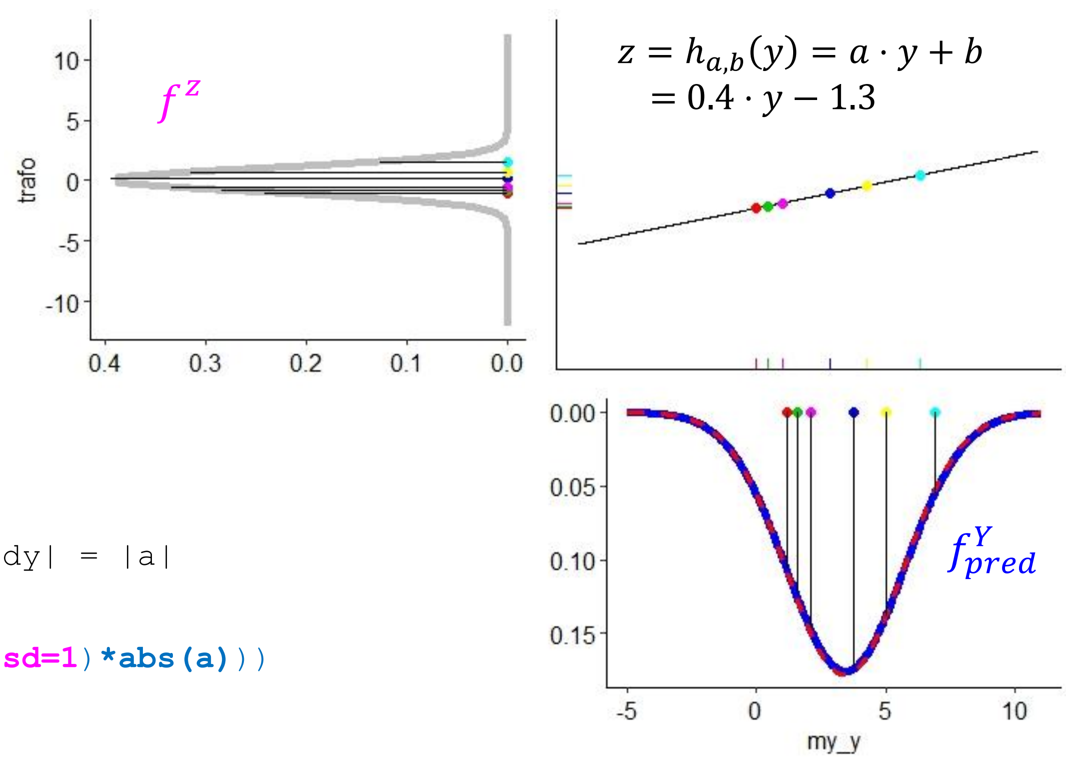
Let's cheat: Use the good old z-transformation and plug in the ML estimates for μ (3.1), σ(2.3):

$$z = \frac{y-\mu}{\sigma} = \frac{1}{\sigma} \cdot y - \frac{\mu}{\sigma}$$

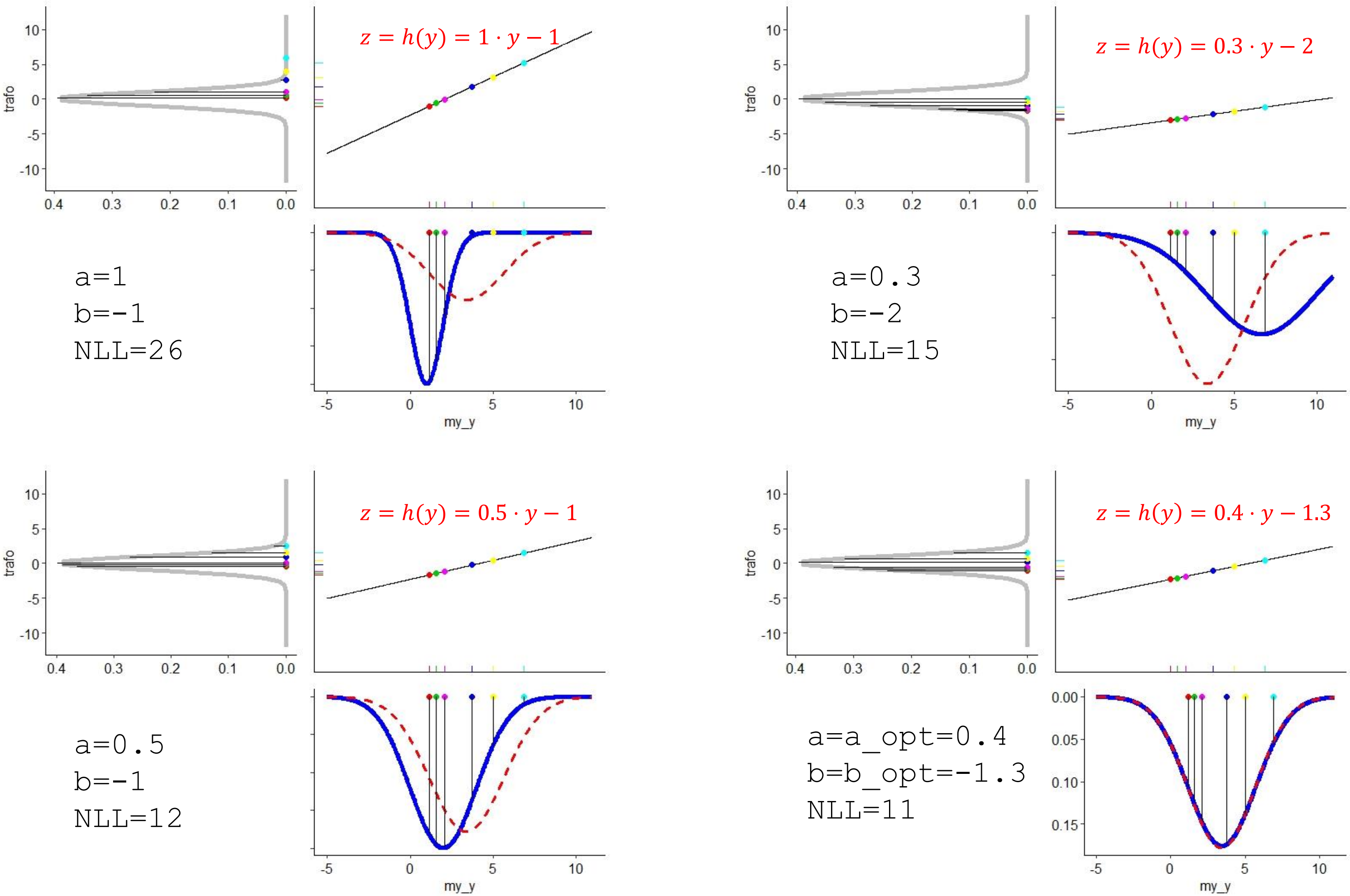
```
# optimal slope and intercept:
a_opt = 1/sd_ml          # 0.4
b_opt = -mu_ml/sd_ml     # -1.3

z = (y_obs - mu_ml)/sd_ml
```

