Neural Network based enhancement of the Bell regression model for claim frequency data

Central University of Rajasthan

September 06, 2023

Debjyoti Mukherjee
2021MSTA005

Under the guidance of

Dr. Deepesh Bhati

Table of contents:

Overview

- Many of the times the actuarial claim frequency data exhibits over-dispersion phenomenon, which is not encountered by the conventional Poisson model.

Overview

- Many of the times the actuarial claim frequency data exhibits over-dispersion phenomenon, which is not encountered by the conventional Poisson model.
- Various explanatory variables are available along with the claim count so a regression modes are becoming a practical tool.

Overview

- Many of the times the actuarial claim frequency data exhibits over-dispersion phenomenon, which is not encountered by the conventional Poisson model.
- Various explanatory variables are available along with the claim count so a regression modes are becoming a practical tool.
- the conventional GLM is a useful device to study the impact of available co-variates on non-normal response distributions.

Overview

- Many of the times the actuarial claim frequency data exhibits over-dispersion phenomenon, which is not encountered by the conventional Poisson model.
- Various explanatory variables are available along with the claim count so a regression modes are becoming a practical tool.
- the conventional GLM is a useful device to study the impact of available co-variates on non-normal response distributions.
- GLMs are not efficient to capture the non-linear interactions present among the co-variates. Hence, neural network based approach are used to study the non-linear interactions in an efficient manner.

Overview

- Many of the times the actuarial claim frequency data exhibits over-dispersion phenomenon, which is not encountered by the conventional Poisson model.
- Various explanatory variables are available along with the claim count so a regression modes are becoming a practical tool.
- the conventional GLM is a useful device to study the impact of available co-variates on non-normal response distributions.
- GLMs are not efficient to capture the non-linear interactions present among the co-variates. Hence, neural network based approach are used to study the non-linear interactions in an efficient manner.
- Hence, in this work we propose a enhancement of the Bell GLM using feed-forward neural network based approach to (i) automate the pre-processing of the co-variates and (ii) boost the conventional GLM by capturing the non-linear interactions via CANN(Combined Acturial Neural Networks) models.

Bell Distribution

The probability mass function of the Bell distribution Bell(θ) is

$$P(Y = y) = \frac{\theta^y e^{-e^\theta + 1} B_y}{y!} \quad for \quad y = 0, 1, 2, , ... \tag{1}$$

where $\theta > 0$, and the coefficient

$$B_n = \frac{1}{e} \sum_{k=0}^{\infty} \frac{k^n}{k!} \tag{2}$$

are the Bell numbers.

- Starting with $B_0 = B_1 = 1$,the first few Bell numbers are $B_2 = 2, B_3 = 5, B_4 = 15, B_5 = 52$, $B_6 = 203$, $B_7 = 877$, $B_8 = 4140$, $B_9 = 21147$, $B_{10} = 115975$, $B_{11} = 678570, B_{12} = 4213597$ and $B_{13} = 27644437$.

Bell Distribution(Cont.)

The mean and variance of the Bell distribution are, respectively,

$$E(Y) = \theta e^{\theta}, \qquad Var(Y) = \theta(1+\theta)e^{\theta} \tag{3}$$

In a regression model framework, it is typically more useful to model the mean of the response variable. So, to obtain a regression structure for the mean of the Bell distribution, we shall work with a different parameterization of the Bell probability mass function. Let $\mu = \theta e^{\theta}$, then $\theta = W_0(\mu)$, where $W_0(.)$ is the Lambert function. Then it follows from (3)

$$E(Y) = \mu, \qquad Var(Y) = \mu(1 + W_0(\mu)) \tag{4}$$

so that $\mu > 0$ is the mean of the response variable Y .

Bell Distribution(Cont.)

The Bell probability mass function can be written, in the new parameterization, as

$$P(Y = y) = exp(1 - e^{W_0(\mu)})\frac{W_0(\mu)^y B_y}{y!} \qquad y = 0, 1, 2, ... \qquad (5)$$

where $\mu > 0$, and B_y are the Bell numbers in (2) . We have that $W_0(\mu) > 0$ for $\mu > 0$ and, therefore, $Var(Y) > E(Y)$. It implies that the Bell distribution may be suitable for modeling count data with over-dispersion.

