Neural Networks Applied to Chain-Ladder Reserving

Mario V. Wüthrich[*]

July 6, 2018

Abstract

Classical claims reserving methods act on so-called claims reserving triangles which are aggregated insurance portfolios. A crucial assumption in classical claims reserving is that these aggregated portfolios are sufficiently homogeneous so that a coarse reserving algorithm can be applied. We start from such a coarse reserving method, which in our case is Mack's chain-ladder method, and show how this approach can be refined for heterogeneity and individual claims feature information using neural networks.

Keywords. Claims reserving, Mack's CL model, individual claims reserving, micro-level reserving, neural networks, individual claims features, claims covariates.

1 Introduction

The starting point of our claims reserving analysis is Mack's chain-ladder (CL) model [14]. For applying Mack's CL model we aggregate individual claims payments by accident period and payment delay. These aggregated claims payments are then assumed to fulfill a (simplified) regression assumption, see formula (2.1) below. The main goal of this work is to extend this simplified regression assumption by allowing for the inclusion of individual claims information. This individual claims information will lead to a refinement of the claims reserves and it will help us to understand structural differences between different types of claims. We explore the framework of neural networks for the modeling of these structural differences, that is, we replace the simplified regression assumption (2.1) by a (non-linear) neural network regression model that accounts for the individual differences between claims. As will become apparent below, this is not possible in Mack's CL model, in general. Therefore, we work in an extended version thereof which is of a similar nature as Schnieper's model [21].
Our approach focuses on differentiating types of claims within Mack's CL model framework. This can be seen as a top-down approach correcting for heterogeneity. Recently, there has been an increasing interest in bottom-up approaches in the spirit of Arjas [2] and Norberg [17, 18], see, for instance, Antonio-Plat [1], Badescu et al. [3, 4], Baudry-Robert [5], Harej et al. [8], Jessen et al. [11], Lopez [13], Pigeon et al. [19], Verrall-Wüthrich [22] or Zarkadoulas [25]. These references aim at modeling claims on a micro-level using either parametric or non-parametric models, and

[*]ETH Zurich, RiskLab, Department of Mathematics, 8092 Zurich, Switzerland

claims reserves are obtained by aggregating individual reserves. Our approach does not consider individual claims as such, but rather (homogeneous) sub-portfolios of different claim types for which a CL assumption is made and reserves are calculated.

Outline of this paper. In the next section we introduce an extended CL model allowing for the inclusion of individual claims feature information. We also describe the complications that may result from this extension. In Section 3 we describe how neural networks can be used for CL reserving and we describe how these networks are calibrated to our data using the gradient descent method. In Section 4 we treat the previously mentioned complications which require additional modeling. In Section 5 we apply our model to insurance data which consists of 4'970'856 individual claims histories. We benchmark our model with Mack's CL approach and we show that our neural network extended CL model passes the use test. In Section 6 we conclude and give an outlook.

2 Model assumptions

We start by introducing the relevant notation. For describing Mack's CL model we introduce indexes $1 \leq i \leq I$ for the accident years and indexes $0 \leq j \leq J$ for the development years under consideration. $C_{i,j}$ denotes the cumulative claims payments for claims with accident year i which are done within the first j development years, and $C_{i,J}$ denotes the total ultimate claim amount of accident year i. The CL algorithm then assumes existence of CL factors (parameters) $f_0, \ldots, f_{J-1} > 0$ such that for all accident years $1 \leq i \leq I$ and all development years $1 \leq j \leq J$ we have the simplified regression structure

$$C_{i,j} \;\approx\; f_{j-1}\, C_{i,j-1}. \tag{2.1}$$

We are going to be more precise below about the explicit meaning of the symbol "$\approx$". Structure (2.1) means that the cumulative claim $C_{i,j}$ is roughly proportional to the previous cumulative claim $C_{i,j-1}$ (of the same accident year i) with a proportionality factor f_j that does not depend on accident year i.

In the present work we are going to refine regression structure (2.1) by describing the claims more accurately using individual claims feature information. We therefore introduce a feature space $\mathcal{X}$ with features $\boldsymbol{x} \in \mathcal{X}$ describing the characteristics of individual claims. In our example the additional feature information comprises:

- the line of business `LoB` $\in \{1, \ldots, 4\}$ the individual claim is belonging to;
- the claims code `cc` $\in \{1, \ldots, 53\}$ denoting the labor sector the injured is working in;
- the accident quarter `AQ` $\in \{1, \ldots, 4\}$ of the occurrence of the individual claim;
- the age of the injured `age` $\in \{15, 16, \ldots, 70\}$ in years at claims occurrence;
- the injured body part `inj_part` $\in \{10, \ldots, 99\}$.

Thus, we consider a 5-dimensional feature space $\mathcal{X}$ collecting the features $\boldsymbol{x} = (x_1, \ldots, x_5)' = (\texttt{LoB}, \texttt{cc}, \texttt{AQ}, \texttt{age}, \texttt{inj\_part})'$. We deliberately omit the accident year i information because a basic assumption of the CL algorithm is that the CL factors f_j do not depend on accident years. We also omit an available claims number because typically the claims number is a unique claims identifier that does not have any predictive power; the explicit choice of the data is illustrated in Listing 1 in the appendix below.

Our feature information considered is static, in particular, it does not change over time. We define $C_{i,j}(\boldsymbol{x})$ to be all cumulative claims payments for claims with accident year i done within the first j development periods and having feature value $\boldsymbol{x} \in \mathcal{X}$. The refinement of (2.1) then uses feature individual CL factors $f_0(\boldsymbol{x}), \ldots, f_{J-1}(\boldsymbol{x}) > 0$ such that

$$C_{i,j}(\boldsymbol{x}) \;\approx\; f_{j-1}(\boldsymbol{x})\; C_{i,j-1}(\boldsymbol{x}). \tag{2.2}$$

Notice that there is a difficulty with this approach (2.2), namely, there may be relevant features $\boldsymbol{x}$ for which we have $C_{i,j-1}(\boldsymbol{x}) = 0$. This may occur due to two different reasons: (a) we do not have any reported claims for accident year i within the first j development periods for that given feature $\boldsymbol{x}$, or (b) we have reported claims for that feature $\boldsymbol{x}$ but we did not make any claims payments for these claims within the first j development periods (note that development periods j start in 0). In these two cases the CL algorithm (2.2) will not work for these feature values $\boldsymbol{x}$, and we need an extension that is similar to Schnieper's model [21], using an intercept.

For all what follows we make the following standing assumptions (not repeated each time):

- Feature space $\mathcal{X}$ is finite. In our case we have $|\mathcal{X}| = 4 \cdot 53 \cdot 4 \cdot 56 \cdot 90 = 4'273'920$ different possible feature values.
- $(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{F})$ is a filtered probability space with discrete time filtration $\mathbb{F} = (\mathcal{F}_t)_{0 \le t \le I+J}$.
- Cumulative claims $C_{i,j}(\boldsymbol{x})$ are $\mathcal{F}_{i+j}$-measurable for all $1 \le i \le I$, $0 \le j \le J$ and $\boldsymbol{x} \in \mathcal{X}$.
- Cumulative claims $C_{i,j}(\boldsymbol{x})$ are non-negative, $\mathbb{P}$-a.s., and on all events $\{C_{i,k}(\boldsymbol{x}) > 0\}$ we have $C_{i,j}(\boldsymbol{x}) > 0$, $\mathbb{P}$-a.s., for all $k \le j \le J$.

We make the following model assumptions.

Model Assumptions 2.1 *Cumulative claims $C_{i,j}(\boldsymbol{x})$ of different accident years $1 \le i \le I$ or different features $\boldsymbol{x} \in \mathcal{X}$ are independent. There exist parameters $f_0(\boldsymbol{x}), \ldots, f_{J-1}(\boldsymbol{x}) > 0$ such that for all $1 \le i \le I$, $1 \le j \le J$ and $\boldsymbol{x} \in \mathcal{X}$*

$$\mathbb{E}\left[C_{i,j}(\boldsymbol{x})\middle|\,\mathcal{F}_{i+j-1}\right] = f_{j-1}(\boldsymbol{x})\; C_{i,j-1}(\boldsymbol{x}) \;+\; \mathbb{E}\left[C_{i,j}(\boldsymbol{x})\middle|\,\mathcal{F}_{i+j-1}\right] \mathbb{1}_{\{C_{i,j-1}(\boldsymbol{x})=0\}}.$$

There exist parameters $\sigma_0^2, \ldots, \sigma_{J-1}^2 > 0$ such that for all $1 \le i \le I$, $1 \le j \le J$ and $\boldsymbol{x} \in \mathcal{X}$

$$\operatorname{Var}\left(C_{i,j}(\boldsymbol{x})\middle|\,\mathcal{F}_{i+j-1}\right) \;=\; \sigma_{j-1}^2\; C_{i,j-1}(\boldsymbol{x}) \;+\; \operatorname{Var}\left(C_{i,j}(\boldsymbol{x})\middle|\,\mathcal{F}_{i+j-1}\right) \mathbb{1}_{\{C_{i,j-1}(\boldsymbol{x})=0\}}.$$

We interpret these assumptions. There are two different situations. Either we have strictly positive cumulative claims payments $C_{i,j-1}(\boldsymbol{x}) > 0$ which implies feature based CL property

$$\mathbb{E}\left[C_{i,j}(\boldsymbol{x}) \middle| \mathcal{F}_{i+j-1} \right] = f_{j-1}(\boldsymbol{x})\ C_{i,j-1}(\boldsymbol{x}). \tag{2.3}$$

This clarifies the meaning of symbol "$\approx$" in (2.1) and (2.2). In the situation $C_{i,j-1}(\boldsymbol{x}) = 0$ we still need to specify the model, this is done in Section 4 below on "zero claims features". The situation $C_{i,j-1}(\boldsymbol{x}) = 0$ is occurring more likely if we have very detailed feature information. If we coarsen feature information by merging feature classes to, say, feature information $\boldsymbol{x}^+ \in \mathcal{X}^+$, it will eventually happen that $C_{i,j-1}(\boldsymbol{x}^+) > 0$ for all $\boldsymbol{x}^+$. In that case we have a situation that is similar to (2.3) for *all* feature values $\boldsymbol{x}^+ \in \mathcal{X}^+$ and aggregation provides

$$\mathbb{E}\left[C_{i,j} \middle| \mathcal{F}_{i+j-1} \right] = \mathbb{E}\left[\sum_{\boldsymbol{x}^+ \in \mathcal{X}^+} C_{i,j}(\boldsymbol{x}^+) \middle| \mathcal{F}_{i+j-1} \right] = \sum_{\boldsymbol{x}^+ \in \mathcal{X}^+} f_{j-1}(\boldsymbol{x}^+)\ C_{i,j-1}(\boldsymbol{x}^+).$$

Under the additional assumption that $f_{j-1}(\boldsymbol{x}^+)$ does not depend on $\boldsymbol{x}^+$, i.e., $f_{j-1}(\boldsymbol{x}^+) \equiv f_{j-1}$, we obtain the classical CL model

$$\mathbb{E}\left[C_{i,j} \middle| \mathcal{F}_{i+j-1} \right] = f_{j-1} C_{i,j-1}.$$

Note that $C_{i,j}$ always denotes the cumulative claim aggregated over all possible feature values $\boldsymbol{x} \in \mathcal{X}$.