NLL= $\sum_i -\log\left(f^{y}_{pred}\,(y_i)\right) = \sum_i -\log\left(f^{z}\,(z_i) \cdot |\frac{\partial h_{(a,b)}}{\partial y}||_y\right)$

```
# z = h(y) = a_opt*y + b_opt ,  |dh/dy| = |a|

NLL_z <- function(a, b, y_obs) {
  -sum(log(dnorm(a*y_obs+b, mean=0, sd=1)*abs(a)))
}

NLL_z(a_opt, b_opt, y_obs) # 11
```

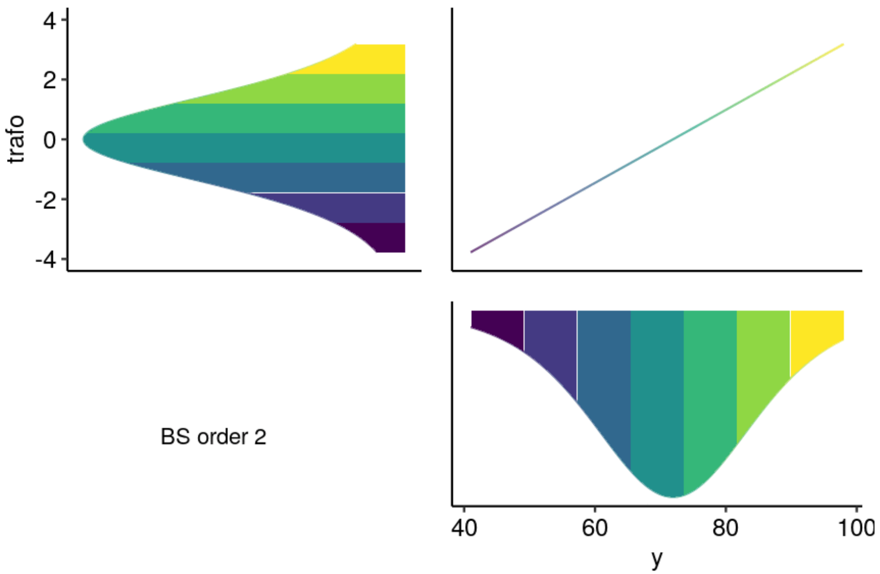


Optimizing the parameters of the transformation via minimizing NLL



Task: find parameter values for *a* and *b*, that minimize $\mathrm{NLL} = \sum_i -\log\left(f^y_{pred}\,(y_i)\right) = \sum_i -\log\left(f^z\,(z_i) \cdot |\frac{\partial h_{(a,b)}}{\partial y}||_y\right)$ → SGD

For non-Gaussian CPDs we need a non-linear transformation



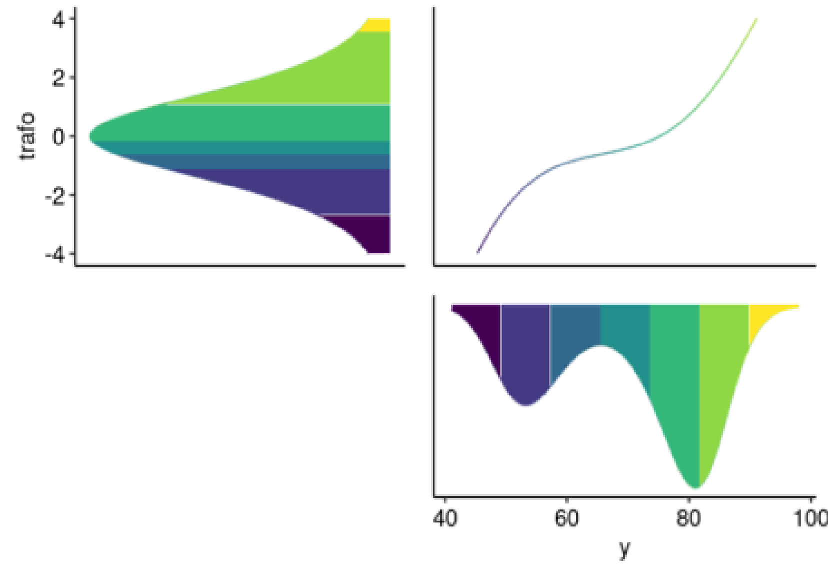
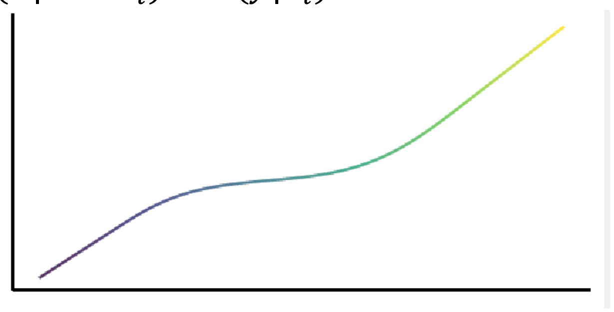


Image credits: Lucas Kook

Using Bernsteinpolynomials to approximate the transformation h

$$z_x = h_{\theta_x}^{MLT}(\tilde{y}|X = x) = \sum_{k=1}^{M} \frac{\vartheta_k(x)}{M+1} Be_k(\tilde{y})$$

