# Distributional Regression for Actuarial Applications: Distributional Refinement Network

Insurance Data Science Conference

Eric Dong[1]
Joint Work With: Prof. Benjamin Avanzi[2], Dr. Patrick Laub[1], Prof. Bernard Wong[1]

[1]Risk & Actuarial Studies, UNSW Sydney
[2]Actuarial Studies, University of Melbourne

London, June 2025

# Topics

# Topics

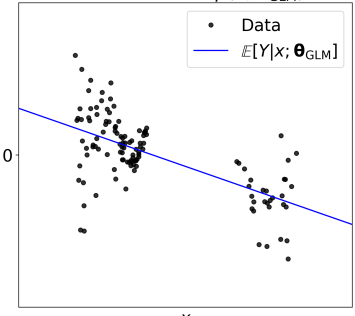
# Actuarial Modelling

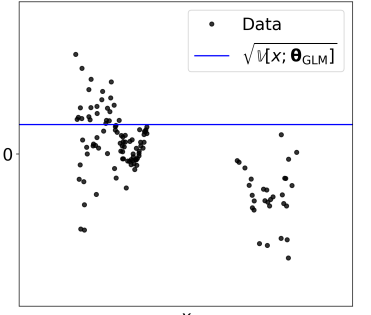
A key task in actuarial modelling is to learn a multivariate function $\mathbf{f}$, or univariate $f$, parameterised by $\boldsymbol{\theta}$, to make "predictions" about insurance loss $Y$ given covariates $\mathbf{X}$, which defines a risk profile.

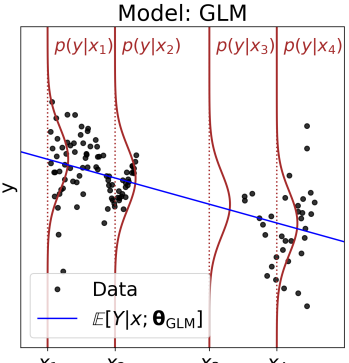
- **Option 1 (Point Estimate):** Assume the loss is exactly equal to the model output, i.e.,
$$f(\mathbf{x};\boldsymbol{\theta}) = \mu(\mathbf{x};\boldsymbol{\theta}) = \mathbb{E}[Y|\mathbf{x};\boldsymbol{\theta}].$$
- **Option 2 (Distributional Regression):** Assume the loss (conditional on $\mathbf{x}$) is drawn from a distribution $\mathcal{D}$, with distributional parameters, e.g., $\mathbb{E}[Y|\mathbf{x};\boldsymbol{\theta}]$, $\sqrt{\mathbb{V}[Y|\mathbf{x};\boldsymbol{\theta}]}, \ldots$, determined by the modelled function:
$$Y|\mathbf{f}(\mathbf{x};\boldsymbol{\theta}) \sim \mathcal{D}(\mathbf{f}(\mathbf{x};\boldsymbol{\theta})).$$

# Model 1: Generalised Linear Models (GLMs)

- Nelder and Wedderburn (1972): $Y|\mathbf{f}(x;\boldsymbol{\theta}_{\text{GLM}}) \sim \mathcal{N}(\mu(x;\boldsymbol{\theta}_{\text{GLM}}), \sigma^2(x;\boldsymbol{\theta}_{\text{GLM}}))$, i.e., $\mathbf{f}(x;\boldsymbol{\theta}_{\text{GLM}}) = [\mu(x;\boldsymbol{\theta}_{\text{GLM}}), \sigma(x;\boldsymbol{\theta}_{\text{GLM}})]^\top$.
  - A simple mean function: $\mu(x;\boldsymbol{\theta}_{\text{GLM}}) = \mathbb{E}[Y|x;\boldsymbol{\theta}_{\text{GLM}}] = \beta_0 + \beta_1 x$.
  - A constant standard deviation function (in the Gaussian case): $\sigma(x;\boldsymbol{\theta}_{\text{GLM}}) = \sqrt{\mathbb{V}[Y|x;\boldsymbol{\theta}_{\text{GLM}}]} = \hat{\sigma}_{\text{GLM}}$.

