Neural Network Embedding of the Negative Binomial Regression Model for Claim Frequencies

George Tzougas Ziyi Li

Department of Statistics
London School of Economics and Political Sciences

June 16, 2021

Table of Contents

The Negative Binomial Distribution

The probability mass function of Negative Binomial (NB) distribution $NB(\mu, \phi)$ is

$$P_k = \mathbb{P}(y = k) = \frac{\Gamma(k+\phi)}{k! \cdot \Gamma(\phi)} \left(\frac{\mu}{\phi+\mu}\right)^k \left(\frac{\phi}{\phi+\mu}\right)^\phi \qquad k = 0, 1, 2, \cdots \qquad (1)$$

$NB(\mu, \phi)$ has the expectation

$$\mathbb{E}[Y] = \mu$$

and the variance

$$Var(Y) = \mu + \frac{\mu^2}{\phi}$$

where ϕ is the dispersion parameter.
Compared with Poisson, the additional dispersion parameter controls the over-dispersion of the distribution.

Regression Model

Assume that the claim numbers, denoted by y_i, are independent and distributed as Negative Binomial distribution

$$y_i \sim NB(\mu_i, \phi)$$

where the mean parameter μ_i depends on the policyholder's characteristics x_i and an offset of the claims o_i.
Note that we assume the distribution of every y_i have a uniform dispersion parameter ϕ.

For the NB regression, by choosing the logarithmic link function, we have

$$\lambda^{NB} : \mathcal{X} \mapsto \mathbb{R}_+ \qquad \mu_i = \lambda^{NB}(x_i) = \exp(o_i + \langle \boldsymbol{\beta}, x_i \rangle) \tag{2}$$

where $\boldsymbol{\beta}$ is the unknown coefficient to be estimated by MLE.

Neural Network

- 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.
- 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.

$$x^{(k)} \mapsto 2\frac{x^{(k)} - \min x^{(k)}}{\max x^{(k)} - \min x^{(k)}} - 1 \tag{3}$$

- 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.

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: 20, 15, 10

Neural Network set-up

- 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 Eq. (2).

Neural Network Architecture

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

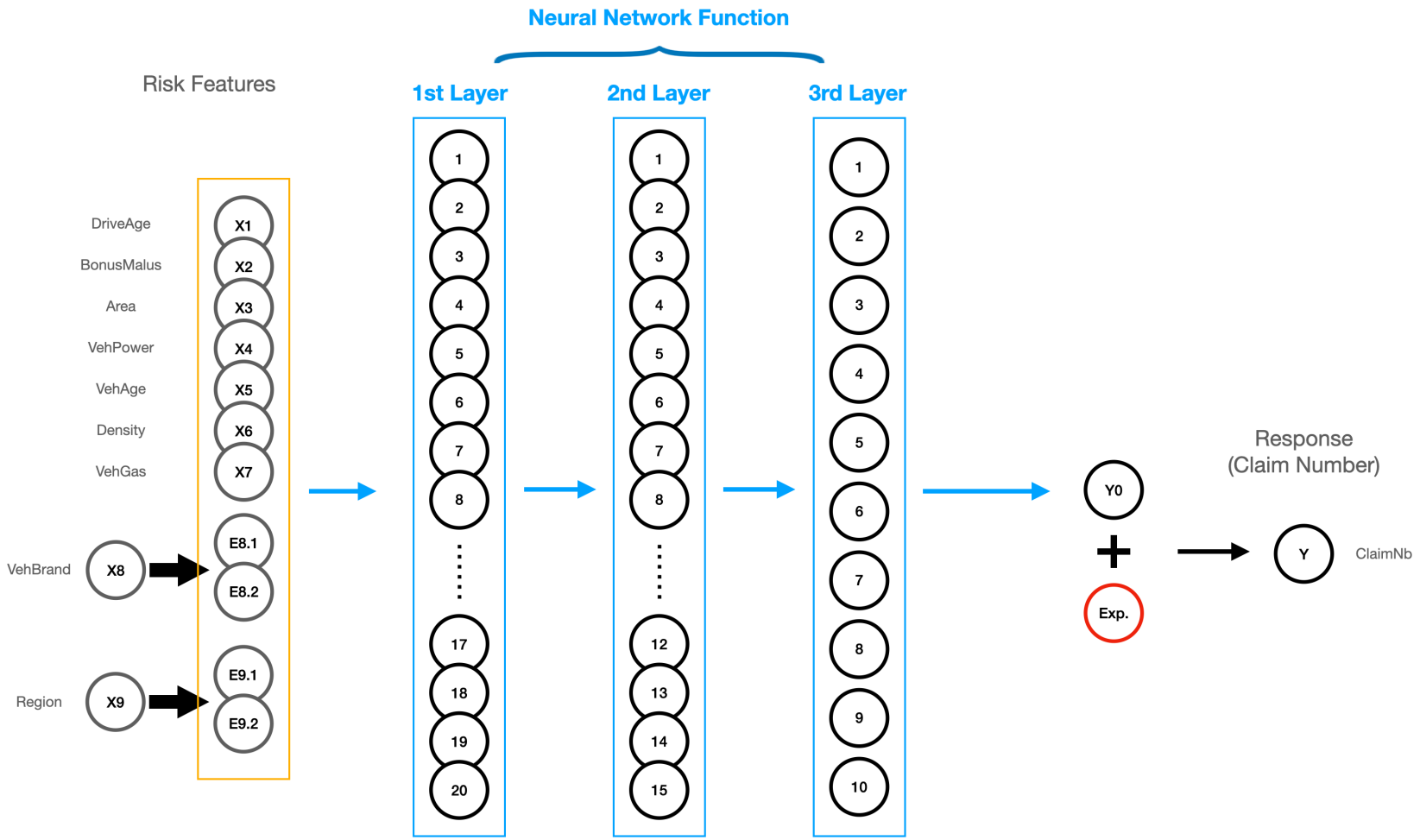
$$\lambda^{NB} : \mathcal{X} \mapsto \mathbb{R}_+ \qquad \mu_i = \lambda^{NB}(x_i) = \exp(o_i + \langle \boldsymbol{\beta}, x_i \rangle)$$