Proposition 2.2 *Assume $I > J$. Under our standing assumptions and under Model Assumptions 2.1 we have for $i + J > I$ and $C_{i,I-i}(\boldsymbol{x}) > 0$*

$$\mathbb{E}\left[C_{i,J}(\boldsymbol{x}) \middle| \mathcal{F}_{I} \right] \ = \ C_{i,I-i}(\boldsymbol{x}) \prod_{j=I-i}^{J-1} f_j(\boldsymbol{x}).$$

Proof. The claim follows by (recursive) application of the tower property (under the standing assumptions).

□

Proposition 2.2 is identical to the classical CL property in Mack's CL model, and there only remains to estimate the regression functions $f_j : \mathcal{X} \to \mathbb{R}_+$. The zero claims features case $C_{i,I-i}(\boldsymbol{x}) = 0$ is more sophisticated. Assume $i + j = I + 2$ then we have, using the tower property,

$$\begin{aligned} \mathbb{E}\left[C_{i,I-i+2}(\boldsymbol{x}) \middle| \mathcal{F}_{I} \right] \mathbb{1}_{\{C_{i,I-i}(\boldsymbol{x})=0\}} \ &= \ f_{I-i+1}(\boldsymbol{x})\ \mathbb{E}\left[C_{i,I-i+1}(\boldsymbol{x}) \middle| \mathcal{F}_{I} \right] \mathbb{1}_{\{C_{i,I-i}(\boldsymbol{x})=0\}} \\ &\quad + \mathbb{E}\left[\mathbb{E}\left[C_{i,I-i+2}(\boldsymbol{x}) \middle| \mathcal{F}_{I+1} \right] \mathbb{1}_{\{C_{i,I-i+1}(\boldsymbol{x})=0\}} \middle| \mathcal{F}_{I} \right]. \end{aligned} \tag{2.4}$$

Note that the second line of (2.4) cannot appear in the case $C_{i,I-i}(\boldsymbol{x}) > 0$ because of the standing assumptions that we stay away from 0 once we have observed a strictly positive claim. Formula (2.4) shows that the case $C_{i,I-i}(\boldsymbol{x}) = 0$ will result in a complicated structure, we are going to treat this in more detail in Section 4, below. In the next section we consider the case (2.3).

3 Neural network modeling of chain-ladder factors

3.1 Chain-ladder factor modeling

In this section we consider feature values $\boldsymbol{x} \in \mathcal{X}$ with $C_{i,j-1}(\boldsymbol{x}) > 0$ for a given development year $1 \le j \le J$ and accident year $1 \le i \le I - j$. In this case we study models of type (2.3) and the CL factors $f_{j-1} : \mathcal{X} \to \mathbb{R}_+$ are found by minimizing a given loss function. Given observations $\mathcal{F}_I$, it is natural under our model assumptions to consider the weighted square loss functions for $1 \le j \le J$ given by

$$\begin{aligned} \mathcal{L}_j &= \sum_{i=1}^{I-j} \sum_{\boldsymbol{x}:\, C_{i,j-1}(\boldsymbol{x})>0} \frac{\left(C_{i,j}(\boldsymbol{x}) - f_{j-1}(\boldsymbol{x})\ C_{i,j-1}(\boldsymbol{x})\right)^2}{\sigma_{j-1}^2\ C_{i,j-1}(\boldsymbol{x})} \\ &= \frac{1}{\sigma_{j-1}^2} \sum_{i=1}^{I-j} \sum_{\boldsymbol{x}:\, C_{i,j-1}(\boldsymbol{x})>0} C_{i,j-1}(\boldsymbol{x}) \left(\frac{C_{i,j}(\boldsymbol{x})}{C_{i,j-1}(\boldsymbol{x})} - f_{j-1}(\boldsymbol{x}) \right)^2 . \end{aligned} \tag{3.1}$$

Here and in the sequel we require $I > J$. This implies that we have at least one observed accident year for each development period $1 \le j \le J$ in the observations $\mathcal{F}_I$. If we would not consider any feature information in f_{j-1}, then minimizing the loss function $\mathcal{L}_j$ would exactly provide Mack's classical (homogeneous) CL factor estimate for f_{j-1}, see [14].

Remark on choice (3.1). On the one hand, the weighted square loss function (3.1) is a natural choice under the conditional moment assumptions given in Model Assumptions 2.1. On the other hand, the (weighted) square loss function suggests a Gaussian model structure because it is identical to Gaussian deviance statistics. If there is an indication for the weighted square loss function not being appropriate, for instance, because of outliers in the data, an other loss function such as the gamma or the log-normal deviance statistics should be chosen. This may also require that the variance assumption in Model Assumptions 2.1 needs to be changed.

We model the CL factors $f_{j-1} : \mathcal{X} \to \mathbb{R}_+$ by feed-forward neural networks having one hidden layer (called shallow neural networks). The hidden layer is assumed to have q hidden neurons. An example with $q = 20$ is given in Figure 1, and for a general overview on neural networks we refer to LeCun et al. [12] and the references therein. Under these choices we make the following model assumptions. The hidden layer is given by the regression models (we denote the dimension of $\mathcal{X}$ by d)

$$\boldsymbol{x} \ \mapsto\ z_k(\boldsymbol{x}) = \tanh\left(w_{k,0} + \sum_{l=1}^{d} w_{k,l} x_l \right), \qquad \text{for } k = 1, \ldots, q, \tag{3.2}$$

for given weights $(w_{k,l})_{k,l}$ and for hyperbolic tangent activation function. These activations are then propagated (feed-forward) to the output layer describing the CL factor $f_{j-1} : \mathcal{X} \to \mathbb{R}_+$ by

$$\boldsymbol{x} \ \mapsto\ \log f_{j-1}(\boldsymbol{x}) = \beta_0 + \sum_{k=1}^{q} \beta_k z_k(\boldsymbol{x}). \tag{3.3}$$

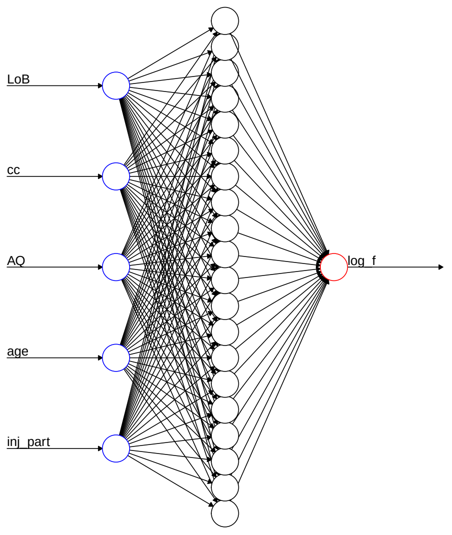

Figure 1: Shallow neural network with one hidden layer having $q = 20$ hidden neurons.

For a given number $q \in \mathbb{N}$ of hidden neurons we aim at minimizing the loss function

$$\boldsymbol{\alpha} \ \mapsto \ \mathcal{L}_j = \mathcal{L}_j(\boldsymbol{\alpha}), \tag{3.4}$$

in the neural network parameter

$$\boldsymbol{\alpha} = (\beta_0, \ldots, \beta_q, w_{1,0}, \ldots, w_{q,d})' \in \mathbb{R}^{q+1+q(d+1)}, \tag{3.5}$$

under the assumption that the CL factor $f_{j-1}(\boldsymbol{x})$ has regression structure (3.3). Every development period $1 \le j \le J$ may have its own number $q = q^{(j-1)}$ of hidden neurons and a different network parameter $\boldsymbol{\alpha} = \boldsymbol{\alpha}^{(j-1)}$. That is, these parameters may depend on $1 \le j \le J$, but for the ease of notation we omit these superscripts. Thus, each development period j will have its own neural network architecture which may be optimized over its hyperparameter q and its network parameter $\boldsymbol{\alpha}$ w.r.t. the loss function (3.4).

In Figure 1 we illustrate a shallow neural network having $q = 20$ hidden neurons. The choice of one hidden layer may seem to be arbitrary, but the universality theorems of Cybenko [6], Hornik et al. [9] and Isenbeck-Rüschendorf [10] say that every compactly supported continuous regression function can be approximated arbitrarily well (in supremum norm and L^p norm, $p \ge 1$) if we allow for arbitrarily many hidden neurons in the single hidden layer. Thus, in general, we do not need more complex network architectures to approximate a regression function. Only, neural networks with more hidden layers may have better convergence properties in calibration, we refer to Montúfar et al. [15]. Having this said, we are left with one hyperparameter of freedom, namely, we should choose q optimally (which will determine the complexity of the model). This choice may be done based on an out-of-sample validation, we come back to this below.

3.2 Gradient descent method for loss function minimization

The general aim is to minimize the loss function $\boldsymbol{\alpha} \mapsto \mathcal{L}_j(\boldsymbol{\alpha})$ in $\boldsymbol{\alpha}$ in order to find the optimal network parameter $\boldsymbol{\alpha}$ (for a given hyperparameter q). State-of-the-art uses the gradient descent algorithm which locally decreases the loss $\mathcal{L}_j$ step by step. Consider the Taylor expansion around $\boldsymbol{\alpha}$ given by

$$\mathcal{L}_j(\widetilde{\boldsymbol{\alpha}}) = \mathcal{L}_j(\boldsymbol{\alpha}) + (\nabla_{\boldsymbol{\alpha}} \mathcal{L}_j(\boldsymbol{\alpha}))' (\widetilde{\boldsymbol{\alpha}} - \boldsymbol{\alpha}) + o(\|\widetilde{\boldsymbol{\alpha}} - \boldsymbol{\alpha}\|),$$

as $\|\widetilde{\boldsymbol{\alpha}} - \boldsymbol{\alpha}\| \to 0$. We see that the locally optimal step points into the direction of the negative gradient $-\nabla_{\boldsymbol{\alpha}} \mathcal{L}_j(\boldsymbol{\alpha})$. If we choose a learning rate $\varrho > 0$ into that direction we obtain a local loss improvement of

$$\mathcal{L}_j(\boldsymbol{\alpha} - \varrho \nabla_{\boldsymbol{\alpha}} \mathcal{L}_j(\boldsymbol{\alpha})) \;\approx\; \mathcal{L}_j(\boldsymbol{\alpha}) - \varrho \|\nabla_{\boldsymbol{\alpha}} \mathcal{L}_j(\boldsymbol{\alpha})\|^2, \tag{3.6}$$

for ϱ small. Iterative application of these locally optimal steps (with tempered learning rates ϱ) will converge to the (local) minimum of the loss function. Note that different starting points of this algorithm should be explored to see whether we obtain better convergence properties; and the speed of convergence should be fine-tuned in ϱ. Remark that this optimization does not require knowledge of the variance parameter σ_{j-1}^2, since by assumption it is the same for all accident years $1 \le i \le I$ and features $\boldsymbol{x} \in \mathcal{X}$, see (3.1).