Regression model

Assume that the claim numbers, denoted by y_i , are independent and distributed as Bell distribution: $y_i \sim Bell(\mu_i)$
where the mean parameter μ_i depends on the policyholder's characteristics $\mathbf{x}_i$ and an offset of the claims o_i .
For the Bell regression, by choosing the logarithmic link function, we have

$$\hat{\mu} : \mathcal{X} \to \mathcal{R}^+ \qquad \mu_i = exp(o_i + \langle \beta, \mathbf{x}_i \rangle) \tag{6}$$

where β is the unknown coefficient vector to be estimated by MLE.

Neural Networks

- Neural Networks (NNs) are computer systems that are often said to operate in a similar fashion to the human brain.

Neural Networks

- Neural Networks (NNs) are computer systems that are often said to operate in a similar fashion to the human brain.
- A NN is a series of units called neurons. These are usually simple processing units which take one or more inputs and produce an output.

Neural Networks

- Neural Networks (NNs) are computer systems that are often said to operate in a similar fashion to the human brain.
- A NN is a series of units called neurons. These are usually simple processing units which take one or more inputs and produce an output.
- Each input to a neuron has an associated weight which modifies the strength of the input, and an overall bias

Neural Networks

- Neural Networks (NNs) are computer systems that are often said to operate in a similar fashion to the human brain.
- A NN is a series of units called neurons. These are usually simple processing units which take one or more inputs and produce an output.
- Each input to a neuron has an associated weight which modifies the strength of the input, and an overall bias
- When a NN is trained these weights are adjusted to bring the output as close as possible to that desired.

Neural Network Feature Preparation

- Continuous predictors: Min-Max Scaler
  The scaler shifts the predictor linearly between -1 and 1, without changing the relative difference between each value to ensure that all the continuous predictors are of similar importance when performing the gradient descent algorithm.

$$x^{(k)} \rightarrow 2\frac{x^{(k)} - min\ x^{(k)}}{max\ x^{(k)} - min\ x^{(k)}} - 1 \tag{7}$$

- Categorical predictors: Embedding Layer
  Embedding layers aim at mapping every level of categorical feature components to a lower dimensional vector, hence reducing the number of network parameters. Each label of the categorical predictors will then be represented by a 2-d vector when flowing into the main architecture of the network.