by the neural network predictor

$$\begin{aligned} &\lambda^{NN} : \mathcal{X} \mapsto \mathbb{R}_+ \\ &\mu_i = \lambda^{NN}(\boldsymbol{x}_i) = \exp\left(o_i + \left\langle \boldsymbol{w}^{(d+1)}, \left(\boldsymbol{z}^{(d)} \circ \cdots \circ \boldsymbol{z}^{(1)}\right)(\boldsymbol{x}_i) \right\rangle\right) \end{aligned} \tag{4}$$

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

Neural Network Architecture



Combined Neural Network Negative Binomial Regression Model

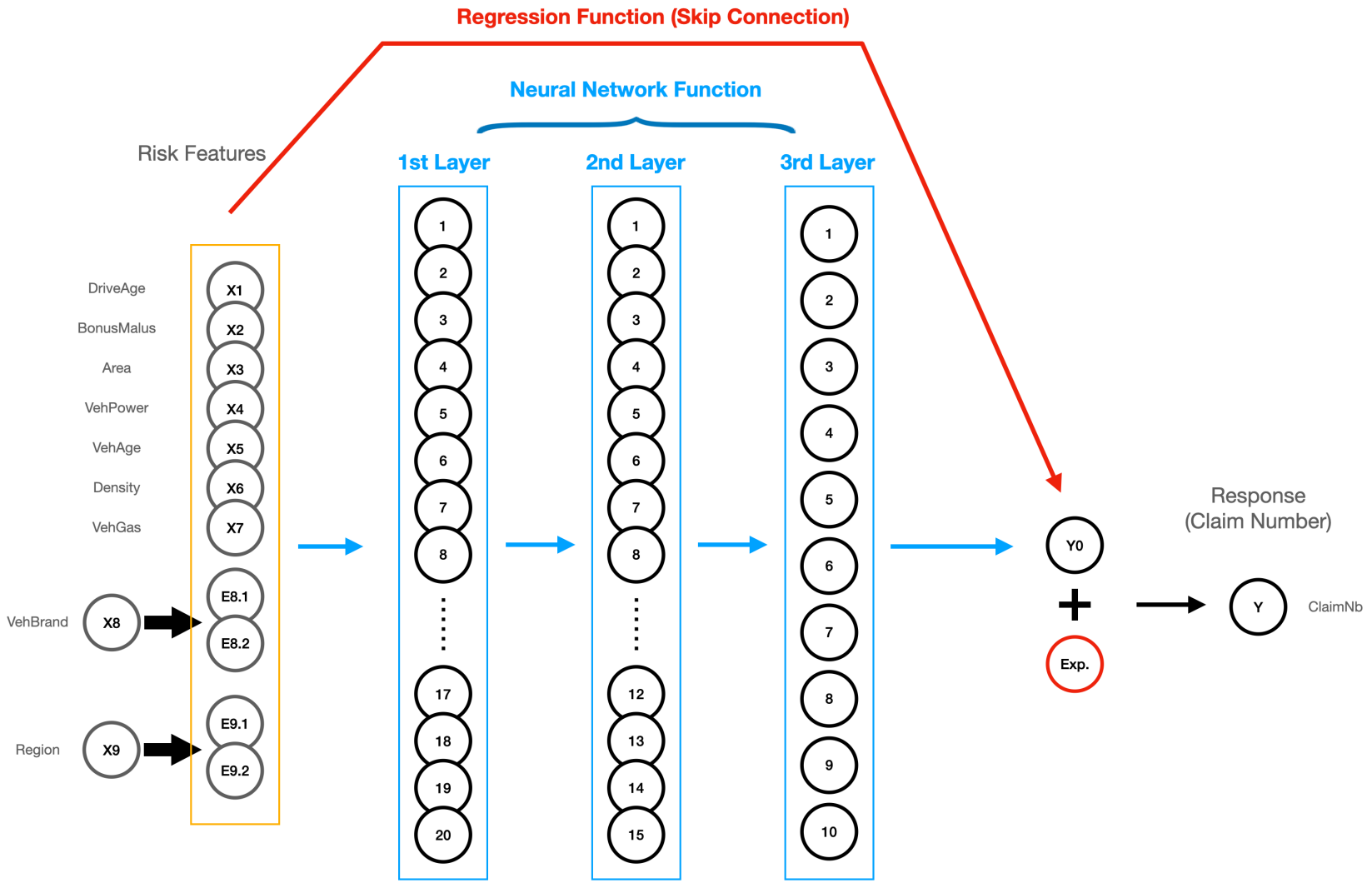
Following Wuthrich ad Merz's work "Yes, we CANN!", we define a combined neural network - Negative Binomial regression model (CNNNBR):