The gradient descent method (3.6) may be improved by considering higher order Taylor expansions. However, in most cases calculation of higher order derivatives is computationally not feasible. For this reason, one often applies a so-called momentum-based improved gradient descent method with momentum coefficient $\nu \in [0, 1)$. For details we refer to Rumelhart et al. [20] and Nielsen [16]. A second modification that is often implemented is that one does not use all data at once (steepest gradient descent) but one iterates the algorithm with random sub-samples of the data (mini batches). The latter is called stochastic gradient descent method and deals with the problem of big data.

There remains the calculation of the gradient. We have

$$\nabla_{\boldsymbol{\alpha}} \mathcal{L}_j(\boldsymbol{\alpha}) = -\frac{2}{\sigma_{j-1}^2} \sum_{i=1}^{I-j} \sum_{\boldsymbol{x}:\, C_{i,j-1}(\boldsymbol{x})>0} C_{i,j-1}(\boldsymbol{x}) \left(\frac{C_{i,j}(\boldsymbol{x})}{C_{i,j-1}(\boldsymbol{x})} - f_{j-1}(\boldsymbol{x}) \right) \nabla_{\boldsymbol{\alpha}} f_{j-1}(\boldsymbol{x}),$$

with

$$\nabla_{\boldsymbol{\alpha}} f_{j-1}(\boldsymbol{x}) = f_{j-1}(\boldsymbol{x})\; \nabla_{\boldsymbol{\alpha}} \left(\beta_0 + \sum_{k=1}^{q} \beta_k z_k(\boldsymbol{x}) \right).$$

For considering this last gradient, we set $\mu(\boldsymbol{x}) = \beta_0 + \sum_{k=1}^{q} \beta_k z_k(\boldsymbol{x})$. We receive the following components, see also (3.5): for $k = 0, \ldots, q$ and setting $z_0(\boldsymbol{x}) = 1$,

$$\frac{\partial \mu(\boldsymbol{x})}{\partial \beta_k} \;=\; z_k(\boldsymbol{x});$$

the derivatives w.r.t. the weights $w_{k,l}$ are given by

$$\frac{\partial \mu(\boldsymbol{x})}{\partial w_{k,l}} \;=\; \beta_k \left(1 - z_k(\boldsymbol{x})^2\right) x_l,$$

where we set $x_0 = 1$. These formulas show that the optimization problem can be solved using the back-propagation method, see Werbos [23] and Chapter 2 in Nielsen [16].

3.3 Feature pre-processing

Typically, the feature components of $\boldsymbol{x} \in \mathcal{X}$ need pre-processing, otherwise the gradient descent method may not result in a reasonable regression function. There are two problems that we have to deal with *(i)* categorical feature components should be converted to numerical ones, and *(ii)* all (continuous) feature components should live on a similar scale.

(i) Categorical feature components. We have the three unordered categorical (nominal) feature components `LoB` $\in \{1, \dots, 4\}$, `cc` $\in \{1, \dots, 53\}$ and `inj_part` $\in \{10, \dots, 99\}$. We need to map their categorical labels to numerical ones for neural network modeling. We apply dummy coding (also called one-hot encoding) to achieve this. We illustrate this for `LoB` $\in \{1, \dots, 4\}$. In dummy coding we replace the $r = 4$ different categorical labels by $(r-1)$-dimensional binary vectors indicating which label a particular feature component possesses. For feature component `LoB` we therefore consider the mapping provided in Table 1. Note that each of the 4 labels is mapped

x_1	dummy $\boldsymbol{x}_1^*$		
`LoB` 1	0	0	0
`LoB` 2	1	0	0
`LoB` 3	0	1	0
`LoB` 4	0	0	1

Table 1: Dummy coding for `LoB` $\in \{1, \dots, 4\}$.

to a 3-dimensional feature vector $\boldsymbol{x}_1^*$ in $\{0, 1\}^3$, with reference label being `LoB` 1. Typically, the label with the biggest volume is chosen as reference label. In our example, dummy coding provides for the three unordered categorical feature components a $3 + 52 + 89 = 144$ dimensional feature vector.

(ii) Continuous feature components. We should apply scaling to continuous feature components such that they all live on a comparable scale. This is important in the application of the gradient descent method in order to get reasonable calibrations. Because the activation function lives around the origin we choose bijection

$$x_l \;\mapsto\; x_l^* = 2\,\frac{x_l - \min x_l}{\max x_l - \min x_l} - 1, \tag{3.7}$$

for the two continuous feature components $l = 3, 4$, and the minimum and the maximum in (3.7) correspond to the minimal and maximal value of the corresponding feature component x_l (note that $\mathcal{X}$ is bounded). Transformation (3.7) is also called MinMaxScaler.

We have now specified all parts and we can start to fit the regression models. This is done in the next section.

3.4 Model calibration and model selection: part 1

We start by fitting the model to the data with $C_{i,j-1}(\boldsymbol{x}) > 0$; this is referred to "part 1" in this section heading. We first describe the data. The individual claims data has been generated by the scenario generator developed in [7]. The specific parameter choices of the chosen simulation are described in Listing 1 in Appendix A. Moreover, we provide selected summary statistics in that appendix. The simulation provides $4'970'856$ reported individual claims at time $I = 12$ with accident years $1 \leq i \leq I$ (for the ease of notation we have relabeled the accident years compared to Listing 1). These claims are established with feature information $\boldsymbol{x}$ and corresponding claims payment information. The claims payment information is summarized in upper triangles in Table 4 of Appendix A.

Based on this data we could now do a model and hyperparameter q selection based on an out-of-sample validation analysis. In neural network calibration, one often finds several models with a similar predictive performance, i.e., one has multiple models with a comparable predictive power. This makes it difficult to choose "a best model" and, therefore, the hyperparameter choice is often done subjectively by an expert. Typically, q should not be chosen too small because otherwise the resulting regression function might be too simple and the gradient descent method may get trapped in saddle points. That is, we should leave the algorithm sufficiently many degrees of freedom. On the other hand, a model with a large q is more prone to over-fitting. Typically, over-fitting is taken care of by stopping the gradient descent method not too late (called early stopping rule) which, of course, is more art than science. For our models, we have chosen $q = 20$ for all $1 \leq j \leq J$, and we have stopped the gradient descent method as soon as the in-sample loss did not change significantly anymore, more details are provided in Appendix B.

In Figures 8-9 we present the resulting calibrations and sensitivities: each row corresponds to a different delay $j = 1, \ldots, 11$, and the columns correspond to the sensitivities in the feature components `LoB`, `cc`, `AQ`, `age` and `inj_part`, respectively. For these plots we calculate the predicted claim amounts in the upper triangle

$$\widehat{C}_{i,j}(\boldsymbol{x}) = \widehat{f}_{j-1}(\boldsymbol{x})\ C_{i,j-1}(\boldsymbol{x}),$$

of all features $\boldsymbol{x}$ with $C_{i,j-1}(\boldsymbol{x}) > 0$, and where $\widehat{f}_{j-1}(\boldsymbol{x})$ denotes the estimated CL factor using the optimal neural network parameter $\widehat{\alpha} = \widehat{\alpha}^{(j-1)}$ for $q = 20$. This allows us to calculate the average neural network parameter (averaged over our portfolio) defined by

$$\bar{f}_{j-1}^{\mathrm{NN}} = \frac{\sum_{i=1}^{I-j} \sum_{\boldsymbol{x}} \widehat{f}_{j-1}(\boldsymbol{x}) C_{i,j-1}(\boldsymbol{x})}{\sum_{i=1}^{I-j} \sum_{\boldsymbol{x}} C_{i,j-1}(\boldsymbol{x})}. \tag{3.8}$$

These values correspond to the gray dotted lines in Figures 8-9. The sensitivities (blue graphs in Figures 8-9) are then obtained by considering marginalized versions instead of (3.8), i.e., we consider the average CL factor as a function of the different labels of one fixed feature component. As seen on the first line of Figure 8, the first development factor $\widehat{f}_0 : \mathcal{X} \to \mathbb{R}_+$ is most sensitive in the accident quarter `AQ` information, followed by the injured body part `inj_part` component. Similar conclusions can be drawn for the other development periods. In particular, we see that the injured body part is an important feature component, whereas the age of the injured seems to be less important in our data. This finishes part 1 of the calibration.

4 Zero claims features

We now turn to the case $C_{i,j-1}(\boldsymbol{x}) = 0$ called zero claims features. From formula (2.4) we see that this may result in an involved model. Therefore we are going to define a crude model for this latter case. This crude model will not consider any feature information: note that we have $|\mathcal{X}| = 4'273'920$ different possible feature values, however, for only $830'380$ of these feature values we have a positive observation $C_{i,0}(\boldsymbol{x}) > 0$ (if we consider development period $j = 1$). From this it is obvious that we cannot model all these missing features individually because it would result in too high complexity, but we need a collective model for all zero claims features $\boldsymbol{x}$, i.e., with $C_{i,j-1}(\boldsymbol{x}) = 0$.

4.1 Model for zero claims features

First note that under our standing assumptions we know that if $C_{i,k}(\boldsymbol{x})$ is strictly positive then it remains strictly positive in all subsequent development periods. Therefore, for the prediction of accident years $i > I - J$ at time I, i.e., given $\mathcal{F}_I$, we can focus in this second case on the features $\boldsymbol{x}$ with $C_{i,I-i}(\boldsymbol{x}) = 0$ (and all other (non-zero claims) features are treated in Section 3). For these zero claims features we define on the lower triangle $j + i > I$

$$C^*_{i,j} \;=\; \sum_{\boldsymbol{x}:\; C_{i,I-i}(\boldsymbol{x})=0} C_{i,j}(\boldsymbol{x}). \tag{4.1}$$

We assume that the following approximation is reasonable for $i + j > I$

$$C^*_{i,j} \;\approx\; C_{i,I-i} \prod_{l=I-i}^{j-1} g_l^{(i)}, \tag{4.2}$$

for given parameters $g^{(i)}_{I-i}, \ldots, g^{(i)}_{J-1}$. Thus, for predicting the ultimate claim $C^*_{i,J}$ of zero claims features $\boldsymbol{x}$ at time I, we choose the $\mathcal{F}_I$-measurable volume $C_{i,I-i} = \sum_{\boldsymbol{x}\in\mathcal{X}} C_{i,I-i}(\boldsymbol{x})$ and project it to the last development period J using the CL factors $g^{(i)}_{I-i}, \ldots, g^{(i)}_{J-1}$. In some sense this is similar to Schnieper [21], however, using a different measure for the exposure. Note that these CL factors $g^{(i)}_{I-i}, \ldots, g^{(i)}_{J-1}$ depend on accident year i through the fact that $\boldsymbol{x}$ is a zero claim feature at development period $I - i$ for accident year i.