Embedding layer

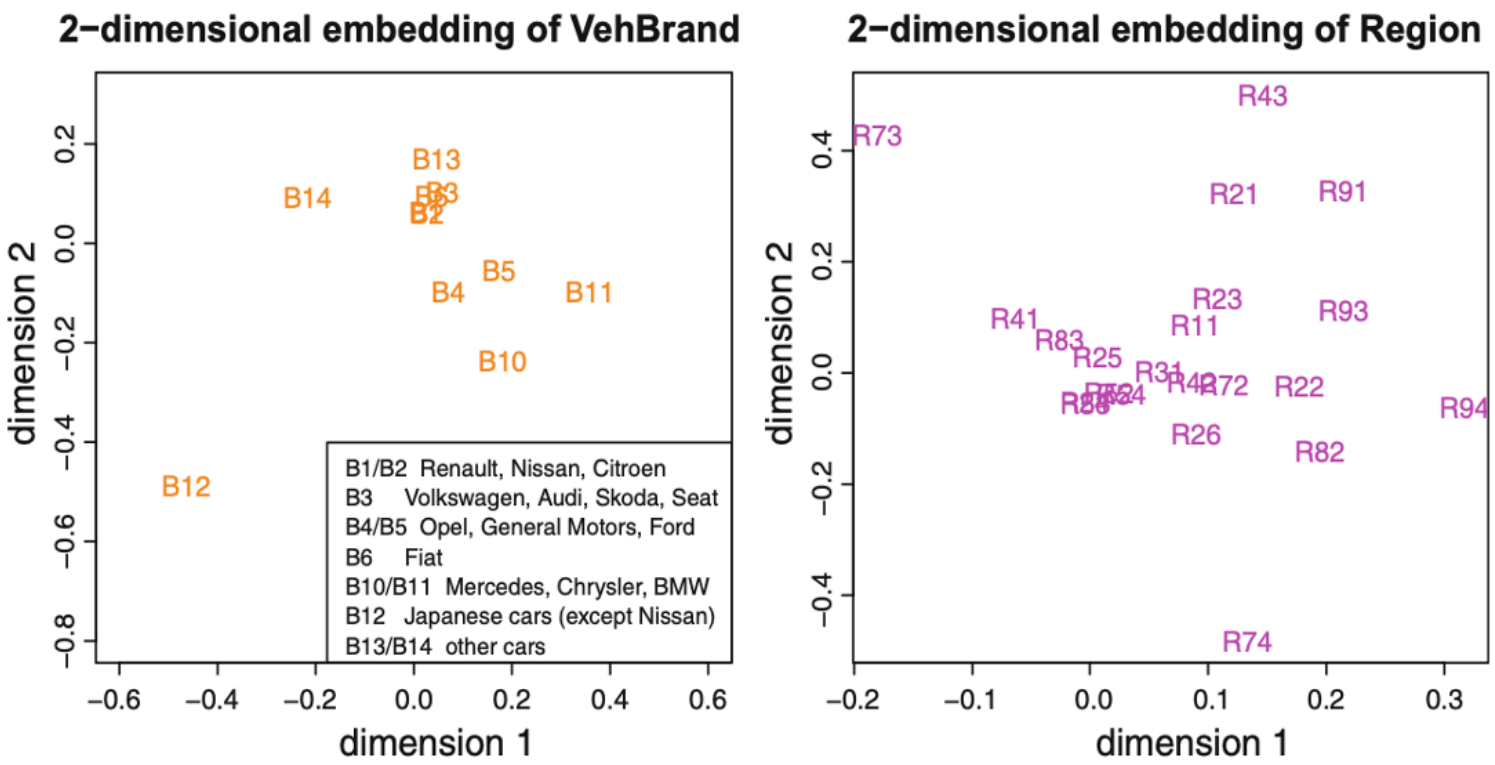

Figure: Embedding weights for $e^{VehBrand} \in \mathcal{R}^2$ and $e^{Region} \in \mathcal{R}^2$ of the categorical variables VehBrand and region for the embedding dimension $d = 2$

Neural Network set-up

- Hyperparameters
  Define the architecture of the neural network. The choice for the hyperparameters (depth, number of neurons, etc.) should be determined carefully to avoid over-fitting. For the dataset we used, the following combination gives relatively good performance:
- Depth: $d = 3$
- Number of Neurons in each layer: 100, 75, 50
- batch size = 100000
- epochs = 1000

Activation Functions

- Activation functions
  Define the link function of the input and output of every individual layer.
  For hidden layers: any non-linear activation function could work; in our case, we choose hyperbolic tangent 'tanh' The choice of non-linear activation functions allows for a non-linear model space and reduce the number of nodes needed.
  This allows the NN to automatically capture the interaction effect of different features.
  For output layer: exponential function, the inverse of log This choice is consistent with the link function of the regression model, as in (7).

Activation Functions(cont.)

	Activation function	Derivative
Sigmoid (logistic) activation	$\phi(x) = (1 + e^{-x})^{-1}$	$\phi' = \phi(1 - \phi)$
Hyperbolic tangent activation	$\phi(x) = \tanh(x)$	$\phi' = 1 - \phi^2$
Exponential activation	$\phi(x) = \exp(x)$	$\phi' = \phi$
Step function activation	$\phi(x) = \mathbb{1}_{\{x \geq 0\}}$	
Rectified linear unit (ReLU) activation	$\phi(x) = x\mathbb{1}_{\{x \geq 0\}}$	

Figure: Various activation functions and their derivatives

Having simple derivatives is an advantage in gradient descent algorithms for model fitting. The hyperbolic tangent activation function is anti-symmetric w.r.t. the origin with range (-1,1). This anti-symmetry and boundedness is an advantage in fitting deep FN network architectures. For this reason we usually prefer the hyperbolic tangent over other activation functions.

Neural-Networks for regression

For neural network fitting, we replace the predictor of regression model

$$\hat{\mu} : \mathcal{X} \to \mathcal{R}^{+} \qquad \mu_i = exp(o_i + \langle \beta, \mathbf{x}_i \rangle) \tag{8}$$

by the neural network predictor

$$\hat{\mu}_{NN} : \mathcal{X} \to \mathcal{R}^{+} \qquad \mu_i = exp(o_i + \langle W^{(d+1)}, (z^{(d)} \bullet ... \bullet z^{(1)})(x_i) \rangle)$$

(9)

where d is the depth of the network and $W^{(d+1)}$ is the weights which maps the neurons of the last hidden layer $z^{(d)}$ to the output layer $\mathcal{R}^{+}$.