$\tilde{y} \in [0,1]$



Bernstein polynomials have nice properties:

- They can approximate each function
- The order M controls the flexibility
- Its bijective, i.e. monotone increasing, if parameters $\vartheta_1 \leq \vartheta_2 \leq \cdots \leq \vartheta_M$

Most Likely Transfromation (MLT) 2017 by T.Hothorn, L.Möst, P.Bühlmann https://onlinelibrary.wiley.com/doi/full/10.1111/sjos.12291

A non-Gaussian CPD requires a flexible transformation function h

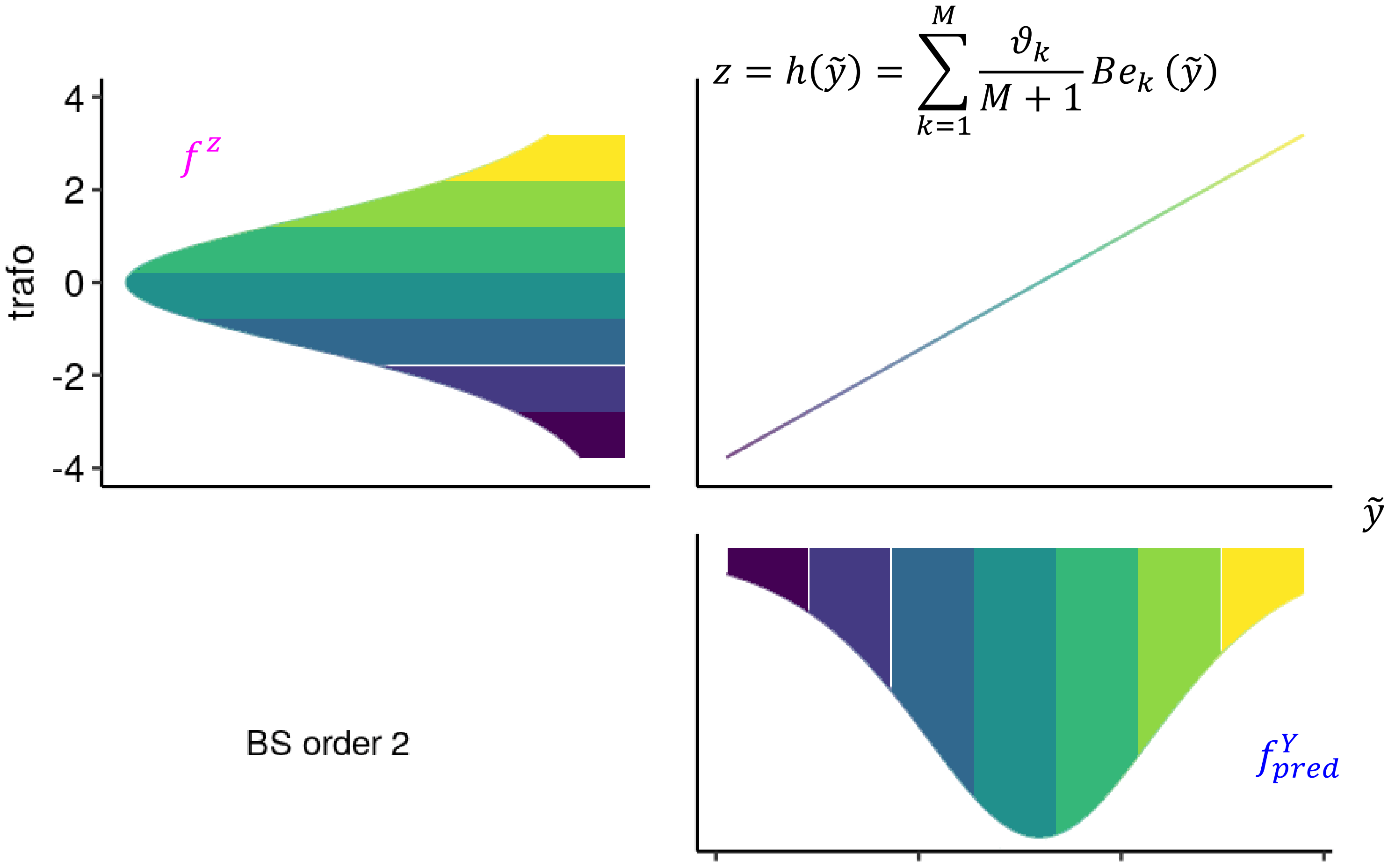
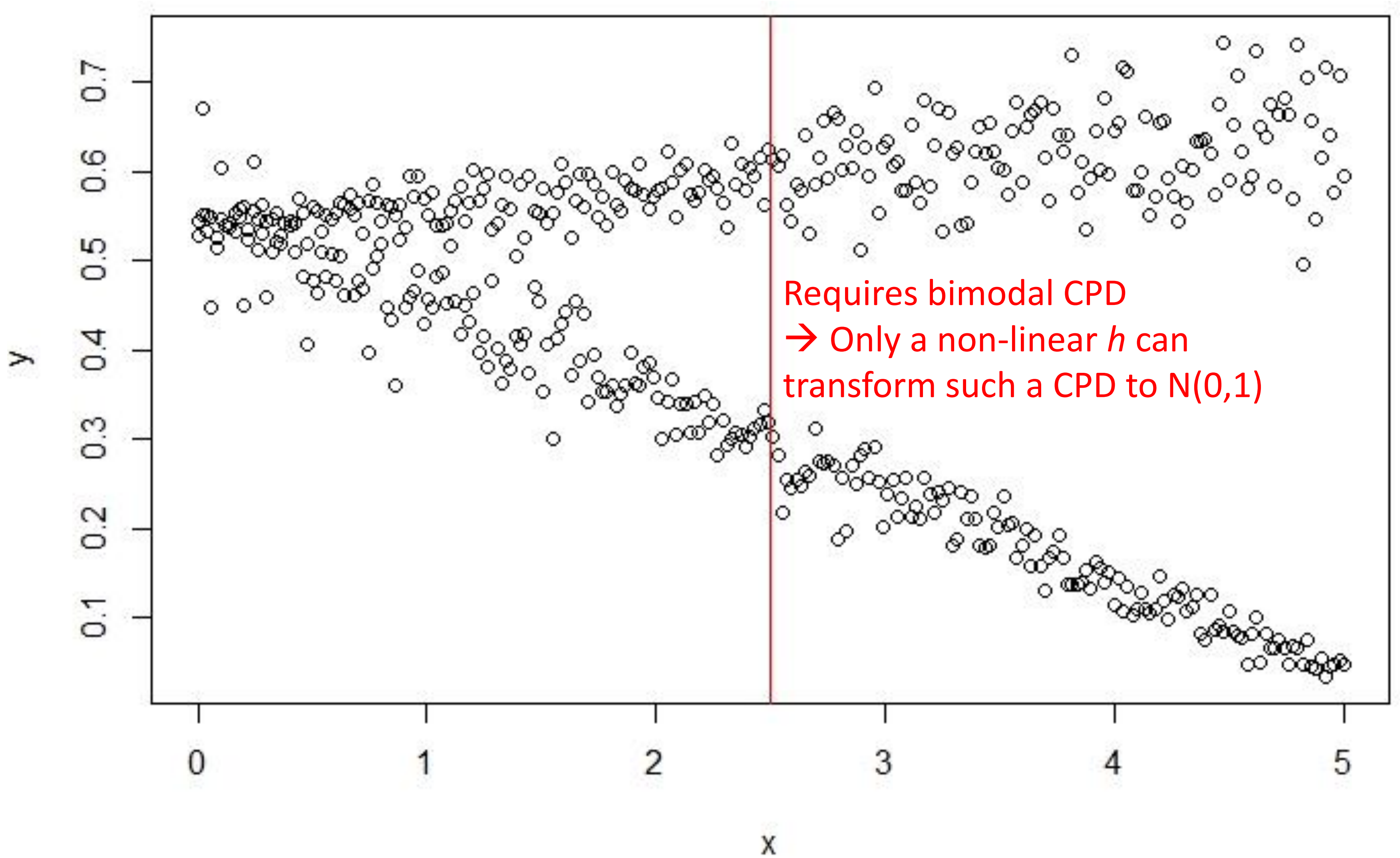