4.2 Model calibration for zero claims features: part 2

In view of (4.2) there remains to estimate the CL factors $g^{(i)}_{I-i}, \ldots, g^{(i)}_{J-1}$. We therefore define the zero claims features of accident year $1 \le k \le i$ at development period $I - i$ by

$$\mathcal{X}^{(i)}_k = \left\{\boldsymbol{x} \in \mathcal{X};\; C_{k,I-i}(\boldsymbol{x}) = 0\right\}.$$

This allows us to define the total claim $C_{k,j}^{(*i)}$ in development period $j > I-i$ of these zero claims features by

$$C_{k,j}^{(*i)} = \sum_{\boldsymbol{x} \in \mathcal{X}_k^{(i)}} C_{k,j}(\boldsymbol{x}).$$

Natural estimators for the CL factors $g_{I-i}^{(i)}, \ldots, g_{J-1}^{(i)}$ are then given by

$$\widehat{g}_{I-i}^{(i)} = \frac{\sum_{k=1}^{i-1} C_{k,I-i+1}^{(*i)}}{\sum_{k=1}^{i-1} C_{k,I-i}}, \qquad \text{and for } I-i < j < J \qquad \widehat{g}_j^{(i)} = \frac{\sum_{k=1}^{I-j-1} C_{k,j+1}^{(*i)}}{\sum_{k=1}^{I-j-1} C_{k,j}^{(*i)}},$$

supposed that all denominators are positive. This then motivates the ultimate claim predictor for zero claims features at time I

$$\widehat{C}_{i,J}^{*} \; = \; C_{i,I-i} \prod_{j=I-i}^{J-1} \widehat{g}_j^{(i)}.$$

This finishes part 2 of the calibration. Now we have a fully calibrated model that allows us to calculate the claims reserves for the outstanding loss liabilities at time I.

5 Claims reserves

5.1 Neural network results versus Mack's CL reserves

Our model is now fully calibrated, in Section 3 for non-zero claims features (referred to part 1 in our derivations) and in Section 4 for zero claims features (referred part 2). The total ultimate claim predictor for accident year $i > I-J$ at time I is given by

$$\widehat{C}_{i,J} \; = \; \sum_{\boldsymbol{x} \in \mathcal{X}} C_{i,I-i}(\boldsymbol{x}) \prod_{j=I-i}^{J-1} \widehat{f}_j(\boldsymbol{x}) \; + \; C_{i,I-i} \prod_{j=I-i}^{J-1} \widehat{g}_j^{(i)},$$

where the CL factor estimates $\widehat{f}_{j-1}(\boldsymbol{x})$ are given by (3.3), using the optimal neural network parameter $\widehat{\alpha} = \widehat{\alpha}^{(j-1)}$ for all development periods $j = 1, \ldots, J = 11$ (for $q = 20$ hidden neurons). This allows us to define the neural network (NN) reserves for each accident year $i > I-J$ at time I given by

$$\widehat{R}_i^{\mathrm{NN}} \; = \; \widehat{C}_{i,J} - C_{i,I-i} \; = \; \sum_{\boldsymbol{x} \in \mathcal{X}} C_{i,I-i}(\boldsymbol{x}) \prod_{j=I-i}^{J-1} \left(\widehat{f}_j(\boldsymbol{x}) - 1 \right) \; + \; C_{i,I-i} \prod_{j=I-i}^{J-1} \widehat{g}_j^{(i)}. \tag{5.1}$$

We calculate these NN reserves $\widehat{R}_i^{\mathrm{NN}}$ and consider them for each line of business `LoB` separately. This requires that we calculate the reserves for zero claims features for each line of business separately, i.e. with `LoB` dependent CL factors $g_{I-i}^{(i)}, \ldots, g_{J-1}^{(i)}$. Moreover, we compare these NN reserves to Mack's classical CL reserves calculated for each line of business separately as well. These can be obtained by applying the CL algorithm to the four claims development triangles given in Table 4.

	line of business LoB 1			
	true	NN res.	Mack [14]	
	payments	$\widehat{R}_i^{\rm NN}$	CL res.	$\sqrt{\rm msep}$
1994	0	0	0	0
1995	1'047	996	1'049	42
1996	2'337	2'220	2'338	55
1997	3'741	3'759	3'894	66
1998	5'559	5'571	5'728	80
1999	8'314	8'059	8'272	90
2000	12'129	11'612	11'681	121
2001	17'568	16'785	16'751	182
2002	21'498	22'716	22'926	463
2003	32'053	33'232	32'827	810
2004	58'110	55'195	53'049	1'776
2005	105'980	107'022	102'827	4'278
total	**268'338**	**267'167**	**261'342**	**4'886**

	line of business LoB 2			
	true	NN res.	Mack [14]	
	payments	$\widehat{R}_i^{\rm NN}$	CL res.	$\sqrt{\rm msep}$
1994	0	0	0	0
1995	89	-257	-299	33
1996	76	237	156	95
1997	1'297	1'023	869	94
1998	1'947	1'897	1'749	293
1999	2'803	3'231	2'981	359
2000	4'716	4'984	4'658	463
2001	7'883	7'592	7'138	585
2002	10'135	12'791	11'888	690
2003	23'172	22'082	21'262	885
2004	47'917	47'679	44'010	2'586
2005	108'341	108'705	100'426	6'815
total	**208'376**	**209'964**	**194'838**	**7'699**

Table 2: True outstanding payments obtained from Table 4, resulting NN reserves $\widehat{R}_i^{\rm NN}$, Mack's CL reserves and Mack's rooted mean square error of prediction [14] at time I (in $1'000$) for LoB 1 and LoB 2.

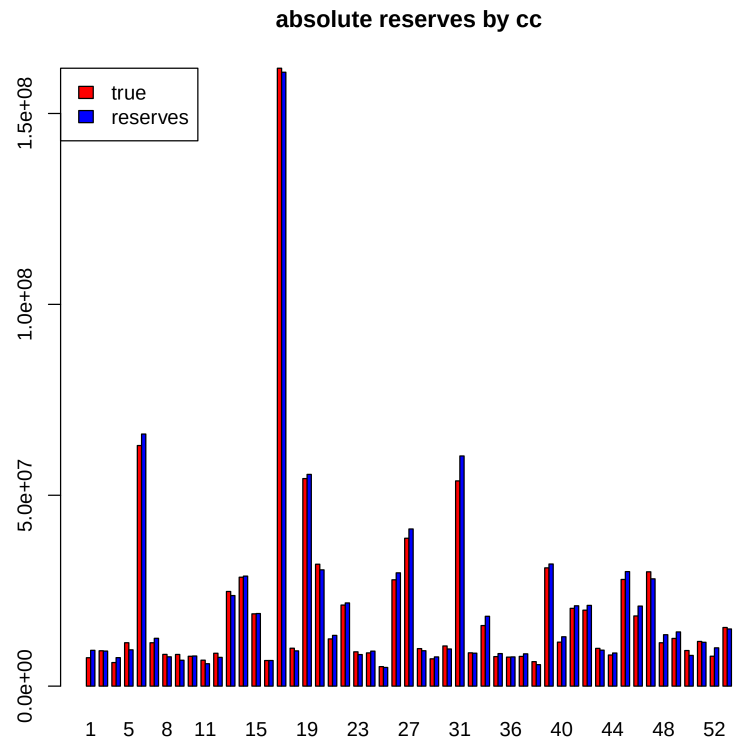

Figure 2: NN reserves for each label of the feature component cc (over all LoBs).

We present the results in Tables 2 and 3: the column 'true payments' provides the true outstanding claims payments (which, exceptionally, are available in our analysis because we have fully developed claims from the scenario generator [7]); the column 'NN res.' provides the NN reserves

	line of business `LoB` 3			
	true	NN res.	Mack [14]	
	payments	$\widehat{R}_i^{\mathrm{NN}}$	CL res.	$\sqrt{\mathrm{msep}}$
1994	0	0	0	0
1995	502	679	602	139
1996	1'534	1'593	1'532	226
1997	1'973	2'695	2'596	336
1998	3'072	4'187	3'839	386
1999	5'042	5'638	5'340	444
2000	5'833	7'414	7'556	462
2001	7'835	11'087	10'544	558
2002	14'202	15'497	16'091	655
2003	25'107	27'749	26'598	780
2004	37'804	48'819	46'574	1'623
2005	111'863	120'021	113'488	3'544
total	**214'768**	**245'378**	**234'760**	**4'736**

	line of business `LoB` 4			
	true	NN res.	Mack [14]	
	payments	$\widehat{R}_i^{\mathrm{NN}}$	CL res.	$\sqrt{\mathrm{msep}}$
1994	0	0	0	0
1995	1'600	1'562	1'620	22
1996	4'388	3'859	3'862	67
1997	6'612	6'204	6'341	197
1998	10'068	9'519	9'543	238
1999	14'040	13'942	13'947	306
2000	19'391	19'172	19'057	352
2001	23'280	25'364	25'518	509
2002	36'204	37'245	36'917	801
2003	59'007	56'141	55'400	1'326
2004	84'607	83'574	83'010	2'907
2005	165'539	162'339	154'628	5'285
total	**424'738**	**418'923**	**409'843**	**6'663**

Table 3: True outstanding payments obtained from Table 4, resulting NN reserves $\widehat{R}_i^{\mathrm{NN}}$, Mack's CL reserves and Mack's rooted mean square error of prediction [14] at time I (in $1'000$) for `LoB` 3 and `LoB` 4.

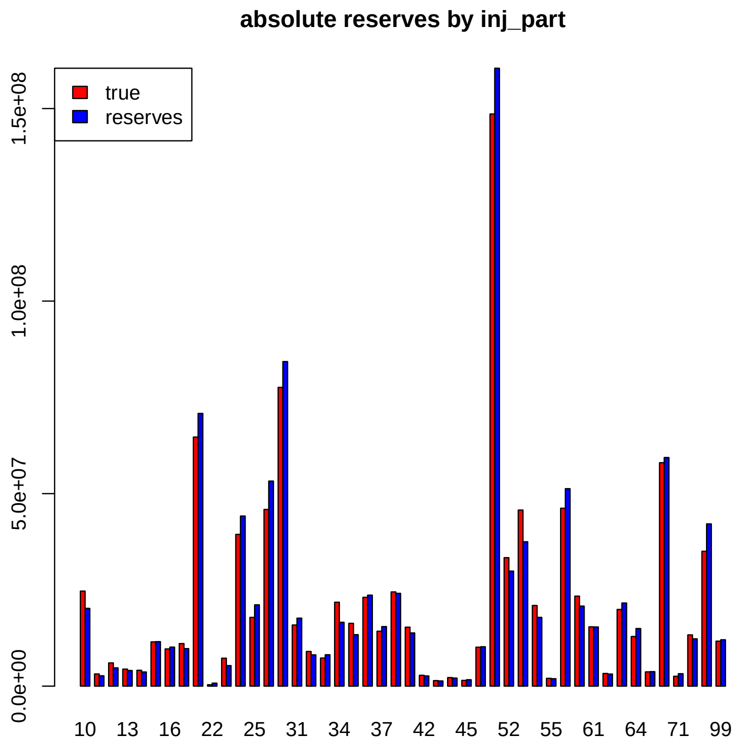

Figure 3: NN reserves for each label of the feature component `inj_part` (over all `LoB`s).

