EDITORIAL: YES, WE CANN!

The aim of this editorial is to increase the acceptance of neural net modeling in the actuarial community. Neural nets may substantially improve classical actuarial models, if appropriately applied. We illustrate this on a toy example but, in fact, this should be understood as a universal concept. Assume we have a classical regression problem where the distribution of a response $Y = Y(\boldsymbol{x})$ can be described by covariates $\boldsymbol{x}$. A common actuarial problem is to determine the premium $\mu(\boldsymbol{x}) = \mathbb{E}[Y(\boldsymbol{x})]$ as a function of the covariates $\boldsymbol{x}$. Actuaries have developed excellent skills to solve such problems through finding appropriate regression functions $\boldsymbol{x} \mapsto \mu(\boldsymbol{x})$. This editorial shows how these skills can further be improved using the toolbox of neural nets. For this, the following two assumptions are crucial:

(1) There exists a sufficiently good classical parametric actuarial model for describing the responses. The parameters of this model can efficiently be estimated with maximum likelihood estimation (MLE) methods.
(2) Embedding of the classical actuarial model in a neural net is feasible.

If (1) and (2) are fulfilled, then one proceeds as follows:

(3) The calibration of the neural net uses the classical actuarial model as initial value in the gradient descent algorithm with the deviance loss as objective function.

This embedding of the classical actuarial model in a neural net can be interpreted as model blending or as neural net boosting of the actuarial model. We describe this on an easy example and call it the Combined Actuarial Neural Net (CANN) approach.

(1) The existence of a classical parametric actuarial model that describes the responses sufficiently accurately is the starting point, and (2) and (3) aim at challenging this model. MLE maximizes the likelihood function or, equivalently, minimizes the deviance loss. As toy example we choose a simple generalized linear model (GLM) with regression function

$$\boldsymbol{x} \mapsto \mu^{\mathrm{GLM}}(\boldsymbol{x}) = \exp\left\{\langle \boldsymbol{x}, \beta \rangle\right\},$$

where $\langle \boldsymbol{x}, \beta \rangle$ denotes the scalar product between the covariate $\boldsymbol{x}$ and the model parameter β. For illustration, we choose an exponential response function. We assume an efficient MLE of β denoted by $\widehat{\beta}^{\mathrm{MLE}}$.

(2) In a first step, we describe the classical parametric actuarial model in a (first) neural net architecture. Neural nets offer great flexibility, and in many/most cases this first step is possible. For our GLM this is straightforward, we choose one neuron in the output layer with weights β and exponential activation function.

In a second step, we choose a second neural net architecture that describes the same regression problem. In our toy example, we assume that this second neural net has two hidden layers with hyperbolic tangent activation function and neurons $\boldsymbol{z}^{(1)}$ in the first hidden layer and neurons $\boldsymbol{z}^{(2)}$ in the second hidden layer. This is schematically shown by the following two sets of hidden neurons

$$\boldsymbol{x} \mapsto \boldsymbol{z}^{(1)}(\boldsymbol{x}) = \tanh\left(\boldsymbol{b}_1 + B_1'\boldsymbol{x}\right) \quad \text{and} \quad \boldsymbol{x} \mapsto \boldsymbol{z}^{(2)}(\boldsymbol{x}) = \tanh\left(\boldsymbol{b}_2 + B_2'\boldsymbol{z}^{(1)}(\boldsymbol{x})\right),$$

with intercept vectors $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$, and weight matrices B_1 and B_2. Remark that $\tanh(\cdot)$ is applied element-wise. This provides a neural net regression function

$$\boldsymbol{x} \mapsto \mu^{\mathrm{NN}}(\boldsymbol{x}) = \exp\left\{b_3 + B_3'\boldsymbol{z}^{(2)}(\boldsymbol{x})\right\},$$

with exponential activation function, intercept b_3, and weights B_3 in the output layer. The art of neural net designing is first to find a good architecture and second a good calibration of that architecture.

In the last step, we blend the two regression functions μ^{GLM} and μ^{NN} to the following CANN regression function

$$\boldsymbol{x} \mapsto \mu^{\mathrm{CANN}}(\boldsymbol{x}) = \exp\left\{\langle \boldsymbol{x}, \beta\rangle + b_3 + B_3'\boldsymbol{z}^{(2)}(\boldsymbol{x})\right\}.$$

In machine learning jargon, this is called adding a skip connection to the feed-forward neural net μ^{NN}. This skip connection exactly contains the classical actuarial regression function $\log(\mu^{\mathrm{GLM}})$. This blended model has network parameter $\theta = (\beta, \boldsymbol{b}_1, \boldsymbol{b}_2, b_3, B_1, B_2, B_3)$. State-of-the-art neural net modeling uses versions of the gradient descent algorithm to minimize the objective function for finding a good network parameter θ.

(3) As described above, we use a version of the gradient descent algorithm for finding a good network parameter θ. To be consistent with the GLM approach of (1) we choose the deviance loss function as the objective function in the gradient descent algorithm. The second important ingredient for gradient descent is the initial value θ_0 for θ in the algorithm. We choose as initial value θ_0 the MLE $\widehat{\beta}^{\mathrm{MLE}}$ of (1) for β, and we set $b_3 = 0$ and $B_3 = \boldsymbol{0}$. This initialization is the crucial and subtle point in our concept, it implies that the gradient descent algorithm starts in μ^{GLM}, and the blended model μ^{CANN} can be understood as a boosting enhancement where the gradient descent algorithm challenges μ^{GLM} for additional model structure. In particular, the gradient descent algorithm tries to reduce the initial GLM deviance loss (in-sample) by exploring the CANN structure.

We close with remarks.

- If the blended model μ^{CANN} provides a substantial improvement in the deviance loss (subject to over-fitting), then μ^{GLM} misses important model structure and the CANN approach μ^{CANN} should be used. Otherwise we should stay with the GLM.
- If we have a sufficiently good GLM from (1), we expect fast (near) convergence of the gradient descent algorithm because we are already starting the algorithm in a model that is close to optimal. In particular, this implies that bootstrapping of neural nets becomes feasible.
- In the gradient descent algorithm described in (3) we may either train β or declare the initial value $\widehat{\beta}^{\text{MLE}}$ to be an untrainable weight. We may also modify the corresponding GLM part to $\langle \boldsymbol{x}, \alpha\widehat{\beta}^{\text{MLE}} \rangle$ with a trainable weight $\alpha \in (0, 1)$ and letting $\widehat{\beta}^{\text{MLE}}$ being untrainable. In this latter case α is interpreted as the credibility weight assigned to the GLM part.
- The CANN approach can be applied to a huge variety of classical parametric actuarial models, in particular, we can easily treat categorical covariates using embedding layers in neural nets.
- Furthermore, the CANN approach even works if we consider several portfolios simultaneously in one blended neural net. This then allows learning across different portfolios. Another option is to have several parametric actuarial models in different skip connections, and letting them compete against each other.

MARIO V. WÜTHRICH AND MICHAEL MERZ