Image credits: Lucas Kook

Have a look on more complex conditional probability distributions



Requires bimodal CPD
→ Only a non-linear *h* can transform such a CPD to N(0,1)

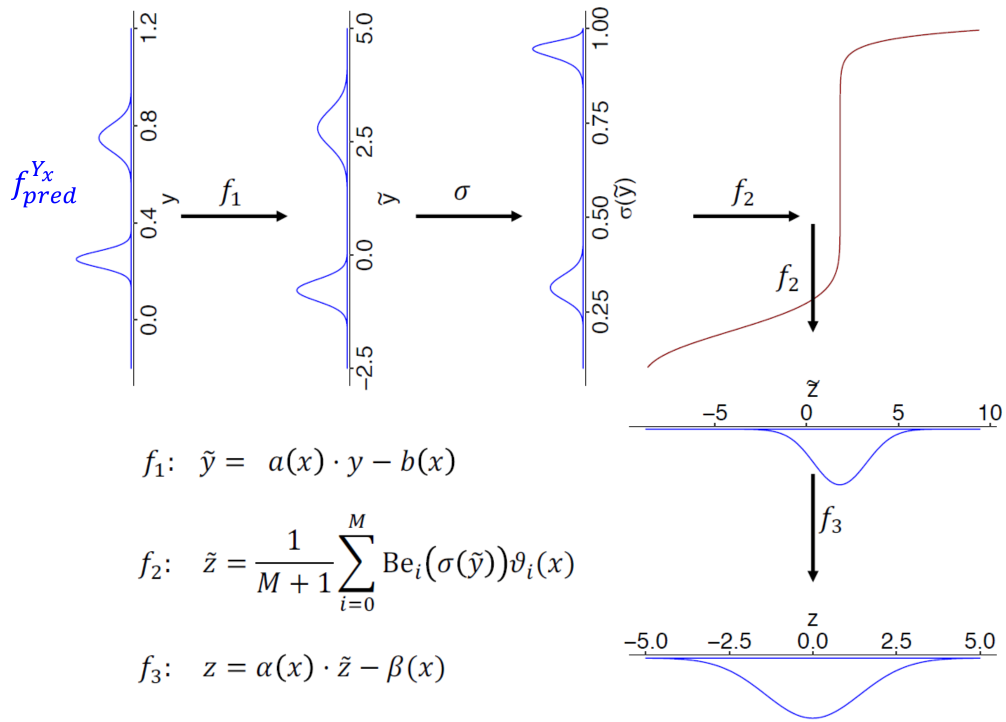
Our deep transformation model

$$z_x = h^{DT}_{\theta(x)}(y|X=x) = \alpha(x)\cdot\left(\sum_{k=1}^{M}\frac{\vartheta_k(x)}{M+1}Be_k\left(\sigma\left(a(x)\cdot y - b(x)\right)\right)\right) - \beta(x)$$

$$z_x = h^{DT}_{\theta(x)}(y|X=x) = f_{3,\alpha_x,\beta_x}\circ f_{2,\vartheta_{0_x},\dots,\vartheta_{M_x}}\circ\sigma\circ f_{1,a_x,b_x}(y|X=x)$$

Get parameter $\theta(x)$ by minimize the NLL

$$NLL = \sum_{train-data} -log\left(f^{z}(z_x)\cdot|h'_{\theta(x)}(y)|\right)$$



f_1: $\tilde{y} = a(x)\cdot y - b(x)$

f_2: $\tilde{z} = \frac{1}{M+1}\sum_{i=0}^{M}\mathrm{Be}_i(\sigma(\tilde{y}))\vartheta_i(x)$