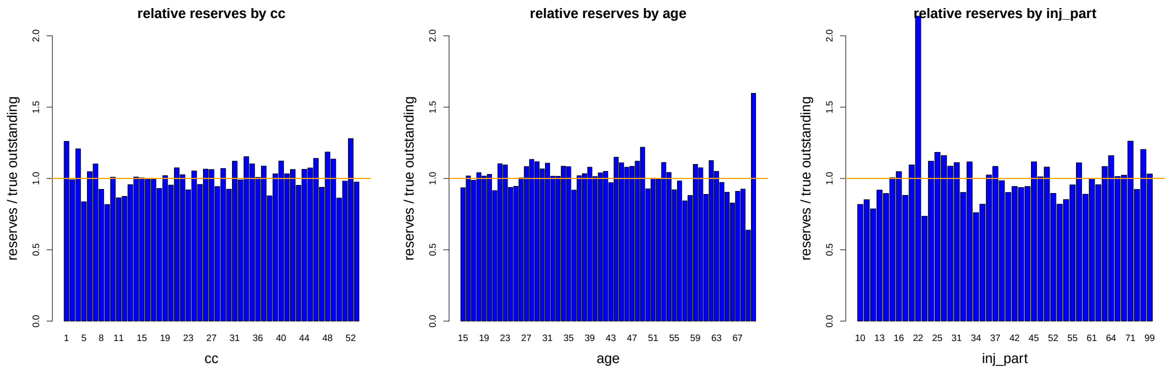

Figure 4: NN reserves relative to the true outstanding claims payments for `cc`, `age` and `inj_part` (aggregated over all `LoB`s).

(5.1); the column 'CL res.' gives the classical CL reserves; and the column '$\sqrt{\text{msep}}$' illustrates Mack's [14] rooted mean square error of prediction, which serves as a prediction uncertainty measure (related to one standard deviation). Remark that the latter uncertainty measure is the only place where we need estimates for σ_j^2, we use the standard estimates proposed by [14].

The first observation is that the resulting reserves are remarkably good. The resulting overall reserves from the neural network approach are perfect for the first two lines of business `LoB` 1 and 2: they almost completely coincide with the true outstanding claims payments, see Table 2. Mack's CL reserves are in both lines of business `LoB` 1 and 2 too low, but still within two standard deviations of the prediction uncertainty.

The results in lines of business `LoB` 3 and 4 are less accurate. In `LoB` 3, both the NN and the CL reserves are too high, beyond Mack's prediction uncertainty measure. In `LoB` 4, the NN reserves are slightly too low (but still very accurate), where as the CL reserves are too low (they deviate more than two standard deviations of prediction uncertainty from the true outstanding claims payments).

Concluding, these claims reserving results are very much in favor of the neural network regression approach. A main difference to the classical CL method is that the neural network approach considers all data simultaneously, whereas the CL method considers each line of business separately (because of the required homogeneity on CL claims reserving triangles). The latter is not always sensible: in our analysis we consider claims that belong to accident insurance and we would expect that if the same body part is injured, then the recovery process is similar regardless of the line of business affected. Of course, this idea leads to a multivariate analysis (if several feature components are involved simultaneously) which is exactly considered in the neural network regression model.

There is another major advantage of the neural network approach. We are now able to set up claims reserves for different types of claims. In Figure 2 we illustrate the NN reserves per label of the feature component `cc` and in Figure 3 we illustrate the NN reserves per label of the feature component `inj_part`. These reserves respect the sensitivities obtained in Figures 8 and 9, and the plots show that the reserves per type of claim are very accurate. In Figure 4 we provide the ratio between the NN reserves and the true outstanding claims payments per label of the feature components `cc`, `age` and `inj_part`. Again this shows that the neural network partition is very sensible because these ratios fluctuate around 1 (the few outliers are observed on labels with very small volumes). This partition on types of claims is very useful in insurance, for instance, it allows the insurance company to price different claims modules individually because it has now sensible claims reserves on each of these modules.

We close with a remark on calendar year inflation. For the CL method being appropriate it needs to be assumed that calendar year inflation is (approximately) constant over all calendar years under consideration. If this is not the case, the CL method will result in a bias. For this reason, betimes, claims are deflated before applying the CL method. Our analysis does not consider this, i.e., it uses a roughly constant calendar year inflation assumption over time. The neural network approach would allow us to include calendar year inflation. This would require an additional feature component for the calendar year in which a particular payment has been done (i.e. a feature component $x_l = i + j$ for a payment of accident year i with a development delay j). However, we recommend to only use this approach with restraint because it will also require that we extrapolate a calendar year inflation pattern over the next $J = 11$ development years (in the lower triangle), where a constant inflation often provides the best predictions.

6 Conclusions and outlook

We have extended Mack's CL model for claims reserving to include individual claims feature information. This extension was done by using a neural network model for the CL factors, i.e., we have augmented the homogeneous CL factors by considering (heterogeneous) individual claims feature information. This extended model has the advantage that we can analyze reserves on individual claims, which allows us, for instance, to capture changing portfolio mixes.

We admit that our work is only the starting point of a much broader modeling task and several next steps have to be considered. We mention two of them. Our analysis only considers static feature information, i.e., the feature labels do not change over time. The modeling task becomes much more complex for dynamic feature components, because dynamic features will require modeling of multidimensional stochastic processes. An example of a dynamic feature component is a disability degree that may change over time. A first attempt to treat such a dynamic feature component has been done in [24]. Moreover, in the dynamic approach one may also use other time scales than an annual grid. Of course, this choice will be closely related to the available data.

Another issue which goes beyond this work is prediction uncertainty. Triggers of prediction uncertainty are model uncertainty and parameter estimation error. In many situations this is dealt with by doing a bootstrap or bagging analysis. At the moment, these techniques are not feasible for several reasons in our neural network approach: the computational time is still too large, see Appendix B, and it is unclear how we should deal with model uncertainty, in particular, in the situation of over-parametrized models that are prone to over-fitting and contain potential redundancies. For these reasons alternatives need to be found to analyze prediction uncertainty. A straightforward idea is to use Bayesian methods, however, also here we face the curse of dimensionality which may result in Markov chain Monte Carlo simulations that are far too time consuming.

Acknowledgment. We would like to kindly thank Philipp Reinmann (AXA) and Ronald Richman (AIG) who have provided very useful remarks on previous versions of this manuscript.

References

[1] Antonio, K., Plat, R. (2014). Micro-level stochastic loss reserving for general insurance. *Scandinavian Actuarial Journal* **2014/7**, 649-669.

[2] Arjas, E. (1989). The claims reserving problem in non-life insurance: some structural ideas. *ASTIN Bulletin* **19/2**, 139-152.

[3] Badescu, A.L., Lin, X.S., Tang, D. (2016). A marked Cox model for the number of IBNR claims: theory. *Insurance: Mathematics & Economics* **69**, 29-37.

[4] Badescu, A.L., Lin, X.S., Tang, D. (2016). A marked Cox model for the number of IBNR claims: estimation and application. Version March 14, 2016. SSRN Manuscript 2747223.

[5] Baudry, M., Robert, C.Y. (2017). Non parametric individual claim reserving in insurance. Preprint.

[6] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)* **2/4**, 303-314.

[7] Gabrielli, A., Wüthrich, M.V. (2018). An individual claims history simulation machine. *Risks* **6/2**, 29.

[8] Harej, B., Gächter, R., Jamal, S. (2017). Individual claim development with machine learning. ASTIN Report.

[9] Hornik, K., Stinchcombe, M., White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks* **2/5**, 359-366.

[10] Isenbeck, M., Rüschendorf, L. (1992). Completeness in location families. *Probability and Mathematical Statistics* **13**, 321-343.

[11] Jessen, A.H., Mikosch, T., Samorodnitsky, G. (2011). Prediction of outstanding payments in a Poisson cluster model. *Scandinavian Actuarial Journal* **2011/3**, 214-237.

[12] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. *Nature* **521/7553**, 436-444.

[13] Lopez, O. (2018). A censored copula model for micro-level claim reserving. *HAL* Id: hal-01706935.

[14] Mack, T. (1993). Distribution-free calculation of the standard error of chain ladder reserve estimates. *ASTIN Bulletin* **23/2**, 213-225.

[15] Montúfar, G., Pascanu, R., Cho, K., Bengio, Y. (2014). On the number of linear regions of deep neural networks. *Neural Information Processing Systems Proceedings*$^\beta$ **27**, 2924-2932.

[16] Nielsen, M. (2017). *Neural Networks and Deep Learning.* Online book available on http://neuralnetworksanddeeplearning.com

[17] Norberg, R. (1993). Prediction of outstanding liabilities in non-life insurance. *ASTIN Bulletin* **23/1**, 95-115.

[18] Norberg, R. (1999). Prediction of outstanding liabilities II. Model variations and extensions. *ASTIN Bulletin* **29/1**, 5-25.

[19] Pigeon, M., Antonio, K., Denuit, M. (2013). Individual loss reserving with the multivariate skew normal framework. *ASTIN Bulletin* **43/3**, 399-428.

[20] Rumelhart, D.E., Hinton, G.E., Williams, R.J. (1986). Learning representations by back-propagating errors. *Nature* **323/6088**, 533-536.

[21] Schnieper, R. (1991). Separating true IBNR and IBNER claims. *ASTIN Bulletin* **21/1**, 111-127.

[22] Verrall, R.J., Wüthrich, M.V. (2016). Understanding reporting delay in general insurance. *Risks* **4/3**, 25.

[23] Werbos, P. (1982). Applications of advances in nonlinear sensitivity analysis. *System Modeling and Optimization* **1982**, 762-770.

[24] Wüthrich, M.V. (2018). Machine learning in individual claims reserving. *Scandinavian Actuarial Journal* **2018/6**, 465-480.

[25] Zarkadoulas, A. (2017). Neural network algorithms for the development of individual losses. MSc thesis, University of Lausanne.