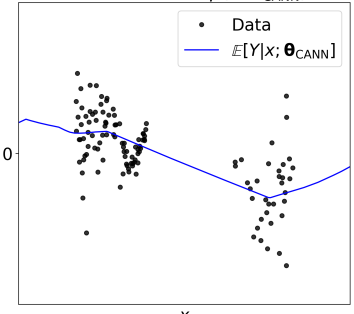


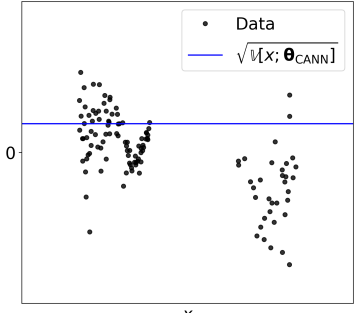



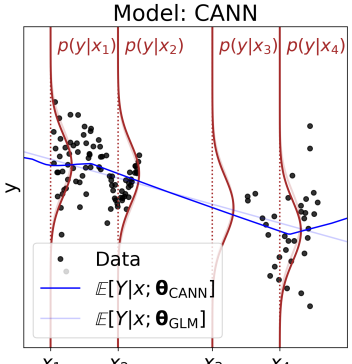



# Model 2: Combined Actuarial Neural Networks (CANNs)

- Schelldorfer and Wüthrich (2019): $Y|\mathbf{f}(x;\boldsymbol{\theta}_{\text{CANN}}) \sim \mathcal{N}(\mu(x;\boldsymbol{\theta}_{\text{CANN}}), \sigma^2(x;\boldsymbol{\theta}_{\text{CANN}}))$, i.e., $\mathbf{f}(x;\boldsymbol{\theta}_{\text{CANN}}) = [\mu(x;\boldsymbol{\theta}_{\text{CANN}}), \sigma(x;\boldsymbol{\theta}_{\text{CANN}})]^\top$.
  - Flexible and correlated mean and standard deviation functions: $\mu(x;\boldsymbol{\theta}_{\text{CANN}}) = \sum_{j=1}^{J} \phi_j(x;\boldsymbol{\theta}_{\text{CANN}}) + \mu(x;\boldsymbol{\theta}_{\text{GLM}}), \sigma(x;\boldsymbol{\theta}_{\text{CANN}}) = \hat{\sigma}_{\text{CANN}}$, where $\phi_j$'s are expressive basis functions learnt by the neural network.