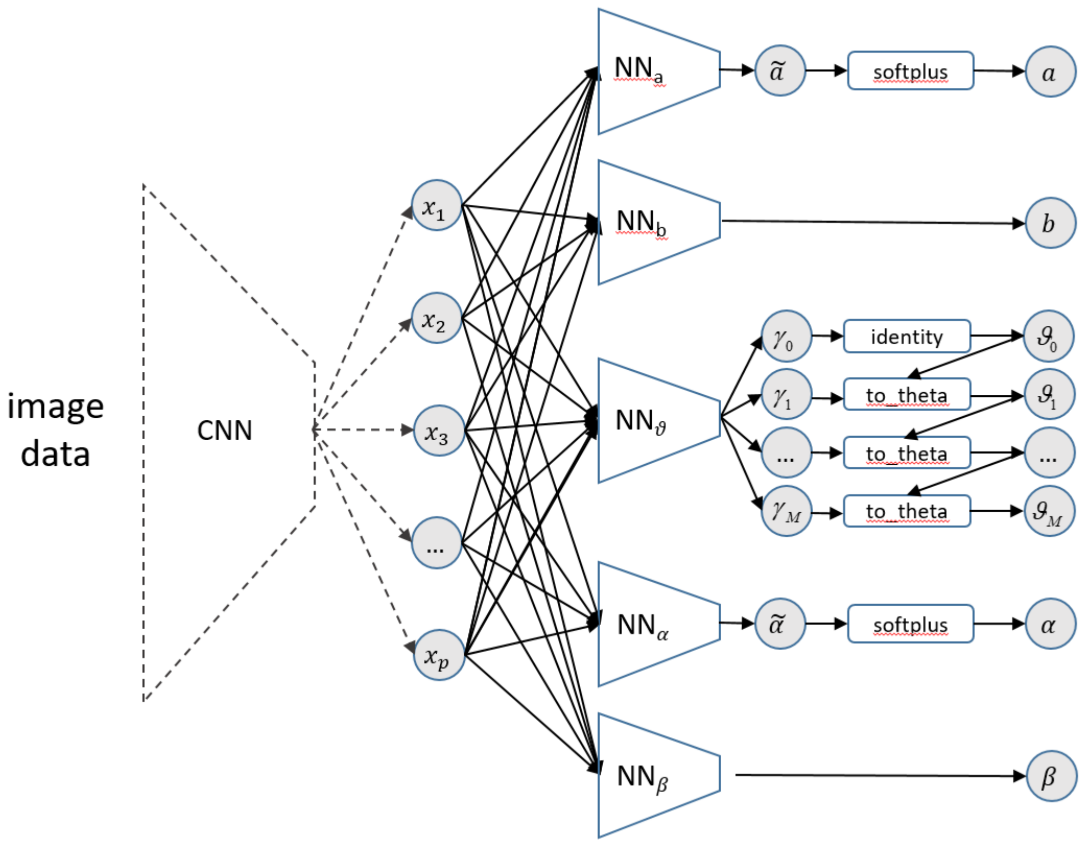
f_3: $z = \alpha(x)\cdot\tilde{z} - \beta(x)$

$f^{z} = f^{N(0,1)}$

Deep transformation models 2020 by B.Sick, T.Hothorn, O.Dürr https://arxiv.org/abs/2004.00464

Architecture of our Deep transformation model

$$z_x = h^{DT}_{\theta(x)}(y|X = x) = \alpha(x) \cdot \left(\sum_{k=1}^{M} \frac{\vartheta_k(x)}{M+1} Be_k\left(\sigma\left(a(x) \cdot y - b(x)\right)\right)\right) - \beta(x)$$