A Data

We generate synthetic individual claims data using the scenario generator provided in [7]. The

Listing 1: Simulating individual claims histories using the scenario generator [7].

```
 1  > source(file="./Functions.V1.R")
 2  > V <- 5000000
 3  > LoB.dist <- c(0.35,0.20,0.15,0.30)
 4  > growth <- c(0,0,0,0)
 5  > seed1 <- 100
 6  > features <- Feature.Generation(V=V,LoB.dist=LoB.dist,inflation=growth,seed1=seed1)
 7  > str(features)
 8  'data.frame':   5003204 obs. of  7 variables:
 9   $ ClNr    : int  1 2 3 4 5 6 7 8 9 10 ...
10   $ LoB     : Factor w/ 4 levels "1","2","3","4": 3 2 4 2 1 1 2 4 1 4 ...
11   $ cc      : Factor w/ 51 levels "1","3","4","5",..: 16 5 4 16 16 5 16 19 28 25 ...
12   $ AY      : num  1994 1994 1994 1994 1994 ...
13   $ AQ      : num  1 3 1 2 2 1 4 2 3 3 ...
14   $ age     : num  67 39 55 22 38 16 36 45 29 45 ...
15   $ inj_part: Factor w/ 46 levels "10","11","12",..: 31 26 6 34 31 36 9 15 21 41 ...
16
17  > npb <- nrow(features)
18  > seed1 <- 100
19  > sd1 <- 0.85
20  > sd2 <- 0.85
21  > output <- Simulation.Machine(features=features,npb=npb,seed1=seed1,std1=sd1,std2=sd2)
```

use and parametrization of this scenario generator is illustrated in Listing 1. On lines 2-6 we generate a claims portfolio consisting of feature components `LoB`, `cc`, `AY`, `AQ`, `age` and `inj_part`, see lines 8-15 of Listing 1. We receive 5'003'204 individual claims. The marginal distributions of these feature components are illustrated in Figure 5 and the two-dimensional contour plots are given in Figure 6. The latter contour plots do not show co-linearity in feature components. This portfolio of 5'003'204 individual claims features is then used to generate fully developed individual claims histories. This is done on lines 17-21 of Listing 1 and provides an individual reporting delay `RepDel` as well as claims payments over 12 development years for each claim. In claims reserving problems, we typically do not have information about the full claims developments. If the latest observed calendar year is 2005, we only have information about the reported claims with $\texttt{AY} + \texttt{RepDel} \leq 2005$ (upper triangles). In our synthetic data set, these are 4'970'856 reported claims, and the remaining 32'348 claims are only reported after calendar year 2005. Therefore, these latter claims are not available at the end of 2005. The 4'970'856 reported claims provide claims payments in upper claims reserving triangles. These payments aggregated per line of business $\texttt{LoB} \in \{1, \ldots, 4\}$ are shown in Table 4. In our special situation of simulated claims histories we also know the total ultimate claim amounts $C_{i,J}$. These are provided in the last columns of Table 4. The general aim is to predict these last columns based on the given upper claims development triangles illustrated in Table 4.

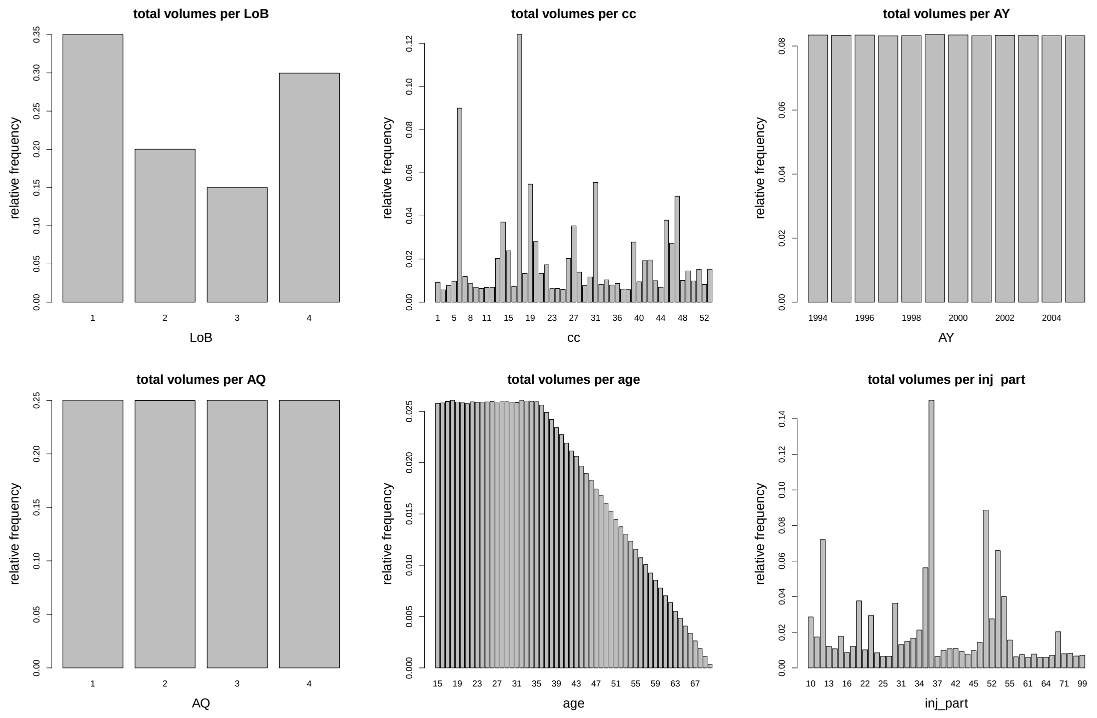

Figure 5: Marginal distributions of the feature components `LoB`, `cc`, `AY`, `AQ`, `age`, `inj_part`.

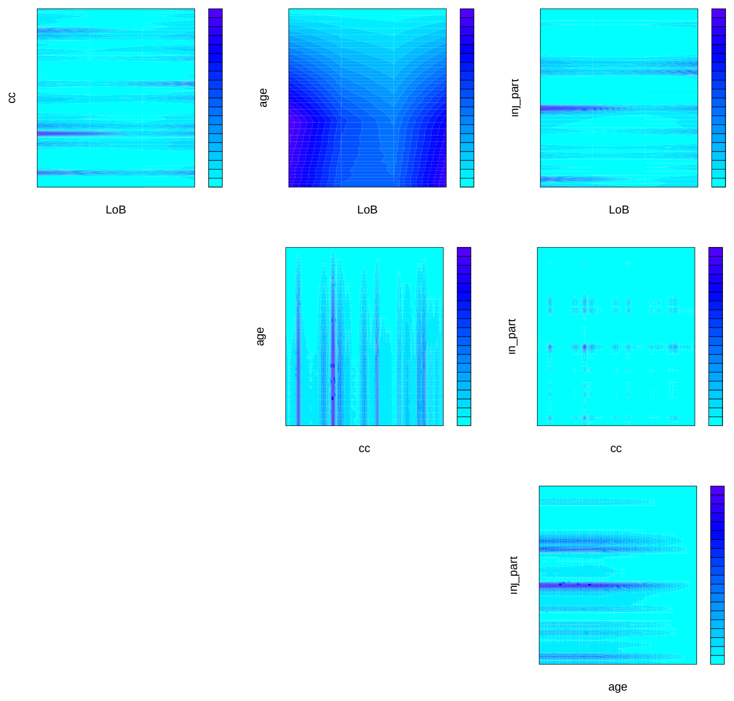

Figure 6: Two-dimensional contour plots of the portfolio distributions of the feature components `LoB`, `cc`, `age`, `inj_part`.

LoB 1	0	1	2	3	4	5	6	7	8	9	10	11	$C_{i,J}$
1994	70’866	106’683	118’701	125’633	130’217	133’723	136’347	138’459	140’124	141’549	142’816	143’832	143’832
1995	71’258	108’859	121’983	129’482	134’473	138’133	140’959	143’141	144’837	146’350	147’618		148’666
1996	71’025	109’063	122’152	129’612	134’535	138’104	140’884	143’140	144’869	146’324			148’661
1997	71’568	110’178	123’960	131’987	137’309	141’008	143’809	146’035	147’717				151’458
1998	72’721	111’689	125’837	134’012	139’489	143’408	146’351	148’628					154’187
1999	73’396	114’401	129’591	138’254	143’965	148’013	151’104						159’418
2000	75’343	118’043	134’051	143’159	149’221	153’477							165’606
2001	77’976	124’061	141’749	151’756	158’459								176’027
2002	79’531	125’451	142’721	152’469									173’967
2003	81’355	128’111	145’697										177’750
2004	85’372	138’105											196’215
2005	88’805												194’785

LoB 2	0	1	2	3	4	5	6	7	8	9	10	11	$C_{i,J}$
1994	102’186	148’628	164’152	171’223	174’861	177’084	178’603	179’583	180’381	181’147	181’653	181’357	181’357
1995	98’294	145’309	162’863	170’519	174’620	177’275	179’332	180’769	182’002	182’817	183’222		183’311
1996	96’593	143’401	160’746	168’511	172’492	174’754	176’428	177’534	178’154	178’950			179’026
1997	90’397	132’134	147’124	154’363	158’585	161’291	162’861	164’078	164’853				166’150
1998	90’203	137’358	155’447	163’426	167’388	169’276	170’733	171’796					173’743
1999	92’431	141’721	160’717	168’701	172’552	174’786	176’043						178’846
2000	91’363	142’683	162’677	170’766	174’711	177’092							181’808
2001	92’501	144’275	164’144	172’677	176’933								184’816
2002	97’416	152’027	173’579	182’476									192’612
2003	101’666	158’590	181’086										204’259
2004	106’343	168’333											216’249
2005	108’877												217’218

LoB 3	0	1	2	3	4	5	6	7	8	9	10	11	$C_{i,J}$
1994	88’735	140’690	158’553	167’405	172’571	175’914	178’285	180’068	181’654	182’675	183’724	184’325	184’325
1995	86’903	141’337	160’678	168’916	173’832	176’827	179’108	180’799	182’128	183’155	183’998		184’500
1996	85’846	138’819	157’035	165’845	170’977	174’339	176’497	178’412	179’719	181’112			182’647
1997	84’424	137’015	154’629	162’761	167’257	170’343	172’427	173’732	174’847				176’820
1998	80’764	132’191	150’684	159’278	164’040	167’222	169’495	171’082					174’155
1999	80’213	130’047	147’850	156’333	161’001	164’067	166’289						171’331
2000	81’435	131’821	150’546	158’948	163’485	165’923							171’756
2001	77’781	130’723	150’625	158’874	163’024								170’859
2002	83’479	138’681	159’421	168’147									182’348
2003	91’806	149’656	170’513										195’620
2004	91’127	147’471											185’275
2005	99’324												211’186

LoB 4	0	1	2	3	4	5	6	7	8	9	10	11	$C_{i,J}$
1994	80’645	129’535	148’090	159’028	166’099	171’395	175’621	178’956	181’585	183’937	185’961	187’579	187’579
1995	79’549	129’207	147’986	158’932	166’338	171’668	175’858	179’179	181’899	184’243	186’201		187’801
1996	82’102	135’243	156’091	168’427	176’687	182’746	187’385	191’093	194’131	196’916			201’304
1997	80’330	132’219	152’781	165’008	173’219	179’216	183’704	187’486	190’459				197’071
1998	80’761	135’037	157’073	169’781	178’602	185’059	189’949	193’868					203’937
1999	84’814	141’255	164’606	178’268	187’585	194’202	199’254						213’294
2000	83’936	141’788	166’003	180’069	189’424	196’222							215’613
2001	85’202	142’611	166’530	180’329	189’252								212’533
2002	89’256	150’820	177’637	193’228									229’433
2003	96’335	163’538	192’737										251’745
2004	98’797	166’962											251’569
2005	103’556												269’096

Table 4: Cumulative payments (in $1'000$) of `LoB`s 1-4 for accident years $1994 \leq i \leq 2005$ and development years $0 \leq j \leq J = 11$; the last column gives the true ultimate claims $C_{i,J}(\cdot)$ for each accident year $\texttt{AY} = i \in \{1994, \ldots, 2005\}$ and line of business $\texttt{LoB} \in \{1, \ldots, 4\}$.