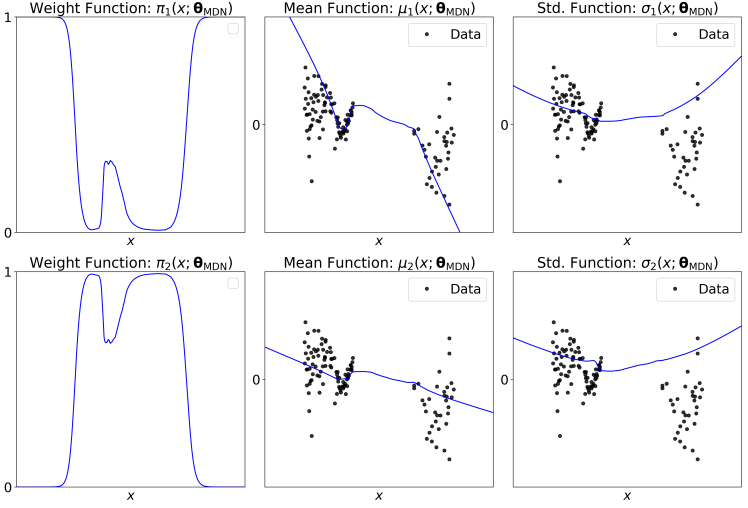


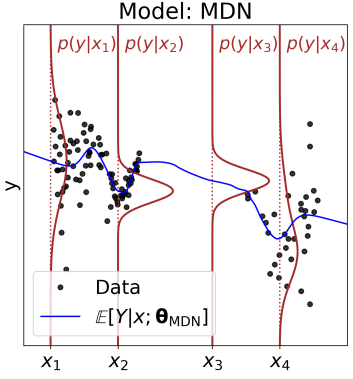



# Model 3: Mixture Density Networks (MDNs)

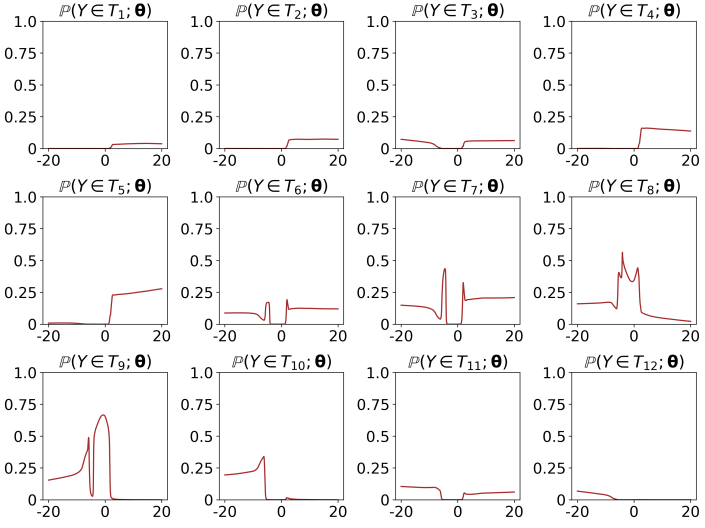
- Bishop (1994): $Y|\mathbf{f}(x;\boldsymbol{\theta}_{\text{MDN}}) \sim \sum_{k=1}^{K} \pi_k(x;\boldsymbol{\theta}_{\text{MDN}}) \cdot \mathcal{N}(\mu_k(x;\boldsymbol{\theta}_{\text{MDN}}), \sigma_k^2(x;\boldsymbol{\theta}_{\text{MDN}}))$ , i.e., $\mathbf{f}(x;\boldsymbol{\theta}_{\text{MDN}}) = [\boldsymbol{\pi}(x;\boldsymbol{\theta}_{\text{MDN}})^\top, \boldsymbol{\mu}(x;\boldsymbol{\theta}_{\text{MDN}})^\top, \boldsymbol{\sigma}(x;\boldsymbol{\theta}_{\text{MDN}})^\top]^\top$.

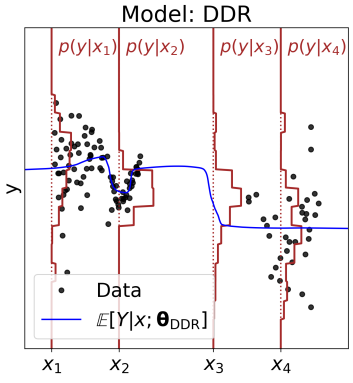


# Model 4: Deep Distribution Regression (DDR)

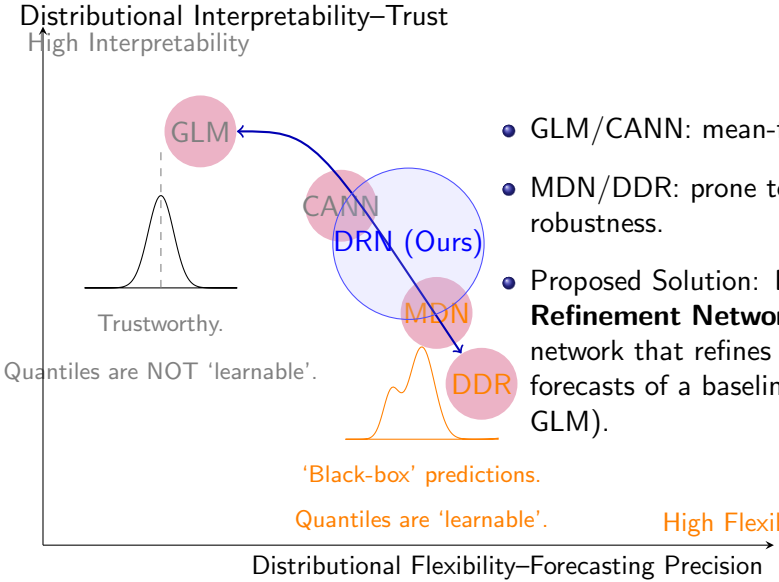
- Li et al. (2021): $Y|\mathbf{f}(x;\boldsymbol{\theta}_{\text{DDR}}) \sim \sum_{k=1}^{K} f_k(x;\boldsymbol{\theta}_{\text{DDR}}) \cdot \mathcal{U}(c_{k-1}, c_k)$, where $\mathbf{f}(x;\boldsymbol{\theta}_{\text{DDR}}) = [\mathbb{P}(Y \in T_1|\mathbf{x};\boldsymbol{\theta}_{\text{DDR}}), \ldots, \mathbb{P}(Y \in T_K|\mathbf{x};\boldsymbol{\theta}_{\text{DDR}})]^\top$, and $T_k = [c_{k-1}, c_k)$.