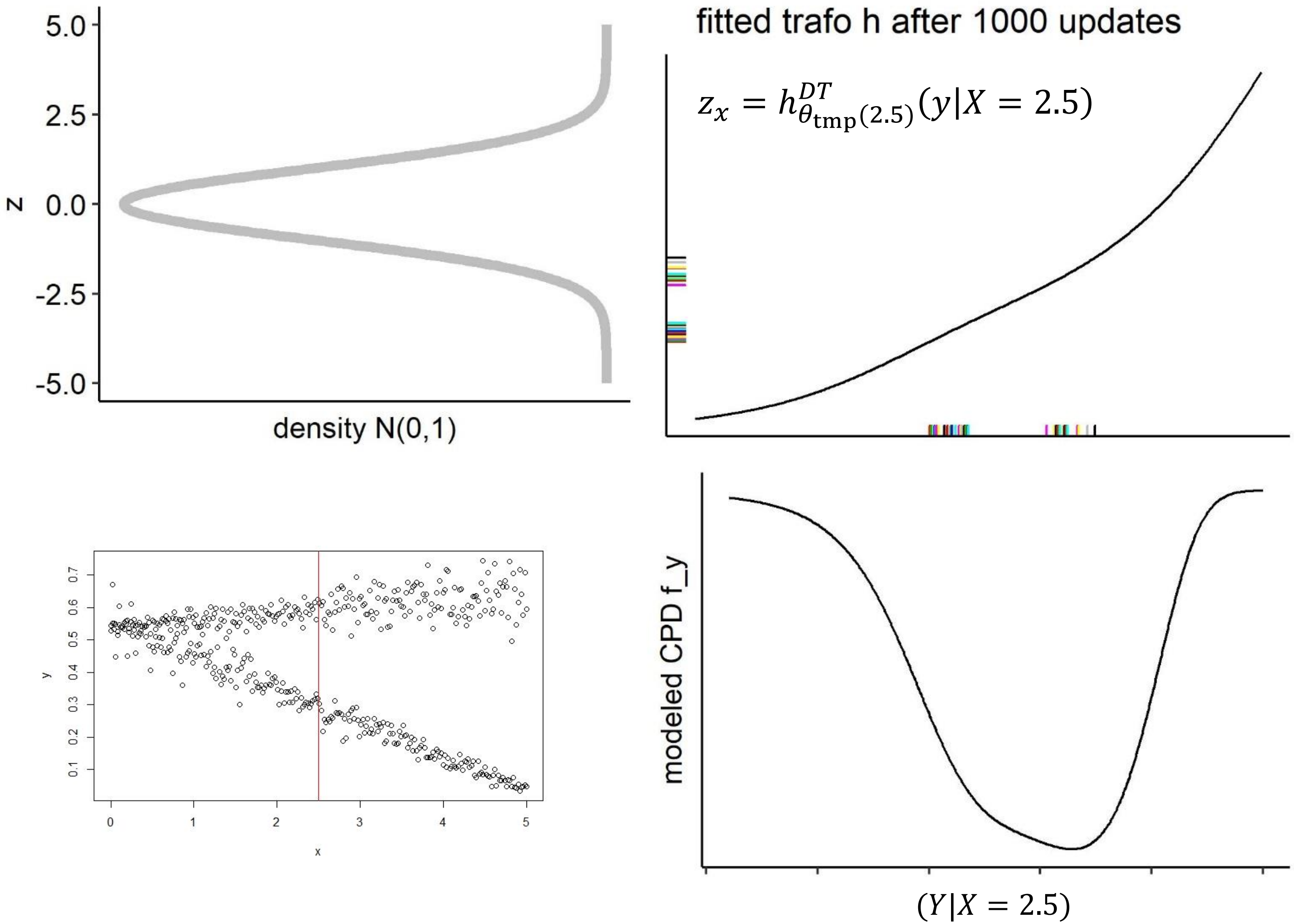


$\vartheta_0 = \gamma_0$
$\vartheta_1 = \vartheta_0 + e^{\gamma_1}$
...
$\vartheta_M = \vartheta_{M-1} + e^{\gamma_M}$
$\rightarrow \vartheta_0 < \vartheta_1 ... < \vartheta_M$

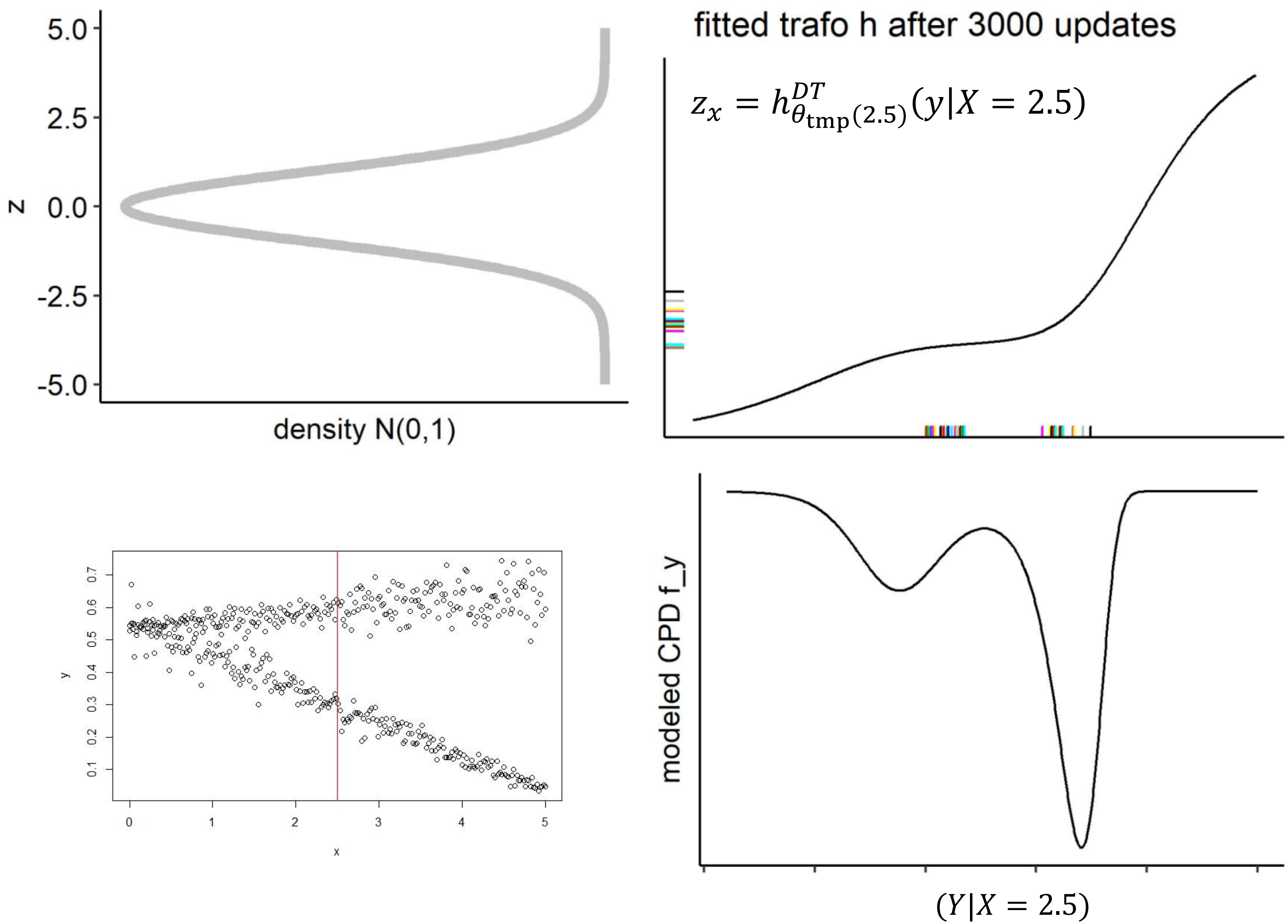
We tune the weights via SGD to minimize $\mathrm{NLL} = \sum_i -\log\left(f^z\left(z_{x_i}\right) \cdot |h'_{\theta(x_i)}(y_i|x_i)|\right)$ yielding the parameters θ of h.