B Neural network architecture and calibration

We come back to the choice of the network architecture and to the network calibration mentioned in Section 3.4. In our case, the network architecture requires the choice of the number of hidden neurons q (we also remind of the universality theorems mentioned in Section 3.1). The first step in the neuron activation (3.2) is the scalar product

$$\langle \boldsymbol{w}_k, \boldsymbol{x} \rangle = \sum_{l=1}^{d} w_{k,l} x_l,$$

that projects the d-dimensional feature $\boldsymbol{x}$ to a scalar for each neuron $1 \leq k \leq q$. All features $\boldsymbol{x}$ that lie in hyperplanes orthogonal to $\boldsymbol{w}_k$ give the same neuron activation $z_k(\boldsymbol{x})$ in the k-th neuron. This implies that each neuron provides a substantial reduction in dimension at the price of a loss of information. Therefore, we need an appropriate minimal number q of hidden neurons (with different $\boldsymbol{w}_k$'s) so that we are still able to capture in the hidden layer the main differences between the claim types. On the other hand, q should not be chosen too large because this acts negatively on computational time, leads to proneness to over-fitting, and provides more model redundancies (if early stopping in calibration is applied). For these reasons, people often perform a grid search to find a good hyperparameter q. Our experience is that a hyperparameter q that is roughly 1 to 3 times larger than the dimension d of the feature space is often a good choice, if the wanted regression function should not look too wildly.

Listing 2: R script for fitting networks in Keras.

```
library(keras)
# we fit j=0 with feature matrix dat.X, claims dat$C1 and volumes dat$C0
dat.Y <- as.matrix(dat$C1/sqrt(dat$C0))          # observed responses
dat.W <- as.matrix(sqrt(dat$C0))                 # volumes used as offsets
q <- 20                                          # number of hidden neurons
# definition of neural network
features <- layer_input(shape=c(ncol(dat.X)))
net <- features %>%
     layer_dense(units = q, activation = 'tanh') %>%
     layer_dense(units = 1, activation = k_exp)
volumes <- layer_input(shape=c(1))
offset <- volumes %>%
     layer_dense(units = 1, activation = 'linear', use_bias=FALSE, trainable=FALSE,
                                                  weights=list(array(1, dim=c(1,1))))
merged <- list(net, offset) %>%
     layer_multiply()
model <- keras_model(inputs=list(features, volumes), outputs=merged)
model %>% compile(loss = 'mse', optimizer = 'rmsprop')
# fit neural network
fit <- model %>% fit(list(dat.X, dat.W), dat.Y, epochs=100, batch_size=10000,
                                                            validation_split=0.1)
# predict claims dat$C1 and in-sample loss
dat$pred <- as.vector(model %>% predict(list(dat.X, dat.W)))*sqrt(dat$C0)
sum((dat$C1 - dat$pred)^2/dat$C0)
```

With this in mind we fit the model. We use the R interface to Keras which is a user-friendly application programming interface (API) to TensorFlow. The corresponding code is provided in

Listing 2. The first remark is that among the available loss functions in Keras there is the square loss function (called '`mse`', see line 18 of Listing 2), but not a weighted square loss function. For this reason, we modify (3.1) to

$$\mathcal{L}_j^0 \;=\; \sigma_{j-1}^2 \mathcal{L}_j \;=\; \sum_{i=1}^{I-j} \sum_{\boldsymbol{x}:\; C_{i,j-1}(\boldsymbol{x})>0} \left(\frac{C_{i,j}(\boldsymbol{x})}{\sqrt{C_{i,j-1}(\boldsymbol{x})}} - f_{j-1}(\boldsymbol{x}) \sqrt{C_{i,j-1}(\boldsymbol{x})} \right)^2. \tag{B.1}$$

We remark that the explicit choice of $\sigma_{j-1}^2 > 0$ does not influence the calibration. Therefore, it can be dropped. Secondly, the loss function $\mathcal{L}_j^0$ on the right-hand side of (B.1) is now an un-weighted square loss function, and we can use '`mse`' on line 18 of Listing 2. The responses are given by $Y_{i,j}(\boldsymbol{x}) = C_{i,j}(\boldsymbol{x})/\sqrt{C_{i,j-1}(\boldsymbol{x})}$, and the corresponding regression function reads as

$$\boldsymbol{x} \;\mapsto\; f_{j-1}^0(\boldsymbol{x}) \;=\; f_{j-1}(\boldsymbol{x})\sqrt{C_{i,j-1}(\boldsymbol{x})} \;=\; \exp\left\{ \beta_0 + \sum_{k=1}^{q} \beta_k z_k(\boldsymbol{x}) + \frac{1}{2} \log C_{i,j-1}(\boldsymbol{x}) \right\}, \tag{B.2}$$

see (3.3), and where $\frac{1}{2}\log C_{i,j-1}(\boldsymbol{x})$ plays the role of an offset. The responses $Y_{i,j}(\boldsymbol{x})$ are defined on line 3 of Listing 2 and the offsets (on the exponential scale) are given on line 4. Lines 7-17 then define the regression function: on lines 7-10 we build network (3.3) with q hidden neurons and excluding the offset. On lines 11-14 we build the offset part. On lines 15-17 we merge the two parts to regression function (B.2). On line 18 we compile the model using the square loss function '`mse`' and the optimizer '`rmsprop`'. Note that there are different available optimizers in Keras which are different versions of the gradient descent method. '`rmsprop`' stands for 'root mean square propagation' and it is momentum-based improved version of the gradient descent method with integrated optimal learning rates ϱ and momentum coefficients ν.

Finally, the model is fitted on lines 20-21 of Listing 2. We choose mini batches of size 10'000, i.e. every gradient descent step is based on 10'000 observations. We run these gradient descent steps for 100 epochs which means that every claim is considered 100 times during the optimization. Finally, we choose `validation_split=0.1` which means that 10% of the data is used for out-of-sample validation.

In Figure 7 we show the decrease in the loss functions $\mathcal{L}_j^0$ during the 100 epochs of the gradient descent algorithm. The left-hand side shows the model with $q = 5$ hidden neurons and the right-hand side the model with $q = 20$ hidden neurons; the red color shows the in-sample loss and the green color the validation loss on the 10% of observations chosen for out-of-sample validation. We observe in all cases a decrease in loss function which indicates that we do not over-fit by running the gradient descent algorithm in Listing 2 for 100 epochs on mini batches of size 10'000. We could now fine-tune these calibrations for the number of epochs, the mini batch sizes, the optimizer, etc. We refrain from doing so, but run the algorithm in Listing 2 for 100 epochs on mini batches of size 10'000. We do this for neural networks with $q = 5, 10, 20$ hidden neurons in the single hidden layer.

The results are presented in Table 5. The first block of Table 5 shows the results in the homogeneous case which corresponds to Mack's CL model. This model has one parameter (CL factor) for each development period $j = 1, \ldots, 11$ and results in the in-sample losses given on the last line of the first block. The remaining three blocks give the neural network results for

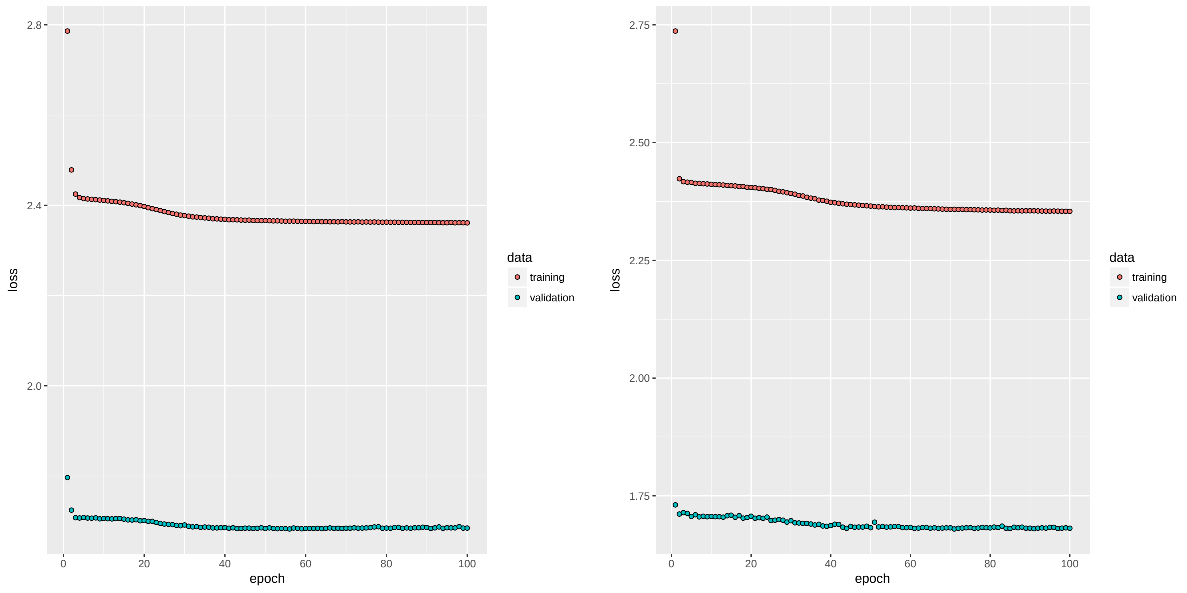

Figure 7: Decrease in loss function during the gradient descent algorithm for $j = 1$: in-sample loss in red color and validation loss in green color for the 100 epochs; the left-hand side shows the model with $q = 5$ hidden neurons and the right-hand side the model with $q = 20$ hidden neurons.

	development period j										
	1	2	3	4	5	6	7	8	9	10	11
homogeneous model											
number of parameters	1	1	1	1	1	1	1	1	1	1	1
losses $\mathcal{L}_j^0$ (in millions)	2255.9	252.9	51.4	24.6	16.9	8.7	5.9	2.0	1.3	1.2	0.4
$q = 5$ hidden neurons											
number of parameters	511	511	511	511	511	511	511	511	511	511	511
run time (in seconds)	116	125	120	107	100	93	84	74	63	48	30
losses $\mathcal{L}_j^0$ (in millions)	1906.5	224.8	46.4	22.8	16.0	8.3	5.7	1.9	1.2	1.1	0.4
$q = 10$ hidden neurons											
number of parameters	1021	1021	1021	1021	1021	1021	1021	1021	1021	1021	1021
run time (in seconds)	122	132	129	112	105	97	89	78	67	52	32
losses $\mathcal{L}_j^0$ (in millions)	1901.8	224.5	46.7	22.8	16.0	8.3	5.7	1.9	1.2	1.1	0.4
$q = 20$ hidden neurons											
number of parameters	2041	2041	2041	2041	2041	2041	2041	2041	2041	2041	2041
run time (in seconds)	138	150	135	127	121	111	101	89	76	59	36
losses $\mathcal{L}_j^0$ (in millions)	1892.9	224.1	46.6	22.8	16.1	8.2	5.7	1.9	1.2	1.1	0.4

Table 5: Results of the gradient descent algorithm in Listing 2 for $q = 5, 10, 20$ hidden neurons after 100 epochs with mini batch of 10'000 observations.