Neural Network Architechture

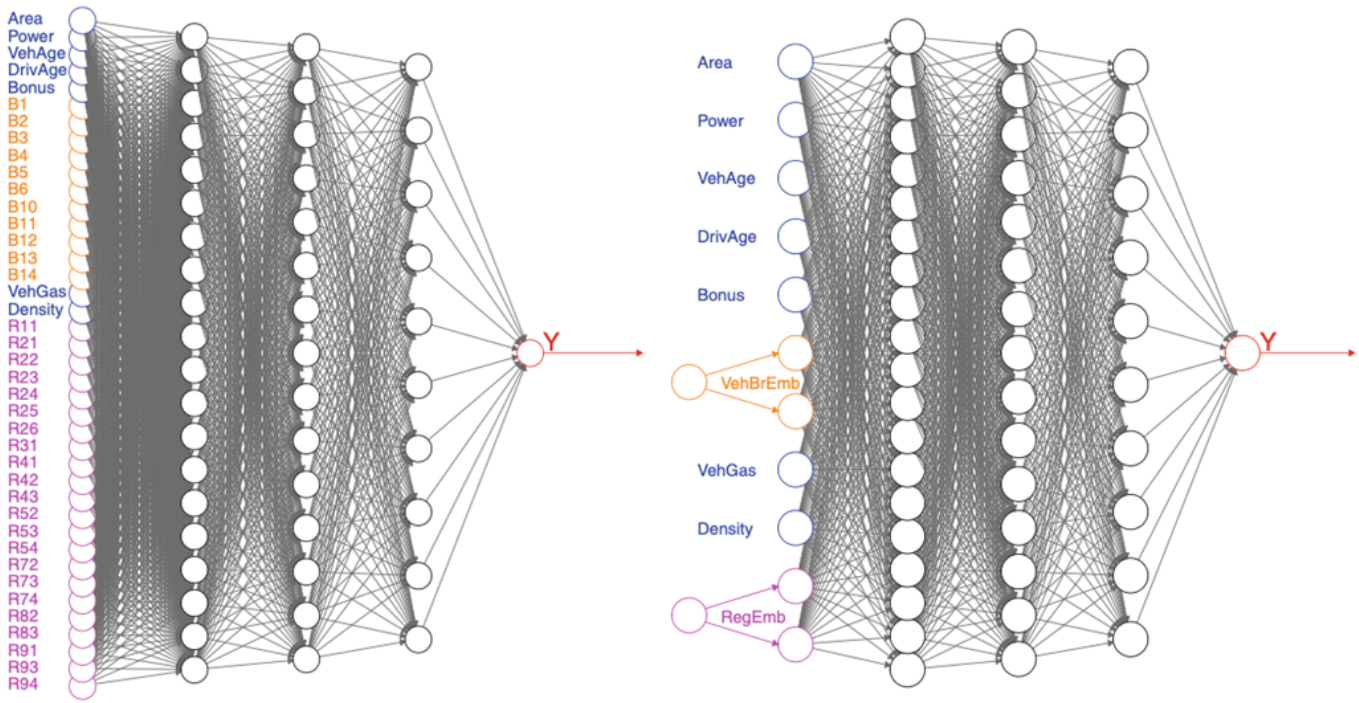

Figure: (lhs) One-hot encoding with $q_0 = 40$ and (rhs) embedding layer for VehBrand and region with embedding dimension 2 and network depth 3

Combined Neural Network for Bell Regression

Following Wuthrich ad Merz's work "Yes, we CANN!", we define a combined neural network - Bell regression model:

$$\hat{\mu}_{CANN} : \mathcal{X} \to \mathcal{R}^{+}$$

$$\mu_i = exp(o_i + \langle \beta, \mathbf{x}_i \rangle + \langle W^{(d+1)}, (z^{(d)} \bullet ... \bullet z^{(1)})(x_i) \rangle) \quad (10)$$

The
first term in (16) is the regression function (also called the skip connection), and the second term in (16) is the neural network function.