Deep transformation models 2020 by B.Sick, T.Hothorn, O.Dürr https://arxiv.org/abs/2004.00464

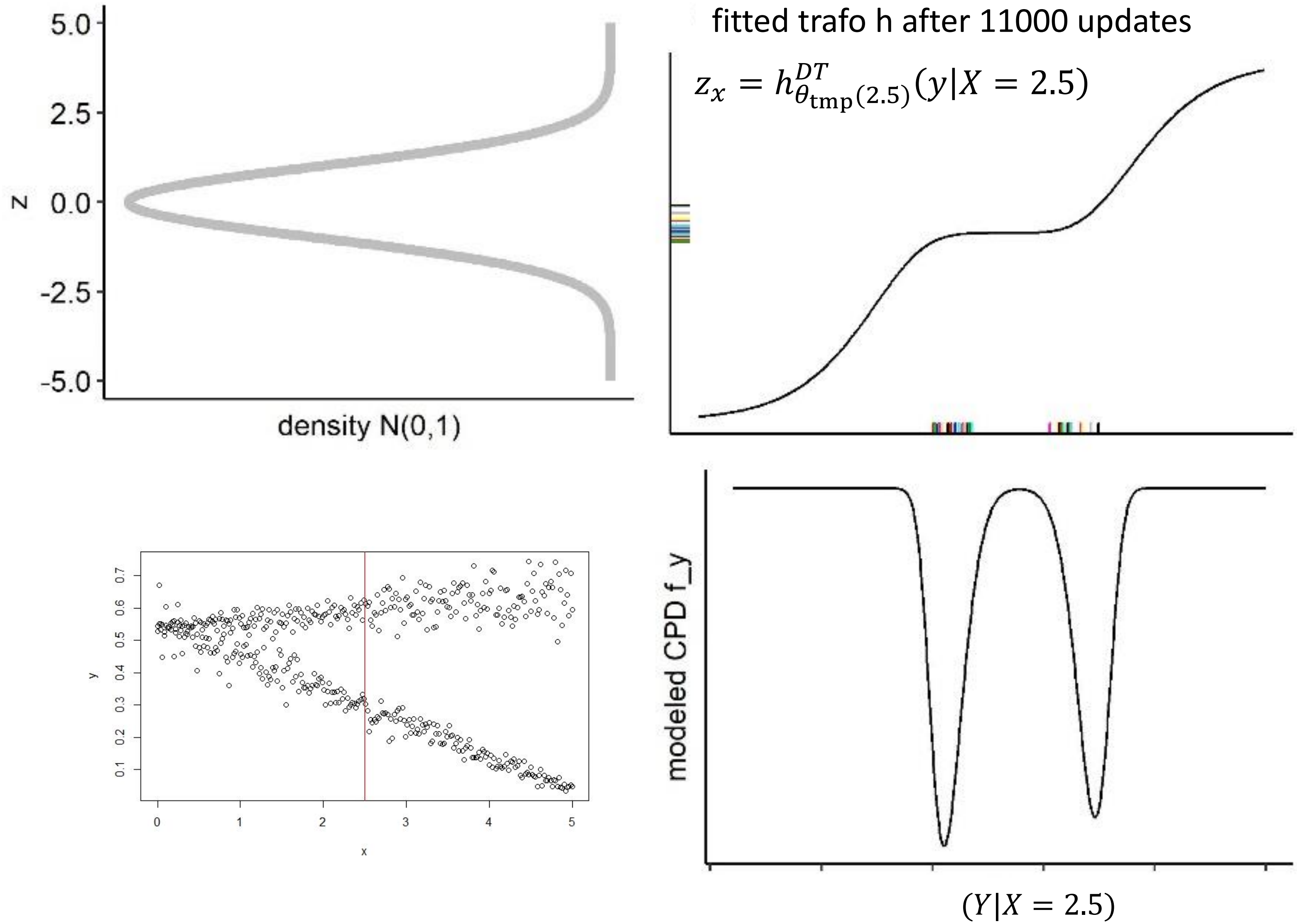
Learning the parameters of the deep transformation model via SGD