$q = 5, 10, 20$ hidden neurons. The first lines in these blocks provide the number of network parameters involved, i.e. the dimension $q + 1 + q(d + 1)$ of the network parameters $\boldsymbol{\alpha}$. The second lines in these blocks provide the run times for 100 epochs. These run times were obtained

on a personal laptop with CPU @ 2.50GHz (4 CPUs) with 16GB RAM. We observe that the run times increase in the number of hidden neurons q and they decrease in the development periods j because we have less accident years and observations in later development years. The maximal run time of 150 seconds has been observed for $q = 20$ and $j = 2$, thus, if one parallelizes optimizations for the development periods it takes roughly 3 minutes to fit the model.
Finally, the last lines of the blocks provide the resulting in-sample losses $\mathcal{L}_j^0$ of the three network models. Firstly, the neural network regression models provide clearly lower losses than the homogeneous model. From this we conclude that we should refine Mack's CL model for heterogeneity. Secondly, for periods $j = 1, 2$ the model with $q = 20$ hidden neurons outperforms the other two network models. For higher development periods $j \geq 3$ the situation is less clear and we could also opt for a regression model with less hidden neurons.

C Sensitivities of neural network calibrations

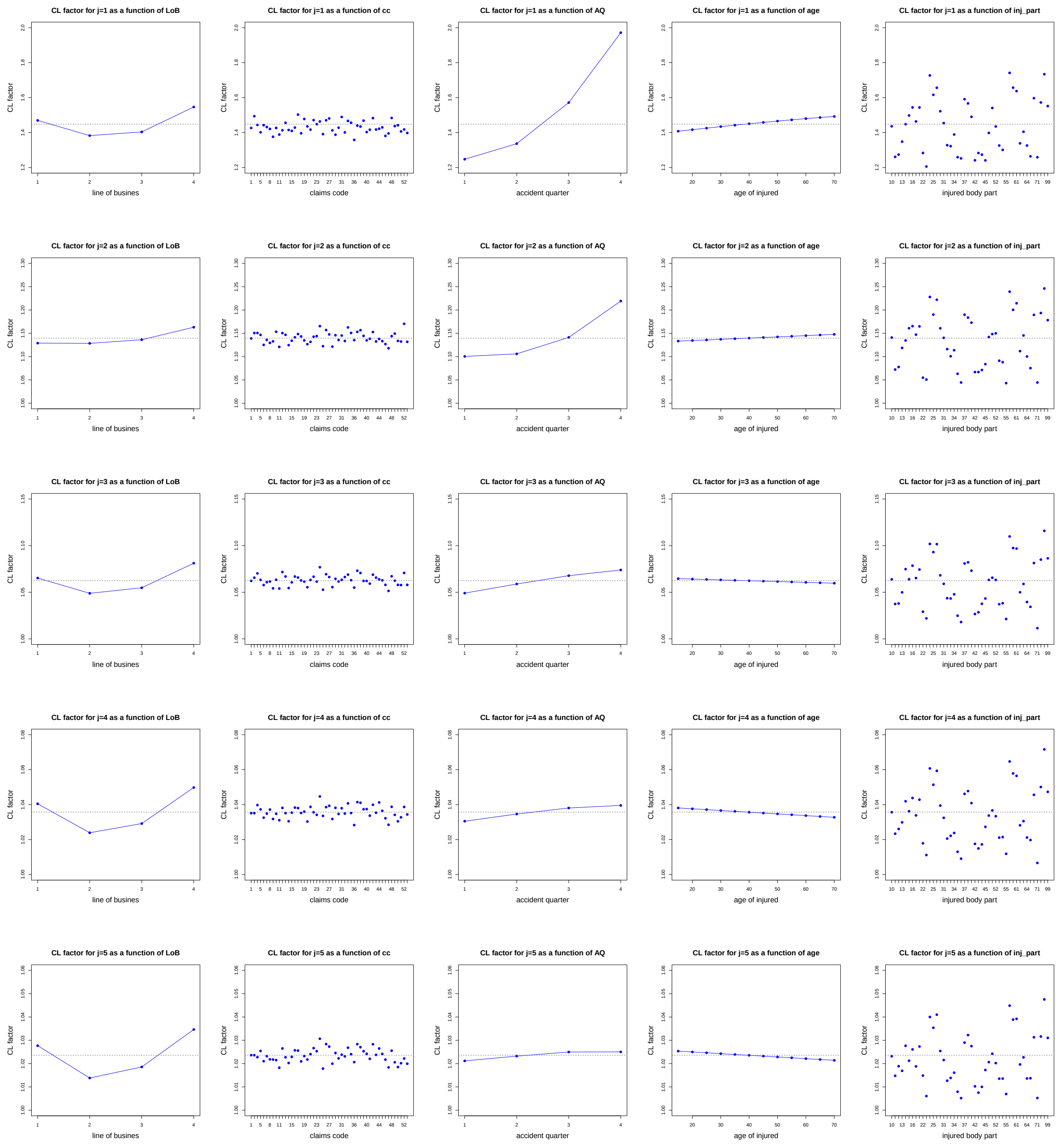

Figure 8: Sensitivities of the estimated CL factors $\widehat{f}_{j-1}(\boldsymbol{x})$ in individual feature components for $j = 1, \ldots, 5$ (in blue color); the gray dotted lines show the average neural network factor $\bar{f}_{j-1}^{\mathrm{NN}}$, see (3.8); the y-scales are same on each row.

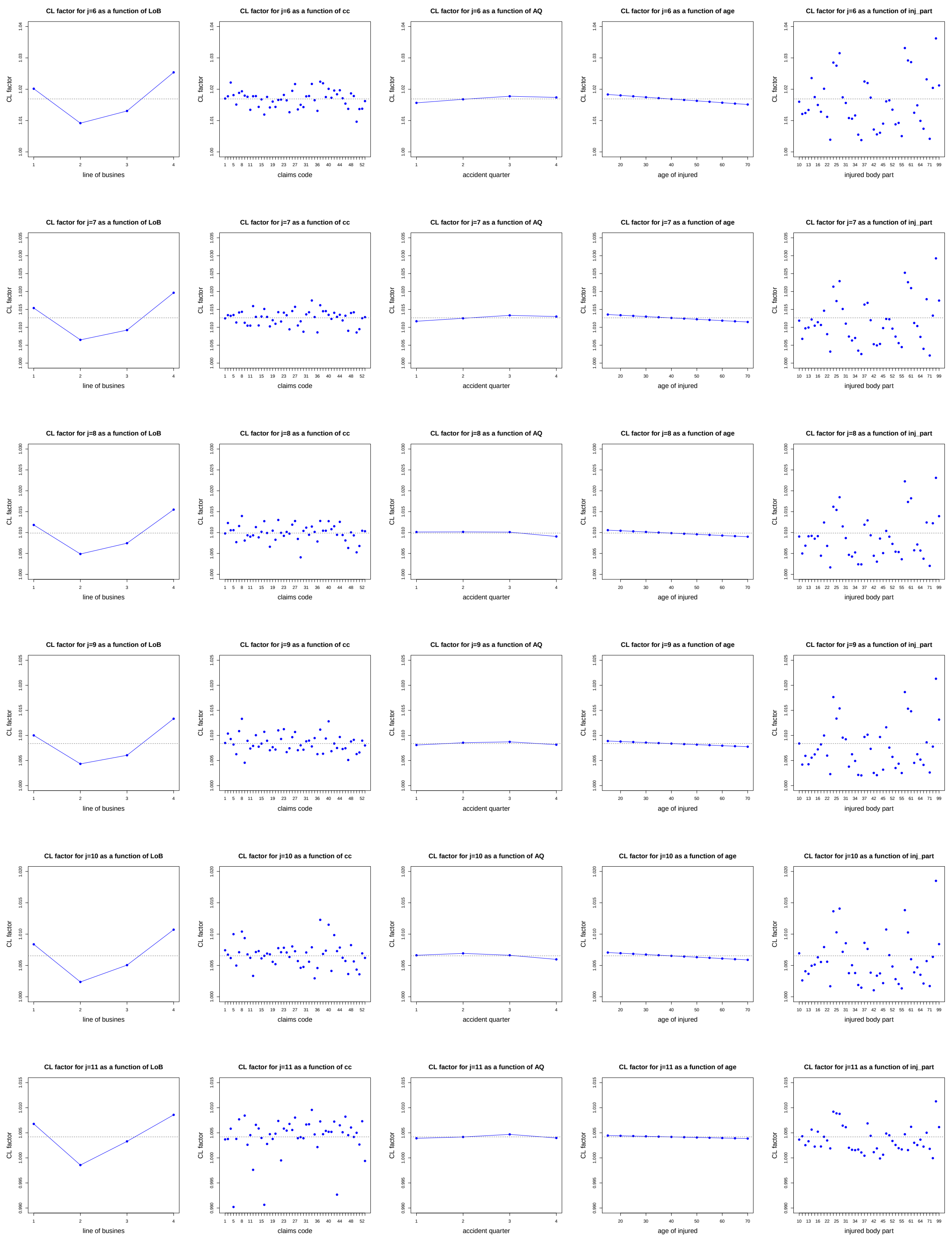

Figure 9: Sensitivities of the estimated CL factors $\widehat{f}_{j-1}(\boldsymbol{x})$ in individual feature components for $j = 6, \ldots, 11$ (in blue color); the gray dotted lines show the average neural network factor $\bar{f}^{\mathrm{NN}}_{j-1}$, see (3.8); the y-scales are the same on each row.