Combined Neural Network Architechture

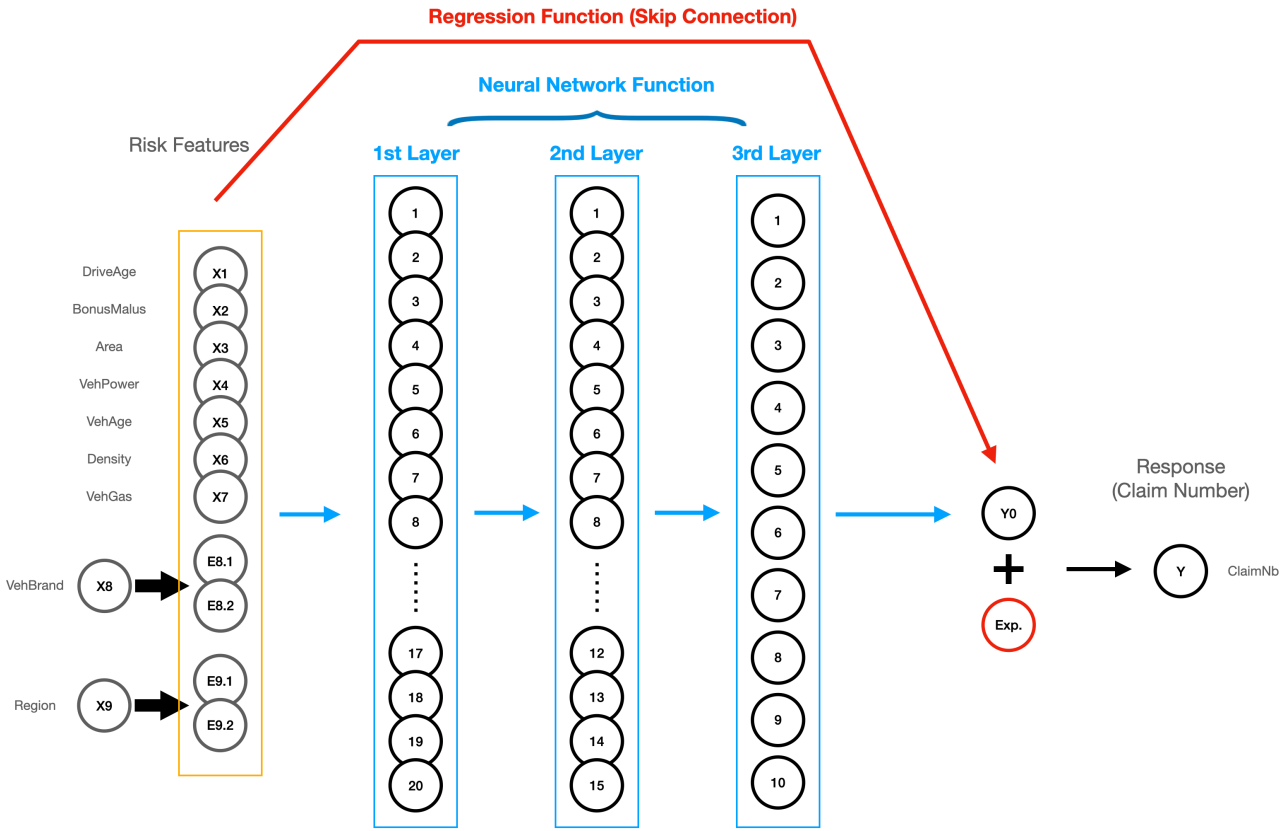

Figure: CANN architechture with Skip connection

Deviance Loss

To train the above Neural Network model, the gradient descent methods will perform on an objective loss function to find the coecient which gives the minimum value of the loss function. Therefore, the deviance loss is introduced. For Bell distribution, the deviance loss is given by:

$$\mathcal{D}(\mathbf{Y}, \hat{\boldsymbol{\mu}}) = \begin{cases} \frac{2}{n}\sum_{i=1}^{n}(e^{W_0(\hat{\mu}_i)} - 1) & y_i = 0 \\ \frac{2}{n}\sum_{i=1}^{n}[e^{W_0(\hat{\mu}_i)} - e^{W_0(y_i)} + y_i.log(\frac{W_0(y_i)}{W_0(\hat{\mu}_i)})] & y_i > 0 \end{cases} \tag{11}$$

The reason why we introduce the deviance loss is because minimization of deviance loss is equivalent to MLE, so we can make the Bell regression model comparable to the above discussed neural network models.