Learning the parameters of the deep transformation model via SGD

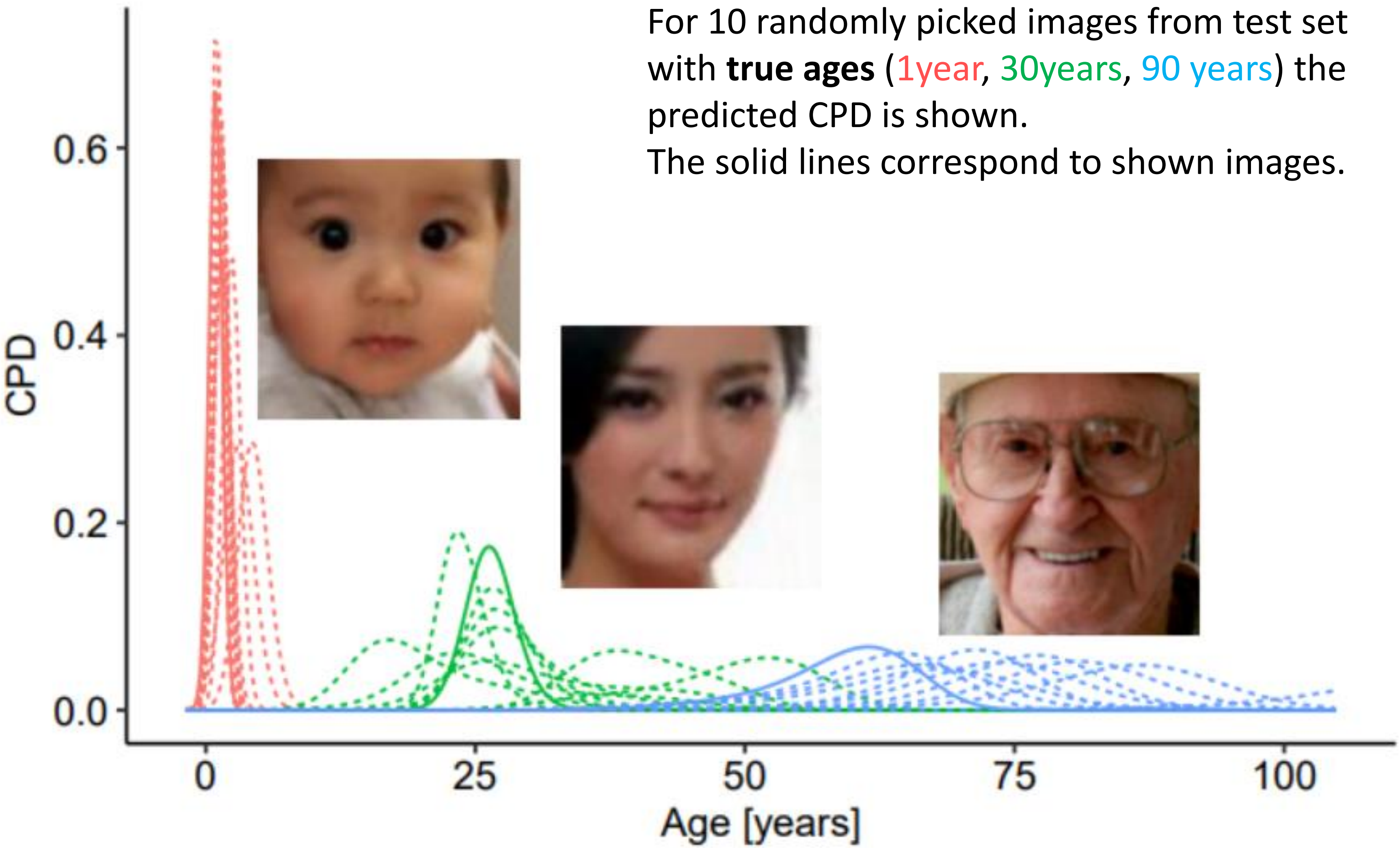


Learning the parameters of the deep transformation model via SGD



Application: Predict CPD for age based on an image

For 10 randomly picked images from test set with **true ages** (1year, 30years, 90 years) the predicted CPD is shown.
The solid lines correspond to shown images.

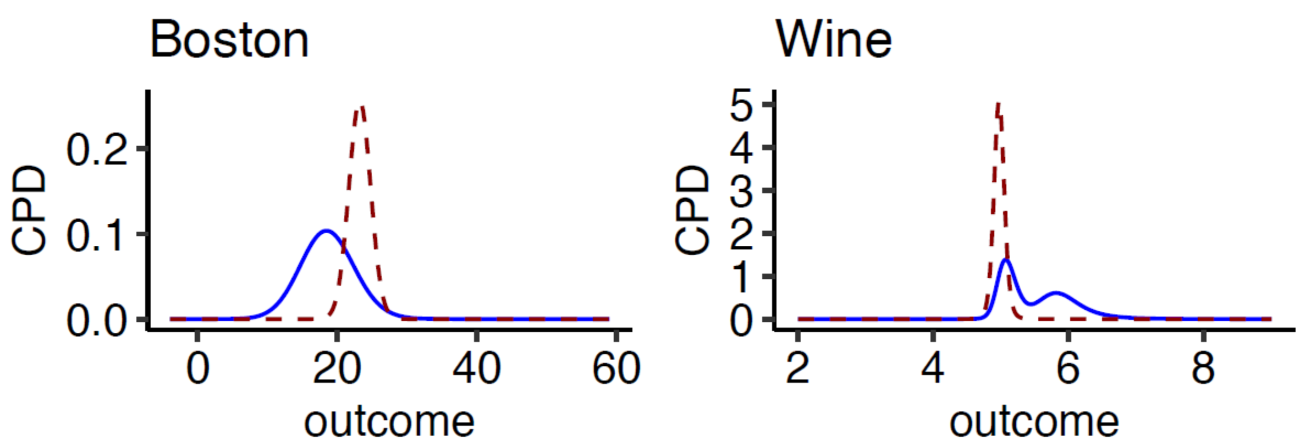


Deep transformation models 2020 by B.Sick, T.Hothorn, O.Dürr https://arxiv.org/abs/2004.00464

Application: Benchmarking our model

TABLE I
COMPARISON OF PREDICTION PERFORMANCE (TEST NLL, SMALLER IS BETTER) ON REGRESSION BENCHMARK UCI DATASETS. THE BEST METHOD FOR EACH DATASET IS BOLDED, AS ARE THOSE WITH STANDARD ERRORS THAT OVERLAP WITH THE STANDARD ERRORS OF THE BEST METHOD.

Data Set	N	DL_MLT	NGBoost	MC Dropout	Deep Ensembles	Gaussian Process	MDN	NFN
Boston	506	**2.42 ± 0.050**	**2.43 ± 0.15**	**2.46 ±0.25**	**2.41 ±0.25**	**2.37 ±0.24**	**2.49 ± 0.11**	**2.48 ±0.11**
Concrete	1030	3.29 ± 0.02	**3.04 ± 0.17**	**3.04 ±0.09**	**3.06 ±0.18**	**3.03 ±0.11**	**3.09 ± 0.08**	**3.03 ±0.13**
Energy	768	**1.06 ± 0.09**	**0.60 ± 0.45**	1.99 ±0.09	1.38 ±0.22	**0.66 ±0.17**	**1.04 ± 0.09**	1.21 ±0.08
Kin8nm	8192	-0.99 ± 0.01	-0.49 ± 0.02	-0.95 ±0.03	**-1.20 ±0.02**	-1.11 ±0.03	NA	NA
Naval	11934	**-6.54 ± 0.03**	-5.34± 0.04	-3.80 ±0.05	-5.63 ±0.05	-4.98 ±0.02	NA	NA
Power	9568	**2.85 ± 0.005**	**2.79 ± 0.11**	**2.80 ±0.05**	**2.79 ±0.04**	**2.81 ±0.05**	NA	NA
Protein	45730	**2.63 ± 0.006**	2.81 ± 0.03	2.89 ±0.01	2.83 ±0.02	2.89 ±0.02	NA	NA
Wine	1588	**0.67 ± 0.028**	0.91 ± 0.06	0.93 ±0.06	0.94 ±0.12	0.95 ±0.06	NA	NA
Yacht	308	**0.004 ± 0.046**	**0.20 ± 0.26**	1.55 ±0.12	1.18 ±0.21	0.10 ±0.26	NA	NA



The 2 CPDs (dashed and solid line) correspond to 2 picked observations in the respective data set.

Deep transformation models 2020 by B.Sick, T.Hothorn, O.Dürr https://arxiv.org/abs/2004.00464

I want to thank my colleagues

Project Collaborators:

- Prof. Dr. Oliver Dürr (HTWG)
- Prof. Dr. Torsten Hothorn (UZH)

Thanks for fruitful discussions to:

- Lucas Kook (UZH & ZHAW)
- Lisa Herzog (UZH & ZHAW)
- Elvis Murina (ex ZHAW)
- Matthias Hermann (HTWG)

Thank you for your attention!