$$\lambda^{CNNNBR} : \mathcal{X} \mapsto \mathbb{R}_+$$

$$\mu_i = \lambda^{CNNNBR}(\boldsymbol{x}_i) = \exp\left(o_i + \langle \boldsymbol{\beta}, x_i \rangle + \left\langle \boldsymbol{w}^{(d+1)}, \left(\boldsymbol{z}^{(d)} \circ \cdots \circ \boldsymbol{z}^{(1)}\right)(\boldsymbol{x}_i) \right\rangle\right) \tag{5}$$

The first term in Eq. (5) is the regression function (also called the skip connection), and the second term in Eq. (5) is the neural network function.

Combined Model Architecture



Deviance Loss

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

$$\mathcal{L}_{\mathcal{A}}(\boldsymbol{\beta}) = \frac{2}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} \Big(y_i \log y_i - (y_i + \phi) \log(y_i + \phi) - y \log \hat{\mu}_i + (y_i + \phi) \log(\hat{\mu}_i + \phi) \Big) \tag{6}$$

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

Data Description

We used real claim frequency data from a French Motor Third-Part Liability dataset in the R package `CASdatasets`.

- **Response** is the number of claims.
- The **explanatory variables** are:
  - 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 : Factor w/ 2 levels " Diesel "," Regular ": 2 2 1 1 1 2 2 1 1 1 ...
  - 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 ...

over 40 dimensional feature space!

Comparison of the Negative Binomial Regression, Neural Network and Combined Model

We compare the above three methods:

- Training loss, testing loss: evaluate the performance in training set and testing set, respectively
- Portfolio average: gives a indicator for the level of the portfolio bias

Model	Training loss	Testing loss	Portfolio average
Real data			0.053
NB regression	27.8210	28.5570	0.054
Neural Network	26.9288	27.9248	0.055
Combined method	26.9150	27.9150	0.055

The NN model and the combined model give better performance compared with the regression model, i.e. they give more accurate estimation on an individual policy level.
Despite the higher testing loss, the NB regression gives a closer estimation to the real data for the portfolio average.

Bias regularization method

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

1. The NB 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 Negative Binomial distribution. (The usage of canonical link will cause issues in the MLE process, especially for complex feature spaces.)
2. The fitting process of neural network model and combined model utilizes a 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 $\hat{Y}_i$.

Then it can be proved that the new predicted values $c\hat{Y}_i$ have the same mean as the observed values Y_i.

$$\frac{1}{n}\sum c\hat{Y}_{=}c\frac{1}{n}\sum \hat{Y}_i = \frac{\bar{\mu}}{\hat{\mu}}\hat{\mu} = \bar{\mu}$$

Regularized Models

We also compare the results after applying the regularization method.

Model	Training loss	Testing loss	Portfolio average
Real data			0.053
NB regression	27.8210	28.5570	0.054
Reg. NB regression	27.8218	27.5630	0.053
Neural Network	26.9288	27.9248	0.055
Reg. NN	26.9300	27.9396	0.053
Combined method	26.9150	27.9150	0.055
Reg. Combined	26.9139	27.9313	0.053

As seen in the above table, the regularized model significantly narrow the difference of the portfolio average between the real data and the model without much influence on the testing loss.

Summary

1. Normally the regression model gives less accurate prediction on an individual policy level. By pre-specifying the fitting formula of the regression, the accuracy would very possibly improve.
2. Although the neural network model captures the general interaction effect of features automatically, it results in a biased estimation on the portfolio level due to the early stopping.
3. The bias regularization method reduces the portfolio bias without influencing the accuracy. Also, more advanced bias regularization methods could be researched, e.g. utilizing GLM or some penalty in the loss function.

These three models, along with effective bias reduction, could further provide some insights to improve the process of rate-making, reserving, and claim prediction in the industrial practice.