Data Description

We used claim frequency data from a French Motor Third-Party Liability (freMTPL) dataset in the R package CASdatasets.

```
> str(data)
'data.frame':   678013 obs. of  12 variables:
 $ IDpol     : num  1 3 5 10 11 13 15 17 18 21 ...
 $ ClaimNb   : num [1:678013(1d)] 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "dimnames")=List of 1
  .. ..$ : chr [1:678013] "139" "414" "463" "975" ...
 $ Exposure  : num  0.1 0.77 0.75 0.09 0.84 0.52 0.45 0.27 0.71 0.15 ...
 $ Area      : Factor w/ 6 levels "A","B","C","D",..: 4 4 2 2 2 5 5 3 3 2 ...
 $ VehPower  : int  5 5 6 7 7 6 6 7 7 7 ...
 $ VehAge    : int  0 0 2 0 0 2 2 0 0 0 ...
 $ DrivAge   : int  55 55 52 46 46 38 38 33 33 41 ...
 $ BonusMalus: int  50 50 50 50 50 50 50 68 68 50 ...
 $ VehBrand  : Factor w/ 11 levels "B1","B10","B11",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ VehGas    : chr  "Regular" "Regular" "Diesel" "Diesel" ...
 $ Density   : int  1217 1217 54 76 76 3003 3003 137 137 60 ...
 $ Region    : Factor w/ 22 levels "R11","R21","R22",..: 18 18 3 15 15 8 8 20 20 12
```

Figure: Strucutre of the dataset

Data Description(Cont.)

Table: Split of the porfolio w.r.t. the number of claims

0	1	2	3	4	5	6	8	9	11	16
643953	32178	1784	82	7	2	1	1	1	3	1

which shows that nearly 95 percent of the data have 0 claim count , hence the data is overdispersed.
A detaile descriptive analysis of this dataset is provided in the tutorial of Noll. et al.. The analysis in that reference also includes a minor data cleaning part on the original data which is used here for further analysis of GLM and neural Networks.

Bias Regularization

There are two reasons that could possibly lead to the above bias on the portfolio level:

- The Bell regression model provides unbiased estimator only if the canonical link is chosen. However, the choice of link function, the logarithm, is not the canonical link of the Bell distribution. (The usage of canonical link will cause issues in the MLE process, especially for complex feature spaces.)
- The fitting process of neural network model and combined model utilizes an early-stopping algorithm to prevent from over-fitting issues on the individual policy level at the cost of bias on the portfolio level.

To tackle this problem, we apply a simple bias regularization method from Wuthrich and Merz's book. We adjust the intercept in the last layer of the neural network, or the regression coefficient, to shift the biased results.

Bias Regularization Method

Since the logarithm is the link function, the adjustment of the intercept in the linear function is simply to multiply the results by a fixed coefficient c, given by

$$c = \frac{\bar{\mu}}{\hat{\mu}}$$

where $\bar{\mu}$ is the mean of the observed claim numbers Y_i and $\hat{\mu}$ is the mean of the predicted values $\hat{y}_i$.

Results for the freMTPL dataset

Table: Training loss, testing loss and portfolio average

model	Training Loss	Testing Loss	Portfolio Average
Data			0.053
Bell GLM	28.82	28.36	0.056
Network Emb (d =1)	27.6	27.225	0.007
Network Emb (d =1) w/ reg.	27.673	27.251	0.006
Network Emb (d =2)	27.884	27.442	0.006
Network Emb (d =2) w/ reg.	27.74	27.322	0.06
CANN (d =1)	27.92	27.562	0.055
CANN (d =1) w\ reg.	27.693	27.361	0.055
CANN (d =2)	27.636	27.316	0.055
CANN (d =2) w\ reg.	27.456	27.159	0.054

Plot of the Training and Validation LOSS

Conclusions:

- CANN allows us to identify missing structure in GLMs (more) explicitly and helps boost the base GLM model.
- CANN model combined with bias regularization methods give better fit to the data (both training and testing) and also handles the portfolio average quiet well.
- CANN allows us to learn across different portfolios.

References

- Ferrario, A., Noll, A., Wuthrich, M. V. (2020). Insights from inside neural networks. *Available at SSRN 3226852*.
- Jose, A., Macdonald, A. S., Tzougas, G., Streftaris, G. (2022). A combined neural network approach for the prediction of admission rates related to respiratory diseases. *Risks, 10(11)*, 217.
- Noll, A., Salzmann, R., Wuthrich, M. V. (2020). Case study: French motor third-party liability claims. *Available at SSRN 3164764*.
- Schelldorfer, J., Wuthrich, M. V. (2019). Nesting classical actuarial models into neural networks. *Available at SSRN 3320525*.
- Tzougas, G., & Kutzkov, K. (2023). Enhancing Logistic Regression Using Neural Networks for Classification in Actuarial Learning. *Algorithms*, 16(2), 99.
- Wuthrich, M. V., & Merz, M. (2019). Yes, we CANN!. *ASTIN Bulletin: The Journal of the IAA*, 49(1), 1-3.

Thank you