# Topics

# Why Distributional Refinement Networks (DRNs)?




- GLM/CANN: mean-focused.
- MDN/DDR: prone to overfitting; lacks robustness.
- Proposed Solution: **Distributional Refinement Network (DRN)**: a neural network that refines the distribution forecasts of a baseline model (e.g., a GLM).

# Distributional Refinement Networks (DRNs)

- DRNs can **incorporate a baseline** model and are initialised to align closely with its distributional behaviour.
- DRNs **adopt a novel training loss** function:

$$\mathbb{E}_{(\mathbf{X},Y)}[\mathcal{L}(Y, \mathbf{f}(\mathbf{X}; \boldsymbol{\theta}_{\text{DRN}})] = \mathbb{E}_{(\mathbf{X},Y)}\Big[\underbrace{-\log(p(Y|\mathbf{f}(\mathbf{X}; \boldsymbol{\theta}_{\text{DRN}})))}_{\text{Data Loss}} \tag{1}$$

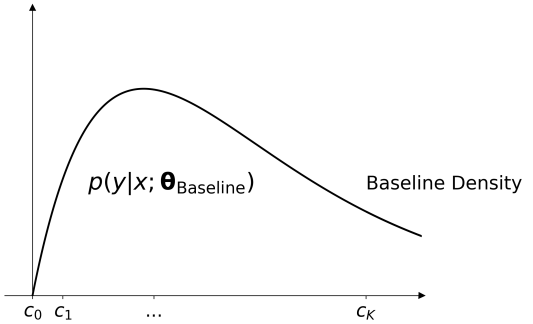
$$+ \alpha_1 \cdot \underbrace{d\left(p_{Y|\mathbf{f}(\mathbf{X};\boldsymbol{\theta}_{\text{DRN}})}, p_{Y|\mathbf{f}(\mathbf{X};\boldsymbol{\theta}_{\text{Baseline}})}\right)}_{\text{Distributional Resemblence}} \tag{2}$$

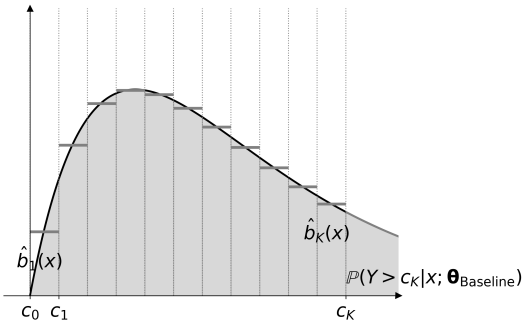
$$+ \alpha_2 \cdot \underbrace{\text{second\_order\_diff}\left(p_{Y|\mathbf{f}(\mathbf{X};\boldsymbol{\theta}_{\text{DRN}})}\right)^2}_{\text{Density Smoothness Loss}} \tag{3}$$

$$+ \alpha_3 \cdot \underbrace{\left(\mathbb{E}[Y|\mathbf{X}; \boldsymbol{\theta}_{\text{DRN}}] - \mathbb{E}[Y|\mathbf{X}; \boldsymbol{\theta}_{\text{Baseline}}]\right)^2}_{\text{Mean Resemblence Loss}}\Big] \tag{4}$$

where $d(\cdot, \cdot)$ is a chosen divergence measure that quantifies the statistical distance between two probability distributions, e.g., KL divergence.

# Baseline Integration

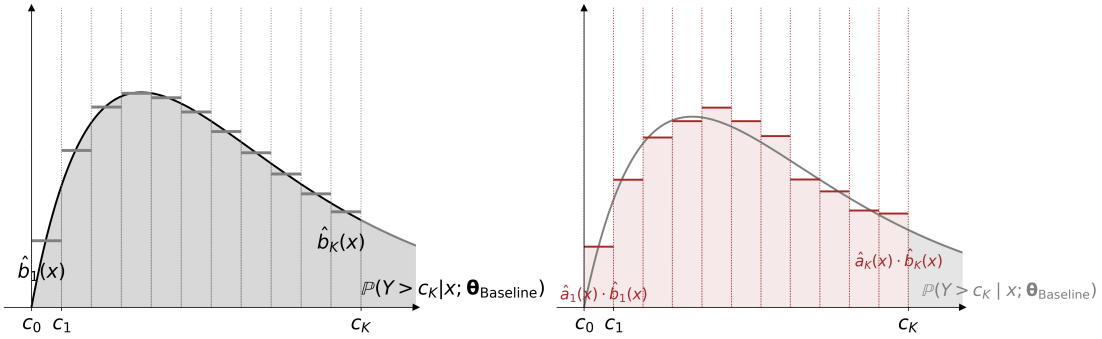



We first approximate the baseline distribution using a $K$-component mixture of uniform distributions over a refinement region $[c_0, c_K)$, defined by ordered cutpoints $\{c_k\}_{k=0}^K$:

$$p(y|\mathbf{f}(\mathbf{x};\boldsymbol{\theta}_{\text{Baseline}})) = \begin{cases} \sum_{k=1}^K \underbrace{\int_{c_{k-1}}^{c_k} p(y|\mathbf{f}(\mathbf{x};\boldsymbol{\theta}_{\text{Baseline}}))\ \mathrm{d}y}_{\hat{b}_k(\mathbf{x})} \cdot \frac{\mathbb{1}_{\{y\in[c_{k-1}-c_k)\}}}{c_k - c_{k-1}}, & \text{if } y \in [c_0, c_K), \\ p(y|\mathbf{f}(\mathbf{x};\boldsymbol{\theta}_{\text{Baseline}})), & \text{otherwise,} \end{cases} \tag{5}$$

# Adjustment Factors



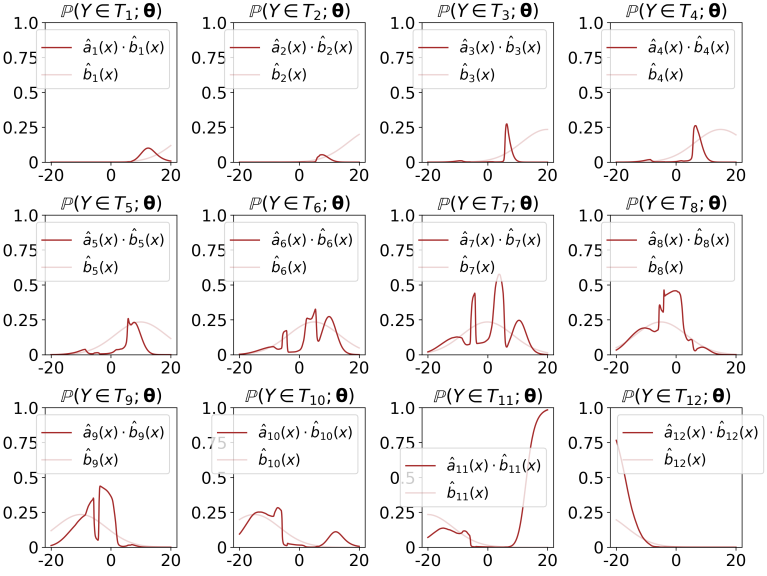

Subsequently, the neural network component learns the "adjustment" function:

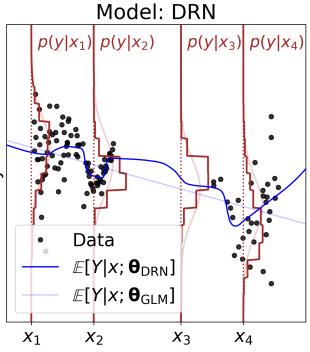
$$\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_{\text{DRN}}) \equiv \hat{\mathbf{a}}(\mathbf{x}) = [\hat{a}_1(\mathbf{x}), \ldots, \hat{a}_K(\mathbf{x})]^\top, \tag{6}$$

which directly refines the baseline model:

$$p(y|\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_{\text{DRN}})) = \begin{cases} \sum_{k=1}^{K} \hat{a}_k(\mathbf{x}) \cdot \hat{b}_k(\mathbf{x}) \cdot \frac{\mathbf{1}_{\{y \in [c_{k-1} - c_k)\}}}{c_k - c_{k-1}}, & \text{if } y \in [c_0, c_K), \\ p(y|\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_{\text{Baseline}})) & \text{otherwise.} \end{cases} \tag{7}$$

# Distributional Refinement



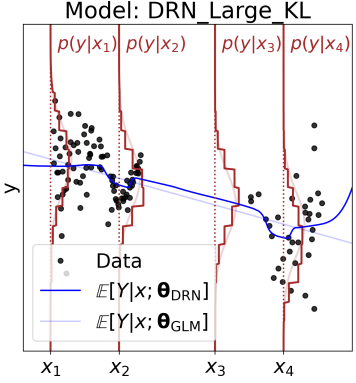
# Controlled Distributional Refinement

Empirically, with a GLM baseline and KL divergence regularisation, a DRN minimises:

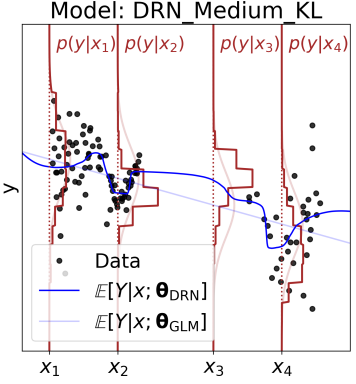
$$\text{Data Loss} + \alpha_1 \cdot \frac{1}{|\mathcal{D}_{\text{Train}}|} \sum_{\mathbf{x}_i \in \mathcal{D}_{\text{Train}}} D_{\text{KL}}[p_{Y|\mathbf{f}(\mathbf{x}_i;\boldsymbol{\theta}_{\text{DRN}})} \| p_{Y|\mathbf{f}(\mathbf{x}_i;\boldsymbol{\theta}_{\text{GLM}})}]$$

$$+ 0.001 \cdot \text{Density Smoothness Loss} + 0 \cdot \text{Mean Resemblence Loss} \qquad (8)$$

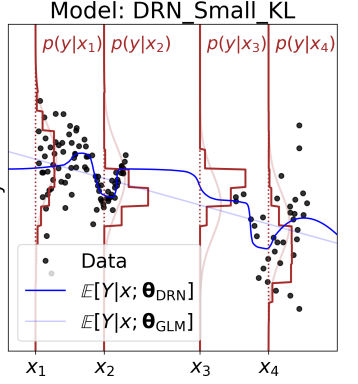
- Large $\alpha_1$




- Medium $\alpha_1$




- Small $\alpha_1$



# A Fun Fact

The KL regularisation term is used in AI Chat Models.

## State-of-the-art LLMs still use this approach

**OpenAI**

"Following Jaques et al. (2017; 2019), we use a KL constraint to prevent the fine-tuned model from drifting too far from the pretrained model."

- *Fine-Tuning Language Models from Human Preferences* (2019)
- *Learning to summarize from human feedback* (2020)
- *InstructGPT* / *ChatGPT* (2022)
- *Direct Preference Optimization (DPO)* (2024)

Google deepseek

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right), \quad (1)$$

$$\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1, \quad (2)$$

# Topics

# drn Python Package

The code below provides a minimal working example of how to use the drn package to train a DRN, with the data preprocessing steps omitted for brevity.

```
from drn import GLM, DRN, drn_cutpoints, train
import pandas as pd
import torch

train_dataset = torch.utils.data.TensorDataset(X_train, y_train)
val_dataset = torch.utils.data.TensorDataset(X_val, y_val)

c_0, c_K = 0, y_train.max().item() * 1.1
cutpoints = drn_cutpoints(c_0, c_K, proportion=0.1, y=y_train)
glm = GLM.from_statsmodels(X_train, y_train, distribution="gamma")

drn_model = DRN(glm, cutpoints, hidden_size=128, num_hidden_layers=2)
train(drn_model, train_dataset, val_dataset, batch_size=256, epochs=10)
```

# Topics

# Conclusion

1. We propose the **Distributional Refinement Network (DRN)**–a neural network that refines a baseline model from a distributional perspective, providing
   - **High distributional flexibility** for distributional regression.
     - E.g., modelling the full conditional distribution.
   - **Controlled anchoring to a trusted baseline**, enhancing trust and improving robustness.
     - E.g., when a strong inductive bias toward the baseline or limited refinement is preferred.
2. A developed `drn` Python package for distributional regression networks, including implementations of MDNs, CANNs, and DRNs.

# Topics

# References I


Bishop, C.M., 1994. Mixture density networks .

Li, R., Reich, B.J., Bondell, H.D., 2021. Deep distribution regression. Computational Statistics & Data Analysis 159, 107203.

Nelder, J.A., Wedderburn, R.W.M., 1972. Generalized linear models. Journal of the Royal Statistical Society. Series A (General) 135, 370–384. URL: http://www.jstor.org/stable/2344614.

Schelldorfer, J., Wüthrich, M.V., 2019. Nesting classical actuarial models into neural networks. SSRN .